뇌를 자극하는

머리 속에 통째로 넣어 드리겠습니다

PHP Programming

PHP

프로그래밍

사소한 궁금증도 놓치지 않는 친절한 설명 | 가장 기본적인 형태에서 점차 업그레이드되는 모습을 보여주는 게시판 예제

조영진 저

★ 특별 온라인 학습 사이트 제공 ★
http://brain.hanb.co.kr/php

한빛미디어

이책을 쓸 당시에는 20대였지만 어느덧 어여쁜 두 딸을 둔 중년이 된
조명진 드림

Chapter

01

PHP 입문

만약 가족이나 연인과 함께하는 행복한 여행을 계획하고 있다면 서점이나 도서관으로 찾아갈 필요 없이 집에 앉아서 인터넷을 이용하여 간단히 여행지의 정보를 얻을 수 있습니다. 여행지의 관광 명소나 다양한 먹을거리 정보, 이미 그곳을 다녀간 사람들의 다양한 여행 경험담까지 관광 관련 사이트나 블로그를 통해서 접할 수 있습니다. 이처럼 웹의 역사는 길지 않지만, 어느덧 우리의 삶에서 꼭 필요한 존재로 자리 잡았습니다.

PHP는 이런 웹을 개발하는 매우 매력적인 웹 프로그래밍 언어로 매우 간단하게 웹 프로그램을 개발할 수 있습니다. C와 Perl의 장점을 모아놓은 듯한 강력하면서도 다루기 쉬운 PHP로 ActiveX와 같은 다른 외부의 컴포넌트 없이도 훌륭하게 웹 사이트를 구축할 수 있습니다. 또한, 플랫폼에 독립적인 특성 덕분에 윈도우와 리눅스 등 다양한 플랫폼에서 PHP 프로그램이 동작하며 약간의 수정이나 혹은 수정이 전혀 없이도 다른 플랫폼으로 옮기는 것이 가능합니다. 더욱 놀라운 것은 이러한 다양한 장점에도 "무료"라는 점입니다.

이 장에서는 PHP를 시작하기에 앞서 PHP가 동작하는 환경에 대해 알아보고 PHP를 개발할 수 있는 환경을 만들어 보겠습니다.

Section 01

World Wide Web

우리는 정보의 바다를 항해하기 위해서 웹 브라우저를 사용합니다. 웹 브라우저의 주소창에 도메인을 입력하고 엔터키 혹은 이동 버튼을 클릭하면 웹 브라우저는 해당 웹 사이트 정보를 웹 브라우저에 보여줍니다. 이렇듯 몇 가지 간단한 사용법을 숙지하면 웹 브라우저를 통해서 남녀노소 누구나 인터넷 세상을 자유롭게 누빌 수 있습니다. 우리는 어떻게 웹 브라우저를 통해서 정보를 얻을 수 있을까요? 웹을 그저 즐기는 사람이라면 내부 구조에 대해서까지 알 필요가 없겠지만 우리는 웹을 개발하는 웹 프로그래머가 되길 원하는 사람이니 웹 브라우저의 동작원리와 웹 서비스의 구조에 대해서 자세히 알아야 할 필요가 있습니다.

지금 바로 네이버에 접속해보자!

[그림 1-1] 웹 브라우저를 이용하여 웹 사이트에 접속

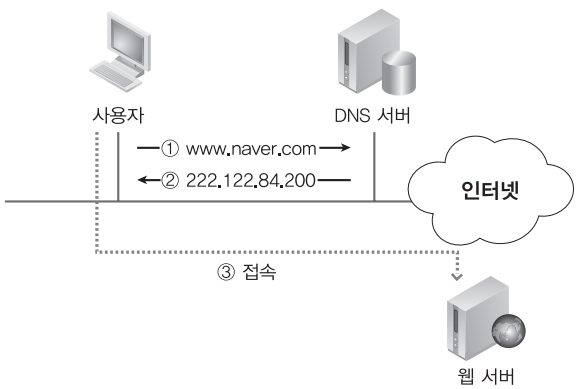
만약 인터넷이 안 되는 곳에 있다면 좌절하지 말고 그동안 숱하게 들락거렸던 익숙한 사이트를 하나 떠올려 봅시다. 웹 브라우저의 주소창에 "www.naver.com"이라고 입력하고 엔터키나 이동 버튼을 누릅니다. 마우스 옆에 나타나는 조그마한 모래시계가 몇 번이나 뒤집히면 하얀 화면에 드디어 하나둘씩 그림과 글자가 나타나기 시작합니다. 물론 최신 PC에 100Mbps급 초고속 인터넷 이용자라면 이 작업은 1초도 채 걸리지 않을 것입니다. 도대체 이들 사이에 무슨 일이 일어나는 것일까요?

① 웹 브라우저의 주소창에 http://www.naver.com이라고 입력한 후 엔터키를 친다.
② 웹 브라우저에 입력된 주소는 DNS 서버를 통해 IP로 변환된다.
③ 변환된 IP를 통해 웹 서버에 접속하고 URL을 통해 요청 파일의 정보를 전달한다.
④ 웹 서버는 요청 파일을 서버 내부에서 찾는다.
⑤ 웹 서버는 찾은 파일을 처리하고 그 결과를 웹 브라우저에 전달한다.
⑥ 웹 브라우저는 전달받은 결과를 해석해서 사용자에게 보여준다.

간단해 보이는 네이버 접속에 이렇게 복잡한 과정이 일어납니다. 그러면 과정 하나하나를 자세히

들여다봅시다.

사용자에 의해서 주소창에 쓰인 URL은 프로토콜과 도메인 그리고 이하 상세 주소로 분리됩니다. 도메인은 DNS 서버(Domain Name Server)를 거쳐서 IP로 변환됩니다. 도메인을 IP로 변환하는 이유는 모든 인터넷 주소가 IP 체계로 이루어져 있기 때문입니다. 163.152.xxx.xxx와 같이 하나의 컴퓨터는 하나의 IP를 갖는데 IP가 숫자여서 사람의 입장에서는 외우기가 쉽지 않습니다. 그래서 도메인이란 것을 만들어 사람이 쉽게 주소를 기억할 수 있도록 IP에 이름을 붙여 주었고 DNS 서버가 이러한 이름과 그에 해당하는 IP 정보를 기록하고 있어서 DNS 서버에 도메인 정보를 전달해주면 그 도메인에 해당하는 IP를 알려줍니다.



[그림 1-2] DNS 서버

IP를 확인하면 프로토콜을 이용하여 본격적으로 통신하게 되는데 프로토콜은 컴퓨터 간의 통신규약입니다. 어떤 두 사람이 대화를 하려면 같은 언어를 이용하여 그 언어의 문법에 맞춰서 말을 해야 합니다. 그래야만 두 사람은 상대방이 어떠한 이야기를 하는지 이해할 수 있으며 그에 대한 대답이나 반응을 할 수 있습니다. 컴퓨터 간의 통신도 이와 다르지 않습니다. 컴퓨터는 서로 이야기를 나누려고 일정한 규칙을 정해 두었으며 이 규칙이 바로 프로토콜입니다.

웹 서비스는 HTTP 프로토콜을 사용하는데, HTTP란 말은 매우 익숙할 것입니다. [그림 1-1]에서 우리가 주소를 입력할 때 가장 먼저 입력하는 것이 바로 'http://'이기 때문입니다(실제 대부분의 웹 브라우저에서는 편의상 'http://'가 자동으로 입력됩니다.). HTTP는 HyperText Transfer Protocol의 약자로 HTML과 같은 Hypertext 문서를 전송하는 프로토콜입니다. 웹 서비스는 이 HTTP 프로토콜을 통해 웹 서버에 접속하는데 원격의 컴퓨터에 접근하려면 반드시 원격 컴퓨터의 주소(IP)와 원격 컴퓨터에서 열어둔 문(Port)이 필요합니다. 내 집에 누구나 아무렇게 들어와 뒤지고 다니는 것을 원하는 사람은 없듯이 컴퓨터는 외부에서 들어오는 접근을 기본적으로 막고, 감시할 수 있는 문을 몇 개 열어놓아 거기로 접근을 허용합니다. 웹 서비스는 기본적으로 80번 포트를 사용하며 위의 두 번째 과정에서 변환한 IP와 80번 포트를 통해서 웹 서버에 접근합니다. 웹 서버의 정보를 수정하면 80번이 아닌 다른 포트 번호를 통해 웹 서비스를 할 수 있는데 이런 경우 반드

시 클라이언트 측에서 이 포트 번호를 알고 있어야만 접속할 수 있습니다.

웹 브라우저가 웹 서버에 접속하면 HTTP 프로토콜에 의해 요청하는 컴퓨터의 주소와 파일 정보 등을 전달하고, 웹 서버는 이 정보를 가지고 요청한 파일을 서버에서 찾아서 그 파일을 그대로 전달하거나 그 파일이 PHP나 ASP와 같은 서버 스크립트면 스크립트를 실행하고 그 결과를 HTTP 프로토콜을 통해 웹 브라우저에 되돌려 줍니다. 여기서 웹 서버의 실행결과는 반드시 웹 브라우저가 해석할 수 있어야 하므로 웹 브라우저가 이해할 수 있는 HTML 문서 형태로 전송합니다. 만약 웹 브라우저가 요청한 파일이 서버의 지정된 주소에 없거나 사용 권한이 없는 등의 문제가 발생하면 에러 메시지를 되돌려 줍니다.

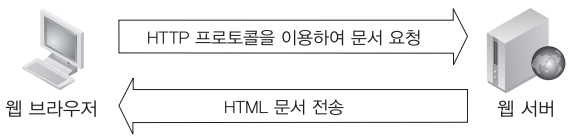
에러 메시지	설명
403 Forbidden/Access Denied	파일이나 폴더 접근 불가
404 Not Found	파일을 찾을 수 없음
500 Internal Server Error	웹 프로그램의 오류로 서비스 불가
503 Service Unavailable	과도한 접속으로 인한 일시적 서비스 불가

[표 1-1] 대표적인 웹 서버 에러 메시지

이처럼 여러 가지 절차와 각 절차에서 정의된 규칙을 잘 지켜나가야만 웹 브라우저가 올바른 웹 페이지 결과를 얻을 수 있습니다. 물론 앞서 나열한 처리 과정은 크게 바라본 과정이며 또한 모든 과정을 설명한 것은 아닙니다. 그 예로 첫 번째와 두 번째 과정의 사이에 경우에 따라 프록시 서버(Proxy Server)가 존재할 수도 있습니다. 프록시 서버는 일단 한번 접속한 경우 서버에 파일을 저장해두고 있다가 웹 브라우저의 다음번 요청이 들어오면 해당 사이트에 다시 접속하지 않고 프록시 서버가 저장해둔 파일을 전송해 줍니다. 프록시 서버는 이러한 방식으로 해당 사이트가 느리거나 일시적으로 접속량이 많아져 반응속도가 느릴 때보다 빠른 웹 서비스를 제공합니다. 그러나 항상 프록시 서버를 거쳐야 하기 때문에 점점 네트워크 속도가 빨라지고 있는 현 시점에서는 점차 그 용도가 줄어들고 있습니다.

Section 02 정적인 웹 페이지 vs 동적인 웹 페이지

웹 서버의 결과는 언제나 웹 브라우저가 해석할 수 있는 형식으로 전송됩니다. HTML이 가장 대표적인 형식이며 SGML이나 DHTML, XML 등의 문서 형식으로도 그 결과를 얻을 수 있습니다.



[그림 1-3] 정적인 웹 서비스

초창기의 웹 서비스는 [그림 1-3]과 같이 단순히 웹 페이지를 요청하면 그에 해당하는 문서를 찾아서 전송해주는데 그쳤습니다. 이와 같은 일방적인 서비스는 점차 인터넷의 보급과 웹 서비스의 활성화 탓에 부족함을 느끼게 하였습니다. 인터넷 사용자들은 단순한 정보의 수혜자에서 벗어나서 정보의 제공자와 서로 교류하기를 원하고 또한 자신의 생각이나 지식을 공유하고자 하는 욕망이 커졌기 때문입니다. 즉, 정보의 수혜자가 다시 정보의 제공자가 되는 시대가 온 것입니다.

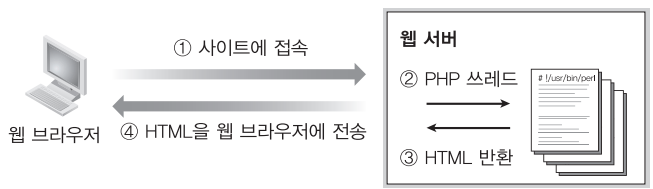
초기의 홈페이지들은 순수한 HTML만으로 구성된 경우가 많았기 때문에 정보의 수혜자가 공급자에게 어떠한 행동도 취할 수 없었습니다. 이러한 문제를 해결하고자 역으로 클라이언트에서 웹 서버로 정보를 보내고 이 정보를 바탕으로 사용자의 입력에 대해 처리할 수 있는 방법을 찾게 되었고 이에 등장한 것이 CGI(Common Gateway Interface)입니다.

[그림 1-4] CGI (Common Gateway Interface) 방식

HTML만으로 이루어진 웹 페이지는 [그림 1-4]에서 ①번과 ④번의 과정으로 처리됩니다. 그러나 CGI의 경우는 ②번과 ③번의 과정이 추가되어 웹 서버 측에서 계산이나 데이터 처리 등이 가능해졌습니다. CGI 프로그램은 웹 서버에서 실행되는 서버 측의 프로그램입니다. 웹 브라우저를 통해 CGI의 호출을 받으면 웹 서버는 해당 CGI 프로그램 파일을 찾아서 실행합니다. 실행된 CGI 프로그램은 작성된 목적에 따라 데이터를 처리하고 그 결과를 HTML로 웹 브라우저에 전달해 줍니다. 즉, CGI 프로그램은 웹 서버가 실행하고 실행결과를 HTML 형식으로 출력해주는 프로그램으로 생각할 수 있습니다. PHP와 같은 서버 측 스크립트 언어가 널리 사용되기 전에는 대부분의 웹 게시판들이 CGI 형식으로 작성되었습니다. 또한 초기의 CGI 프로그램은 주로 C 언어로 작성되었으나 C언어는 웹을 위해서 만들어진 언어가 아니라 범용적인 언어이기 때문에 프로그램을 작성하기가 쉽지 않았습니다. 이후 Perl과 같은 스크립트 언어의 등장으로 보다 쉽게 CGI 프로그램을 작성할 수 있게 되었으나 C로 작성한 CGI에 비해서 처리 속도가 느리다는 단점 탓에 1990년대 말까지도 대규모 프로젝트는 C로 작성하는 경우가 많았습니다. 반면 Perl은 개발이 쉽다는 장점 덕분에 많은 개발자가 Perl로 CGI 프로그램을 개발했고 CGI를 널리 보급시키는 역할을 했습니다.

그러나 CGI 방식은 웹 서버의 요청마다 매번 새로운 프로세스를 생성하고 그 결과를 웹 서버에 전

송한 후 프로세스를 종료시키는 방식 탓에 한계를 가지고 있었습니다. 100개의 요청이 들어오면 CGI 프로세스 100개가 생성되고, 하나의 프로세스가 1MB의 메모리를 사용한다고 가정할 때 100개의 프로세스가 총 100MB의 메모리를 점유해버리는 문제가 생깁니다. 이러한 CGI 방식에 대한 문제점을 극복하고자 웹 서버에 탑재되어 쓰레드(Thread)로 동작하는 PHP나 ASP와 같은 서버 측 스크립트가 널리 사용되었습니다. PHP는 웹 서버의 모듈로 장착되어 있어서 웹 서버의 요청이 있으면 CGI처럼 매번 프로세스를 생성하는 것이 아니라 쓰레드를 생성합니다. 쓰레드는 프로세스와 달리 쓰레드 간의 시스템 자원 공유가 가능하기 때문에 100개의 쓰레드는 메모리를 공유하여 100MB보다 훨씬 적은 양의 메모리를 소비합니다. PHP는 개념적으로 CGI와 유사한 처리 방식을 갖지만 PHP의 특성을 조금 더 반영하면 다음과 같습니다.



[그림 1-5] PHP 처리방식

여기서 잠깐

☑ 모듈 vs CGI 방식

PHP는 웹 서버의 모듈로 동작한다고 언급했습니다. 웹 서버의 모듈로 동작한다는 것은 마치 웹 서버의 부품처럼 사용된다는 뜻입니다. PHP가 웹 서버의 부품처럼 장착되었다가 분리되는 것이 가능한 방식이 모듈 방식입니다. 그러나 PHP가 반드시 모듈 형태로만 동작하는 것은 아닙니다. 설치 방법에 따라 FastCGI 형식으로도 PHP를 동작시킬 수 있습니다. FastCGI는 앞서 말씀드린 CGI의 실행 속도를 향상시키기 위해 등장한 방법으로 CGI보다 3~30배 빠른 속도를 나타낸다고 합니다. 그러나 FastCGI 방식보다는 웹 서버의 모듈 형식을 권장합니다.

Section 03 PHP 개발 환경

이제 PHP의 동작방식도 알게 되었으니 본격적으로 PHP를 다루도록 합시다. 우선 PHP 프로그래밍을 하려면 반드시 웹 서버가 필요합니다. PHP는 웹 서버로부터의 요청에 의해서 처리되는 부속

품과 같은 존재이기 때문입니다. 그래서 PHP는 웹 서버 없이는 혼자서 웹 서비스를 제공할 수 없습니다. 웹 서버와 함께 데이터베이스라는 녀석도 필요합니다. 데이터베이스는 반드시 필요한 항목은 아니지만 웹 서비스에서 데이터베이스를 빼놓고 이야기하기는 쉽지 않습니다. 데이터베이스는 제7장에서 자세히 다룰 예정이니 이 정도만 이해하고 넘어가도록 합니다. 다음과 같이 정리합시다.

구성요소는 모두 세 가지! "웹 서버, 데이터베이스 그리고 PHP"

어떤 웹 서버를 사용할 것인가?

웹 서버는 대표적으로 아파치 웹 서버(Apache Web Server)와 마이크로소프트의 인터넷 정보 서버(IIS, Internet Information Server)가 가장 유명하며 많이 사용됩니다. 실제로 이 두 웹 서버 외에도 수백 개의 웹 서버가 존재하지만 현존하는 대부분의 웹 서버는 이들로 구성되어 있다고 해도 과언이 아닙니다. Netcraft Survey의 2007년 12월 조사결과에 따르면 아파치 웹 서버가 49.35%, IIS 웹 서버가 35.76%를 차지하여 둘의 합이 85% 이상을 차지하고 있습니다. 이 정도의 점유율은 5천만에서 7천만 개의 웹 사이트가 해당 웹 서버를 사용하고 있다는 뜻이므로 웹 서버의 안정성이나 기능 등은 나무랄 것이 없으리라 생각합니다. 그렇다면 아파치 웹 서버와 IIS 웹 서버는 PHP와 함께 동작할 수 있는지가 중요하겠군요. 그런데 너무나 고맙게도 PHP는 이 두 종류의 웹 서버에서 모두 사용할 수 있습니다. 아파치 웹 서버에는 DSO 방식으로 PHP를 실행할 수 있고 IIS 서버에는 ISAPI 방식으로 PHP를 실행할 수 있습니다. DSO(Dynamic Shared Object)니 ISAPI니 하는 것은 사실 PHP를 모듈로 어떻게 동작시키는지에 대한 방법일 뿐입니다. 그러니 자세히 다루지 않고 넘어가도록 하겠습니다.

그러면 아파치 웹 서버와 IIS 웹 서버 중의 하나를 선택하는 일이 남았습니다. 아파치 웹 서버는 대부분의 OS와 플랫폼을 지원하고 있고 IIS 웹 서버에는 윈도우 NT 기반의 OS에서만 동작합니다. 그래서 만약 웹 서버를 설치할 컴퓨터가 리눅스 계열이라면 무조건 아파치를 선택해야 하고 윈도우 NT 계열이라면 IIS와 아파치 웹 서버 모두를 사용할 수 있습니다. 아파치 웹 서버는 플랫폼에 독립적인 장점뿐만이 아니라 PHP와 환상의 궁합을 이루는 것으로도 유명합니다. 예전에는 아파치 웹 서버가 윈도우에서 제대로 동작하지 않아서 윈도우 환경은 IIS와 PHP를 함께 사용하였으나 최근에는 윈도우에서도 매우 안정적으로 동작하여 IIS와 PHP를 같이 사용하는 경우가 오히려 드물어졌습니다.

어떤 데이터베이스를 사용할 것인가?

데이터베이스의 종류는 정말 다양합니다. 그뿐만 아니라 PHP가 지원하는 데이터베이스의 수도 매

우 다양하며 ODBC(Open Database Connectivity)를 이용하면 현존하는 데이터베이스 대부분을 사용할 수 있습니다. 그렇다면 어떤 데이터베이스를 사용하는 것이 좋을까요? 데이터베이스는 대표적으로 오라클 사의 오라클(Oracle), IBM의 DB2, 마이크로소프트의 MS-SQL, 썬 마이크로시스템 사가 인수한 오픈 소스 데이터베이스인 MySQL, 볼랜드에서 개발한 InterBase 등등이 있습니다. 일반적으로 관리할 데이터의 규모나 개발자의 취향에 따라서 데이터베이스를 선택하는데, 대규모 사이트 및 공공기관 등에서는 안정성을 이유로 오라클을 많이 사용하고 있습니다. 다음으로 IIS, ASP와 함께 MS-SQL로 연결되는 MS 트리오가 유지 관리의 편리성 때문에 중대형 사이트에서 널리 사용되고 있습니다. 그리고 엄청난 속도를 자랑하는 가벼운 데이터베이스로 MySQL이 중소형 데이터베이스 시장에서 두각을 나타내고 있습니다. 마지막으로 InterBase는 델파이, C++ 빌더와 같은 볼랜드 계열 프로그래밍 툴과 함께 윈도우용 프로그램에 많이 사용되고 있습니다. InterBase의 장점은 볼랜드 사의 상용 데이터베이스였다는 것입니다. 현재는 파이어버드 프로젝트를 통해서 소스가 공개되어 라이센스 비용 없이 사용할 수 있지만 몇 년 전만 하더라도 상용 데이터베이스였기 때문에 안정성과 기능에서 두루두루 잘 갖춰져 있는 우수한 데이터베이스입니다.

이러한 데이터베이스 중에서 어떤 것을 선택할 것인지는 앞서도 말씀드렸지만 사용 용도와 개발자의 숙련도에 따라서 결정하는 것이 좋습니다. 성능과 안정성에서는 오라클과 같은 값비싼 상용 데이터베이스가 좋지만 소규모 사이트에서 오라클을 쓰는 것은 낭비일 뿐입니다. 이것은 마치 소 잡는 칼로 닭 잡는 격입니다. 이에 반해 MySQL은 무료 데이터베이스이면서도 가볍고 빠르며 구글, 야후, 유튜브와 같은 대규모 사이트에서도 사용할 만큼 훌륭한 데이터베이스로 거듭나고 있습니다. 특히 우리는 PHP를 공부하는 처지에서 값비싼 데이터베이스를 구매하여 사용할 수 없으므로 MySQL은 우리에게 최적의 데이터베이스라고 할 수 있습니다. 따라서 이 책에서는 MySQL 데이터베이스를 사용하도록 합니다.

APM

우리는 이제 웹 서버와 데이터베이스를 결정했습니다. 아파치 웹 서버, PHP 그리고 MySQL을 이용한 시스템을 속칭 "APM"이라고 합니다. 각 단어의 첫 글자를 딴말로 누가 이 말을 처음 사용하였는지는 알 수 없으나 어느덧 많은 사람이 이 삼총사를 일컬어 APM이라고 부르기 시작하였습니다. 간혹 마피아(MySQL + PHP + Apache)라고 부르는 사람들도 있으며 APM이 주로 Linux 기반에서 사용되기 때문에 LAMP(Linux + Apache + MySQL + PHP)라고 부르는 사람도 있습니다. 이처럼 여러 가지 별명으로 불릴 만큼 이들의 궁합은 정평이 나있으며 PHP를 공부하는 사람들에게는 정말 탁월한 개발 환경이 아닐 수 없습니다.

Section 04

리눅스 기반 APM 설치

돈 한 푼 안들이고 웹 서버를 구축하는 가장 탁월한 방법은 리눅스 기반에 APM을 설치하는 것입니다. 운영체제부터 데이터베이스까지 어느 것 하나 유료인 것이 없습니다. 그뿐만이 아니라 이들 각각은 앞서도 말씀드렸지만 성능이 매우 우수하며 궁합도 잘 맞기로 소문이 나 있습니다. 최근 리눅스 버전에는 기본적으로 APM이 자동으로 설치된 경우가 많습니다. 그 정도로 많이들 사용하고 있다는 증거입니다. 그러나 리눅스 배포판의 APM은 배포 시점이나 안정성을 이유로 오래된 버전인 경우가 많으므로 만약 리눅스를 다룰 줄 알고 리눅스의 관리자 권한을 보유하고 있다면 [표 1-2]와 같은 방법으로 최신 버전의 APM을 설치해보도록 합시다. 만약 리눅스에 대해 잘 모른다면 윈도우 기반 APM 설치로 넘어가기 바랍니다.

방법	설명
RPM 패키지	레드햇 계열의 리눅스에서 사용
apt-get / yum	GNU Linux / Fedora Core / CentOS에서 사용
소스 코드	모든 리눅스 계열에서 사용

[표 1-2] 리눅스 환경에서 APM을 설치하는 방법

RPM 패키지는 레드햇 계열의 리눅스에서 사용하는 설치 방법입니다. RPM은 우리가 윈도우에서 프로그램을 설치할 때 사용하는 설치 프로그램과 유사합니다. 최근에는 RPM 패키지 방식에서 한 발 더 나아가 프로그램을 자동으로 다운로드하여 설치해주는 GNU Linux 계열의 apt-get 프로그램과 최근 레드햇 계열에서 사용하는 yum이 등장하여 패키지 관리를 더욱 쉽게 해주고 있습니다. 이 방법들은 리눅스에서 소스를 이용해 프로그램을 설치하는 방법이 서툰 사람들에게 매우 유용합니다. 또한 필자와 같이 매우 게으른 사람에게도 유용한 방법입니다. 마지막으로 소스를 이용한 설치 방법이 있습니다. 이 방법은 각 시스템에 맞게 컴파일하여 사용할 수 있고 모든 리눅스 계열에서 사용할 수 있다는 장점이 있습니다.

YUM을 이용한 자동 설치

리눅스에서 APM을 설치하는 간단한 방법 중에서 최근 Fedora Core나 CentOS 등에서 사용하는 yum을 이용한 자동 설치 방법에 대해 알아보겠습니다. 일단 리눅스에 설치하는 경우는 리눅스에 대해 약간의 지식을 가지고 있다는 전제하에 시작하겠습니다.

yum 프로그램은 자동으로 RPM 패키지를 다운받아서 설치해주는 프로그램입니다. 따라서 설치를 위해서 APM 각각의 설치 파일을 다운받을 필요가 없습니다. 설치를 위해서는 인터넷에 연결된 yum을 지원하는 리눅스만 있으면 됩니다. 설치를 하기에 앞서 반드시 관리자(root) 아이디로 로그인합니다.

① 아파치 웹 서버 설치

```
[root@ezphp root]# yum install httpd
```

② PHP 설치

```
[root@ezphp root]# yum install php
```

③ MySQL 설치

```
[root@ezphp root]# yum install mysql-server
```

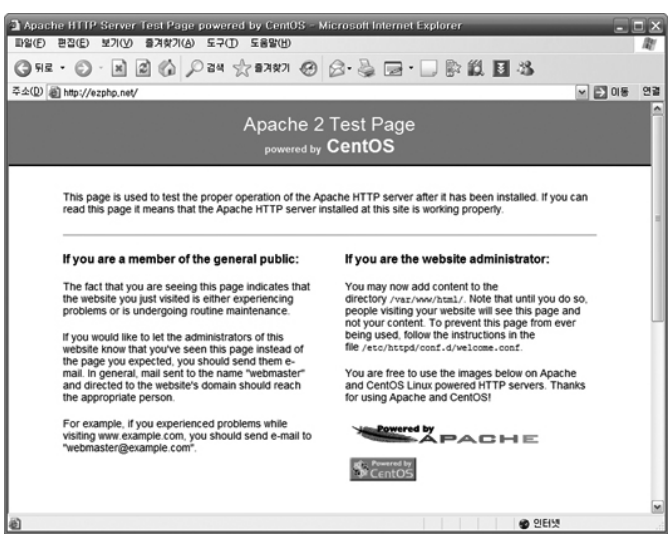
④ PHP와 MySQL 연동

```
[root@ezphp root]# yum install php-mysql
```

위와 같이 4단계의 설치 명령을 통해서 정말 간단하게 APM을 구성할 수 있습니다. 위와 같은 절차에 의해서 설치를 하고 아파치 웹 서버를 실행해 보도록 하겠습니다.

```
[root@ezphp root]# apachectl start
```

CentOS 5.1에서 yum을 이용해서 위와 같이 설치하고 홈페이지에 접속해보면 설치가 올발라졌을 경우 다음과 같은 화면을 볼 수 있습니다.

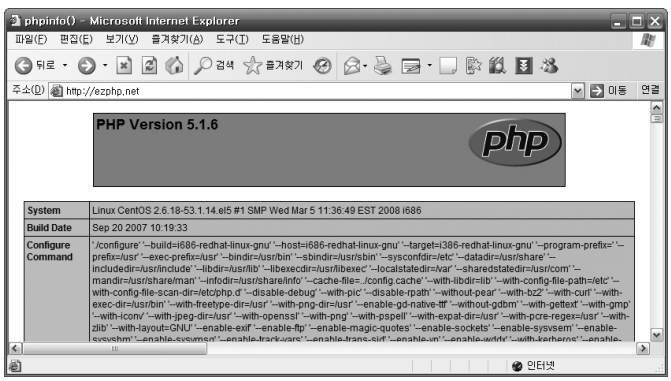


[그림 1-6] CentOS 5.1에서 APM 설치 결과

PHP의 동작을 확인하기 위해서 /var/www/html 디렉토리에 index.php 파일을 만들고 다음의 내용을 기록합니다.

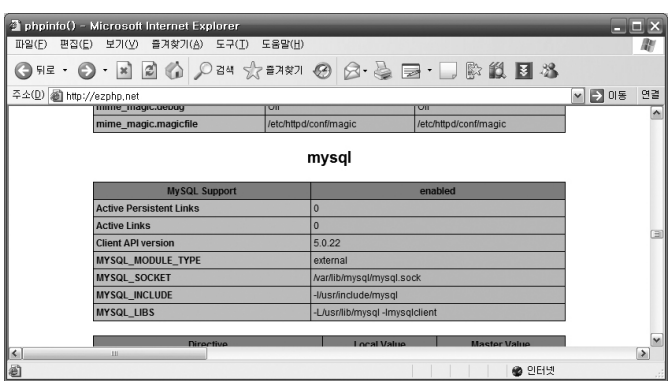
```
<? phpinfo(); ?>
```

위의 내용을 index.php라는 이름으로 저장하고 다시 홈페이지에 접속해보면 [그림 1-6]에서 [그림 1-7]로 웹 페이지가 변경된 것을 확인할 수 있습니다.



[그림 1-7] phpinfo()의 출력결과

이 웹 페이지는 PHP의 설치 정보를 알려주는, PHP의 내장 함수인 phpinfo()의 결과를 보여주고 있습니다. 자세한 것은 다음에 배우기로 하고 일단 [그림 1-7]과 같은 결과를 볼 수 있다면 PHP는 제대로 동작하고 있다는 것만 알아두기 바랍니다. 이 웹 페이지의 중간 정도 부분에서 mysql 항목이 있는지 확인을 합니다.



[그림 1-8] PHP와 MySQL 연동 확인

[그림 1-8]에서 mysql 항목을 확인하는 이유는 설치 ④ 과정에서 php-mysql의 설치가 제대로 설치되었는지를 확인하는 것입니다. 기본적으로 PHP를 설치하면 mysql 관련 함수를 사용할 수 없게 되어 있습니다. 그래서 설치 ④ 과정으로 PHP에서 MySQL을 사용할 수 있도록 php-mysql을 설

치해주는 것입니다.

마지막으로 MySQL 데이터베이스가 제대로 설치되었는지 확인해 보도록 합시다. MySQL이 제대로 동작하는지 확인하기 위해 MySQL 서버를 다음과 같이 실행합니다.

```
[root@ezphp root]# /etc/init.d/mysqld start
MySQL (을)를 시작 중:                                      [  OK  ]
```

위와 같이 MySQL 서버를 실행하면 MySQL 클라이언트 프로그램을 통해서 MySQL 데이터베이스 내부로 들어가 보도록 합니다.

```
[root@ezphp root]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 5.0.22

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

yum을 이용해서 설치하는 경우 MySQL 데이터베이스의 관리자 계정인 root에 대한 암호가 설정되지 않아서 위와 같이 곧바로 접속할 수 있습니다. 만약 위와 같은 화면을 볼 수 있다면 올바르게 설치된 것입니다. 여기서 관리자 계정의 비밀번호는 반드시 변경해주어야 합니다. 그렇지 않으면 권한이 없는 누구나 관리자 계정으로 데이터베이스에 접근할 수 있기 때문입니다. 자세한 내용은 소스를 이용한 APM 설치에서 다루도록 하겠습니다.

이제 우리의 리눅스 컴퓨터는 위와 같은 간단한 설치 절차를 통해서 웹 서버의 기능과 데이터베이스 서버의 기능을 갖추었습니다. 서버를 다시 시작할 때마다 아파치 웹 서버와 MySQL 서버를 실행해야만 이 모두를 사용할 수 있습니다. 매번 다시 시작할 때마다 새로 서버들을 실행해주는 불편함을 없애려면 다음과 같이 아파치 웹 서버와 MySQL 데이터베이스 서버가 자동으로 실행될 수 있도록 설정을 합니다.

```
[root@ezphp root]# setup
```

[그림 1-9] setup을 실행한 결과

setup 프로그램은 [그림 1-9]에서 보듯이 네트워크 설정과 방화벽 설정 및 시스템 서비스 등을 설정하는 프로그램입니다. 아파치 웹 서버와 MySQL 데이터베이스 서버를 서비스로 등록하면 시스템이 재부팅되었을 때 자동으로 서비스가 실행됩니다. 따라서 방향키를 이용하여 시스템 서비스를 선택합니다.

[그림 1-10] 시스템 서비스 설정

시스템 서비스를 선택하면 [그림 1-10]처럼 현재 리눅스 시스템에 설치된 다양한 서비스 항목들이 나타납니다. 그중에서 httpd와 mysqld 항목을 찾아 스페이스바를 이용하여 선택합니다. 별표(*)가 되어 있는 서비스들이 자동으로 시작할 서비스 항목입니다. httpd와 mysqld 항목을 모두 선택하였다면 탭키를 이용하여 확인 버튼으로 이동합니다. 확인 버튼 위에서 엔터키나 스페이스바를 누르면 설정이 반영되며 다시 [그림 1-9]로 되돌아옵니다. 그리고 다시 탭키를 이용하여 종료 버튼을 선택한 다음 엔터키나 스페이스바를 누르면 콘솔창으로 되돌아옵니다. 제대로 서비스가 실행되는지 확인하고 싶다면 reboot 명령을 통해서 재시작을 해보기 바랍니다.

이제 yum을 이용한 APM 설치를 모두 마쳤습니다. yum뿐만이 아니라 apt-get 또한 이 방법과 거의 같습니다. yum이나 apt-get과 같은 프로그램을 지원하는 리눅스를 사용한다면 굳이 다음으로 소개하는 소스를 이용한 설치를 할 필요는 없습니다. yum이나 apt-get과 같은 프로그램을 다운받아서 설치하는 것은 비록 가장 최근 버전은 아닐지라도 많은 사람이 테스트하고 안정적으로 동작하는 것을 확인한 버전이므로 안심하고 사용할 수 있습니다. 만약 설치가 매끄럽게 되지 않는다면 다음과 같은 방법으로 이전에 설치된 패키지들을 깨끗이 지운 후에 다시 한번 시도해 보기를 권해 드립니다.

```
[root@ezphp root]# yum remove httpd
[root@ezphp root]# yum remove mysql
```

yum은 프로그램을 설치하거나 제거할 때 프로그램의 의존성을 검사하여 의존적인 프로그램을 함께 설치하거나 제거합니다. 그래서 httpd를 제거하면 아파치 웹 서버에 의존적인 php가 자동으로 제거됩니다. 눈치가 빠른 분, 혹은 굉장한 기억력을 가진 분이라면 mysql을 설치할 때와 제거

할 때 미묘하지만 다른 이름을 사용하고 있음을 알아차렸을지도 모르겠습니다. 설치를 할 때에는 mysql-server를 설치하였으나 제거할 때는 mysql-server가 아닌 mysql을 사용하고 있습니다. 사실 mysql-server 패키지는 MySQL 데이터베이스 서버 프로그램이고 mysql은 MySQL 데이터베이스의 클라이언트 프로그램입니다. 둘은 의존성을 갖고 있는데 mysql-server를 설치하면 데이터베이스에 접속할 클라이언트 프로그램이 필요하기 때문에 자동으로 mysql 패키지가 설치됩니다. 반대로 mysql 클라이언트 프로그램을 지울 때는 mysql-server가 같이 제거됩니다. 단, mysql 패키지를 설치한다고 해서 mysql-server가 설치되지 않으며 마찬가지로 mysql-server를 제거한다고 해서 mysql이 제거되지는 않습니다.

소스를 이용한 APM 설치

yum이나 apt-get 방식은 특정 리눅스 계열에서만 올바르게 동작합니다. 레드햇 계열이나 데비안과 같은, GNU 리눅스 계열이 아닌 다른 배포판은 이를 지원하지 않는 경우가 많아서 APM을 설치하기 위해서 가장 원초적인 방법인 소스를 이용한 설치 방법을 알아둘 필요가 있습니다. 자동 설치 프로그램보다 소스 파일을 찾아서 다운받아야 하고 설치 과정도 다소 복잡하지만 PHP 프로그래머가 되고자 하는 독자라면 이 방법을 꼭 한 번은 시도해 보길 권합니다. 그렇다고 설치 과정을 모두 외우는 것은 바보 같은 짓입니다. 단지 어떤 식으로 설치하는지 전체적인 그림만 머릿속에 담아 두면 됩니다. 다음에 설치할 일이 생기면 이 페이지를 다시 뒤적거리면 됩니다.

우선, 다음 사이트를 통해서 최신 버전의 소스 파일을 다운로드합니다.

① 아파치 웹 서버 : http://httpd.apache.org/

② PHP : http://www.php.net

③ MySQL : http://www.mysql.com

APM의 각 사이트에서 최신 버전을 다운로드해서 설치할 때는 반드시 운영체제를 확인하여 리눅스용 소스 파일을 다운받도록 합니다. 각 소스 파일은 대부분 'httpd-version.tar.gz'와 같은 이름을 갖습니다. 새로운 버전이 출시될 때 간혹 설치 방법이 약간씩 변경될 가능성이 있으므로 이 방법으로 설치가 안 되는 경우가 있다면 PHP 스쿨(http://www.phpschool.com)과 같은 PHP 개발자 커뮤니티 사이트에서 변경된 설치 방법을 살펴보기 바랍니다.

[그림 1-11] APM 소스 파일(www.hanb.co.kr/exam/1658)

이제 본격적으로 설치를 시작해 보겠습니다.

먼저 다운받은 소스 파일을 APM을 설치할 리눅스의 /usr/local/src 디렉토리에 저장합니다. 설치를 위해서는 앞에서도 언급하였듯이 반드시 관리자(root) 아이디로 로그인해야 합니다.

```
[root@ezphp root]# cd /usr/local/src
[root@ezphp src]# ls
mysql-5.0.51a-linux-i686.tar.tar
httpd-2.2.8.tar.gz
php-5.2.5.tar.tar
```

해당 디렉토리에 소스 파일을 복사하였으면 제일 먼저 MySQL을 설치하도록 합니다.

MySQL 설치

소스 파일을 통한 설치에서는 반드시 설치 순서를 지킬 필요가 있습니다. 제일 먼저 MySQL을 설치해야 하는데, MySQL은 소스 파일을 사용하지 않고 바이너리 버전을 사용하도록 합니다. 바이너리 버전은 리눅스의 종류나 시스템의 환경에 따라 여러 가지 환경에서 미리 컴파일해둔 것으로 자신의 리눅스 버전에 맞는 바이너리를 사용하면 복잡하고 오래 걸리는 설치 과정을 간단하게 줄일 수 있습니다.

먼저 MySQL을 관리 운영할 mysql 사용자 계정을 등록합니다. MySQL 데이터베이스의 사용자 계정을 만드는 것은 필수적인 것은 아니지만 MySQL 설치 문서의 권장사항이므로 다음과 같이 등록하도록 합니다.

```
[root@ezphp src]# adduser mysql
```

MySQL 바이너리 소스 파일의 압축을 해제합니다.

```
[root@ezphp src]# tar xvzf mysql-5.0.51a-linux-i686.tar.tar
```

```
[root@ezphp src]# mv mysql-5.0.51a-linux-i686 /usr/local
```

압축이 해제되면 mysql-5.0.51a-linux-i686과 같은 디렉토리가 생기게 됩니다. 이 디렉토리를 /usr/local/ 디렉토리로 옮깁니다.

```
[root@ezphp src]# cd /usr/local/
[root@ezphp local]# ln -s mysql-5.0.51a-linux-i686 mysql
```

/usr/local/ 디렉토리로 이동하여 긴 디렉토리 이름을 mysql이라는 심볼릭 링크로 만들어 줍니다. 심볼릭 링크는 여기서 일종의 별명이라고 생각하면 됩니다.

```
[root@ezphp local]# cd mysql
[root@ezphp mysql]# chown -R mysql
[root@ezphp mysql]# chgrp -R mysql
```

mysql 디렉토리로 이동합니다. mysql 디렉토리는 심볼릭 링크로 mysql-5.0.51a-linux-i686 디렉토리를 가리키고 있습니다. 따라서 mysql 디렉토리로 이동하면 실제로 mysql-5.0.51a-linux-i686 디렉토리로 이동하게 됩니다. mysql 디렉토리로 이동하고 현재 디렉토리에 대한 권한을 변경합니다.

현재 작업을 루트 계정으로 하고 있기 때문에 모든 파일과 디렉토리에 대한 권한이 루트의 소유로 되어 있습니다. 즉, 루트만이 이 파일들에 접근할 수 있다는 뜻이 됩니다. 그런데 MySQL은 mysql 계정을 통해 설치와 관리가 되므로 mysql 계정은 반드시 이 디렉토리에 대한 사용 권한이 있어야 합니다. 따라서 /usr/local/mysql 디렉토리를 mysql 계정이 마음껏 사용할 수 있게끔 소유권을 넘겨주는 작업을 합니다. 여기서 옵션으로 사용되고 있는 -R은 재귀적으로 해당 디렉토리 내의 모든 파일에 대해 권한을 변경한다는 것을 의미합니다.

```
[root@ezphp mysql]# scripts/mysql_install_db --user=mysql
```

mysql_install_db를 통해 데이터베이스를 설치합니다. 옵션은 데이터베이스 관리자가 mysql 계정이라는 것을 의미합니다.

```
[root@ezphp mysql]# chown -R root
[root@ezphp mysql]# chown -R mysql data
```

이제 mysql 설치를 끝냈으면 설치에 필요했던 mysql 계정의 권한을 다시 루트에 넘겨줍니다. 설치할 때에는 mysql에 mysql 디렉토리에 대한 모든 권한을 부여했지만 설치가 완료되고 나면 더는 모든 권한을 남겨둘 필요가 없기 때문입니다. 그러나 추후 여러 가지 정보가 저장될 data 디렉토리는 계속해서 mysql 계정이 사용되므로 이 디렉토리만은 모든 권한을 부여하도록 설정합니다.

이제 모든 설치가 완료되었으니 MySQL 데이터베이스 서비스를 실행해 보도록 합시다.

```
[root@ezphp mysql]# bin/mysqld_safe --user=mysql &
```

여기서 '&'의 의미는 명령을 실행하되 백그라운드에서 동작하도록 하는 것입니다. 만약 여기에 '&'를 붙이지 않는다면 터미널은 커서만 깜빡거리며 멈춰 있게 됩니다. 왜냐하면 MySQL이 종료될 때까지 기다리고 있을 것이기 때문입니다. '&' 기호를 통해서 mysql은 동작하도록 놔두고 다음 작업을 하고자 mysql을 백그라운드에서 동작하게 합니다.

이제 우리의 리눅스 서버는 MySQL 데이터베이스 서버 기능을 갖추었습니다. 서버 구동 기념으로 일단 서둘러서 MySQL 관리자의 비밀번호부터 바꾸기로 합시다.

```
[root@ezphp mysql]# bin/mysqladmin -u root password '새로운 비밀번호'
```

이로써 안전하게 관리자의 비밀번호도 변경했으니 마음 놓고 MySQL을 사용할 수 있습니다. 여기서 root는 리눅스 시스템 관리자의 계정인 root와는 다른, MySQL 데이터베이스의 관리자인 root를 의미합니다. 따라서 여기서 root의 비밀번호를 변경한다고 해서 리눅스 계정의 root 비밀번호가 변경되는 것은 아닙니다.

제대로 설치되었는지를 확인하기 위하여 MySQL에 접속해 보도록 합시다. yum을 이용한 설치 방법과는 다르게 데이터베이스 관리자 계정의 비밀번호를 변경했기 때문에 다음과 같이 접속합니다.

```
[root@ezphp mysql]# bin/mysql -u root -p
Enter password:
```

mysql 프로그램은 앞서 yum을 이용한 APM 설치에서도 잠깐 언급했듯이 MySQL 데이터베이스 서버에 접속하는 클라이언트 프로그램입니다. -u는 데이터베이스에 접근하는 아이디를 지정하겠다는 것이고 -p는 비밀번호가 있으니 별도로 입력하겠다는 뜻입니다. 앞에서는 root의 비밀번호가 없었기 때문에 이러한 옵션을 입력하지 않아도 되었지만 root의 비밀번호를 변경했으므로 위와 같이 옵션을 사용해야만 접속할 수 있습니다. 위의 명령은 root 아이디로 비밀번호를 입력하여 접속하겠다는 의미입니다. 올바른 비밀번호를 입력하면 [그림 1-12]와 같은 화면을 볼 수 있습니다.

[그림 1-12] mysql 클라이언트를 통해서 MySQL 서버 접속

지금까지는 /usr/local/mysql/ 디렉토리에서 /usr/local/mysql/bin 디렉토리에 있는 MySQL 실행 파일들을 실행하기 위해 불편함을 감수했으나 다음에 보다 편리하게 사용하도록 필요한 몇 개의 실행 파일을 /usr/bin 디렉토리로 옮겨놓도록 합니다.

```
[root@ezphp mysql]# cd bin
[root@ezphp mysql]# cp mysql /usr/bin
[root@ezphp mysql]# cp mysqladmin /usr/bin
[root@ezphp mysql]# cp mysqldump /usr/bin
[root@ezphp mysql]# cp safe_mysqld /sbin
```

mysql 파일은 말씀드렸듯이 MySQL 데이터베이스 서버에 접속할 수 있는 클라이언트 프로그램이고 mysqladmin은 관리자가 사용하는 MySQL 관리용 프로그램입니다. 그리고 mysqldump는 MySQL 데이터베이스에 있는 데이터를 백업하고자 할 때 사용하는 프로그램이며 마지막으로 safe_mysqld는 MySQL 데이터베이스 서버 프로그램입니다. 이 파일들을 PATH가 설정된 /usr/bin이나 /sbin에 복사를 해두면 어느 디렉토리에서 작업을 하건 다음과 같이 실행할 수 있습니다.

```
[root@ezphp mysql]# mysql -u root -p
Enter password:
```

앞서 자동 설치의 경우에는 설치 후 자동으로 위의 파일들을 각 디렉토리에 이동시켜 주기 때문에 이와 같은 번거로운 작업을 할 필요가 없습니다.

아파치 설치하기

예전에는 아파치를 미리 설정하고 PHP를 설치한 후 다시 아파치를 설치하는 과정을 거쳐야만 했습니다. 그런데 DSO 방식으로 PHP 모듈을 사용할 수 있게 되면서 아파치를 미리 설정해주는 과정이 사라졌습니다. 여기서 DSO 방식이란 Dynamic Shared Object의 약자로 모듈을 동적으로 추가 및 제거할 수 있게 해주는 방식을 말합니다. 즉, USB 드라이브를 꽂았다가 뺐다가 하듯이 PHP도 아파치 웹 서버에 꽂았다가 뺐다가 하는 것을 가능하게 하는 것을 의미합니다. 이에 따라 PHP 버전을 업그레이드하려면 예전에는 반드시 아파치를 재설치하는 과정이 필요했으나 DSO 방식 덕분에 인해 더는 그러한 절차가 필요하지 않게 되었습니다.

우선 아파치 웹 서버를 설치하기 위해서 다운받은 파일의 압축을 풉니다.

```
[root@ezphp mysql]# cd /usr/local/src
[root@ezphp src]# tar xvzf httpd-2.2.8.tar.gz
```

압축을 해제한 디렉토리로 이동하여 아파치 웹 서버의 설치 환경을 설정합니다.

```
[root@ezphp src]# cd httpd-2.2.8
[root@ezphp httpd-2.2.8]# ./configure --prefix=/usr/local/apache2 \
```

```
> --enable-module=so
```

configure는 환경 설정 및 컴파일 옵션을 지정하는 프로그램입니다. 다양한 설치 옵션을 제공하는데 여기서는 설치될 위치를 나타내는 prefix와 PHP를 DSO 방식으로 설치하기 위한 --enable-module=so 옵션만을 사용하도록 합니다. 여기서 역슬래시(\)는 다음 줄에 이어서 옵션을 설정하겠다는 의미입니다. 그래서 역슬래시를 사용하면 위와 같이 '>' 기호가 나타납니다. 그러므로 일부러 '>' 기호를 입력하지 마세요.

```
[root@ezphp httpd]# make
[root@ezphp httpd]# make install
```

컴파일에 대한 설정이 모두 끝나면 위와 같이 make를 통해서 소스를 컴파일하고 make install을 이용해서 컴파일이 완료된 실행 파일들을 리눅스 시스템 디렉토리에 설치합니다. 이 과정은 시스템의 성능에 따라 좌우되는데 보통 몇 분 정도의 시간이 소요됩니다.

PHP 설치하기

마지막으로 PHP를 설치할 차례가 왔습니다. 역시 주인공은 제일 마지막에 나타나는 법인가 봅니다.

```
[root@ezphp httpd]# cd /usr/local/src
[root@ezphp src]# tar xvzf php-5.2.5.tar.tar
```

설치 파일의 압축을 해제하고 아파치 웹 서버와 마찬가지로 컴파일을 위한 설정을 합니다.

```
[root@ezphp src]# cd php-5.2.5
[root@ezphp php]# ./configure --prefix=/usr/local/php \
> --with-mysql=/usr/local/mysql \
> --with-apxs2=/usr/local/apache2/bin/apxs
```

각 설치 옵션에 대한 설명은 다음과 같습니다.

옵션	설명
--prefix	설치될 디렉토리
--with-mysql	PHP에서 MySQL 사용
--with-apxs2	아파치 웹 서버의 모듈로 설치

[표 1-3] PHP 설치옵션

설정이 끝났으면 아파치 웹 서버와 마찬가지로 컴파일합니다.

```
[root@ezphp php]# make; make install
```

make 후에 컴파일이 완료될 때까지 기다렸다가 make install을 치는 번거로움을 없애기 위해서 위와 같이 한 번에 두 가지 명령을 지시하는 것도 가능합니다.

컴파일이 완료되면 PHP 설정 파일을 생성해 줍니다. PHP 설정 파일은 php.ini로 php.ini-dist와 php.ini-recommended 두 개의 설정 파일을 제공합니다. 이 둘 중에서 하나를 선택하여 사용해야 하는데 php.ini-dist는 호환성을 비롯하여 가장 무난하게 사용할 수 있는 설정 파일이며 php.ini-recommended은 PHP의 성능을 최적화하기 위한 설정 파일입니다. 무난한 사용을 위해서 php.ini-dist 파일을 다음과 같이 복사하도록 합니다.

```
[root@ezphp php]# cp php.ini-dist /usr/local/lib/php.ini
```

추후 모든 PHP의 설정은 복사된 /usr/local/lib/php.ini 파일을 통하여 수정할 수 있습니다.

길고 길었던 설치 과정이 모두 끝이 났습니다. 그러나 이것으로 모든 것이 끝난 것은 아닙니다. APM을 올바르게 사용하려면 아직 여러 가지 설정해야 할 부분이 남아 있기 때문입니다. 독자분들은 윈도우의 화려한 UI를 통해 수정하고 싶은 소망이 가득하겠지만 리눅스 버전이므로 텍스트에 익숙해지도록 합시다.

아파치 웹 서버 설정

아파치 웹 서버의 설정 파일은 /usr/local/apache2/conf/httpd.conf 파일입니다. 설치 방법은 외우지 못하더라도 설정 파일의 이름은 꼭 외워두어야 합니다. 왜냐하면 종종 웹 서버의 설정을 변경해야 할 일이 생기기 때문입니다. 위와 같은 설정 파일의 위치는 설치 방법에 따라 변할 수도 있습니다. 일반적으로 자동 설치된 경우에는 /etc/httpd/conf/httpd.conf인 경우가 많습니다.

httpd.conf 파일을 열어 다음과 같은 세 줄을 적당한 위치에 추가합니다. 만약 적당한 위치를 모르겠다면 파일의 제일 아랫부분에 넣도록 합니다.

```
LoadModule php5_module modules/libphp5.so
AddType application/x-httpd-php .php .html .inc
AddType application/x-httpd-php-source .phps
```

각 줄에 대한 설명은 다음과 같습니다.

① LoadModule php5_module modules/libphp5.so
PHP를 모듈로 등록한다.

② AddType application/x-httpd-php .php .html .inc
PHP 문서의 확장자를 지정한다. 여기에 등록된 확장자만 PHP 문서로 인식하여 처리된다.

③ AddType application/x-httpd-php-source .phps
PHP 소스 보기 확장자를 지정한다. 확장자가 .phps이면 소스 코드가 여러 가지 색깔의 문법 강조를 통해 예쁘게 표시된다. 즉, 소스를 외부에 공개할 때 이 확장자를 사용하면 유용하다.

위의 세 줄을 통해서 앞으로 아파치 웹 서버에서 PHP를 사용할 수 있게 되었습니다. 이 외에 편의를 위해서 httpd.conf 파일에서 DirectoryIndex를 찾아서 index.php 항목을 추가하도록 합니다.

```
DirectoryIndex index.html index.htm index.php
```

DirectoryIndex란 주소창에서 http://www.domain.com/directory/와 같이 디렉토리 형식으로 접근하였을 때 제일 먼저 찾는 웹 페이지 파일의 이름입니다. 즉, 위와 같이 설정한 경우 해당 디렉토리에서 index.html을 찾아보고 없다면 index.htm을 찾고 그래도 없다면 index.php를 찾아서 보여줍니다. 만약 모든 파일을 찾을 수 없다면 설정에 따라서 파일을 찾을 수 없다는 에러 페이지를 출력하거나 디렉토리 내의 파일 목록을 보여줍니다.

이제 웹 서버의 설정이 모두 끝이 났으니 아파치 웹 서버를 실행해 보겠습니다. 아파치 웹 서버가 /usr/local/apache2에 설치되었으므로 해당 디렉토리에서 설치 파일을 찾을 수 있습니다. 일반적으로 아파치 웹 서버의 관리 프로그램은 apachectl(apache control)이라는 이름을 갖습니다. 이 프로그램을 이용해 아파치를 시작하거나 중지하거나 아니면 다시 시작하는 등의 작업을 할 수 있습니다.

```
[root@ezphp php]# /usr/local/apache2/bin/apachectl start
```

아파치 웹 서버의 서비스가 시작되었다는 메시지를 보았다면 웹 브라우저를 통해 접속합니다.

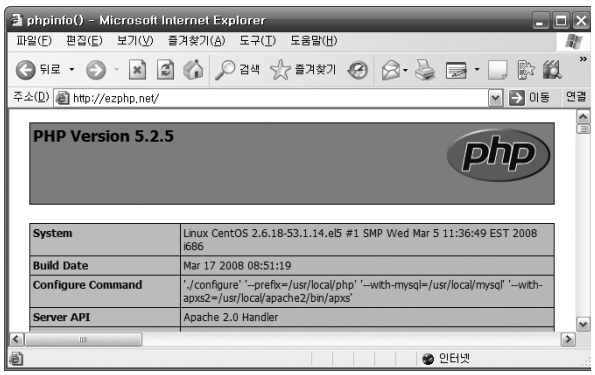
[그림 1-13] 아파치 실행화면

자동 설치 프로그램에서와 마찬가지로 /usr/local/apache2/htdocs/ 디렉토리 밑에 index.html 파일을 다음과 같이 수정합니다.

```
<? phpinfo(); ?>
```

자동 설치했을 때는 아파치 웹 서버의 홈 디렉토리가 /var/www/html이었으나 소스 설치는 아파치 웹 서버가 설치된 /usr/local/apache2 디렉토리 내의 htdocs 디렉토리가 홈 디렉토리가 됩니다. 만약 이 디렉토리를 옮기고 싶다면 httpd.conf 파일에서 DocumentRoot "/usr/local/apache2/htdocs"와 〈Directory "/usr/local/apache2/htdocs"〉 항목을 찾아서 각각 DocumentRoot "/var/www/html"과 〈Directory "/var/www/html"〉로 변경하면 됩니다.

수정된 index.html 파일을 통해서 PHP가 잘 동작하는지 확인해 봅시다.



[그림 1-14] PHP 실행확인

[그림 1-14]와 같은 화면을 볼 수 있다면 PHP가 올바르게 동작하는 것입니다. 앞서 자동 설치에서와 마찬가지로 [그림 1-14]의 웹 페이지에서 mysql 항목을 찾아보면 PHP와 MySQL이 연동되고 있는지도 확인할 수 있습니다.

APM 자동 실행 설정

아파치 웹 서버와 PHP가 모두 잘 동작하는 것을 확인하였습니다. 그런데 리눅스 시스템을 재부팅하게 되는 경우 MySQL과 아파치 웹 서버를 매번 새로 시작해 주어야 합니다. 앞서 자동 설치에서는 setup 프로그램을 통해서 간단하게 자동 실행하게 할 수 있었으나 소스 설치는 다음과 같이 /etc/rc.d/rc.local 파일에 다음과 같은 두 줄을 추가해야 자동으로 실행됩니다.

```
/usr/local/apache2/bin/apachectl start >&/dev/null
/usr/local/mysql/bin/mysqld_safe --user=mysql & >&/dev/null
```

이제 모든 설치 과정이 끝났습니다. 자동으로 설치하거나 rpm과 같은 패키지를 이용하는 방법 등 편리한 방법이 많이 있지만 소스 설치는 자신이 원하는 버전을 설치할 수 있고 자신의 시스템에 최적화된 APM을 설치할 수 있다는 장점이 있습니다. 그러나 초보자가 수행하기에 많이 어렵고 복잡하며 각 리눅스 버전 및 환경에 따라 설치 과정에서 여기에 언급되지 않은 문제가 발생할 수 있기 때문에 대부분은 여러 번의 시행착오를 거친 후에야 설치가 완료되는 경우가 많습니다. 따라서 처음에 시작할 때는 소스를 통한 설치보다는 자동 설치 프로그램을 이용하는 것을 추천합니다. 그러나 추후 중급 레벨이 되었을 때에는 꼭 이 방법을 통해 설치해보기 바랍니다.

여기서 잠깐

☑ CentOS 5.1에서 발생할 수 있는 에러

CentOS 5.1에서는 소스를 통한 설치 과정에서 에러가 발생할 수 있습니다. 이는 강화된 보안 정책으로 인한 것이며 다음과 같이 selinux 설정을 바꾸면 됩니다.

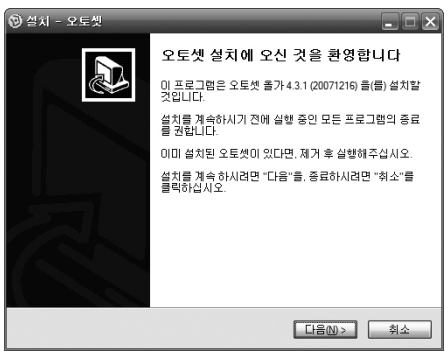
```
[root@ezphp ~]# vi /etc/selinux/config
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
[root@ezphp ~]# reboot
```

Section 05

윈도우 기반 APM 설치

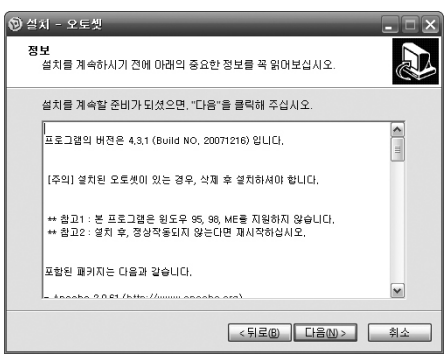
예전에는 APM을 설치하려면 반드시 리눅스가 필요했으며 리눅스와 APM은 서로 때려야 땔 수 없는 관계로 여겨졌었습니다. 그러나 리눅스라는 운영체제에 익숙하지 않은 많은 사용자의 요구로 윈도우용 APM이 등장했으며 초기의 윈도우용 APM은 불안정한 부분이 많았으나 현재는 리눅스에 버금가는 안정적인 성능을 나타내고 있습니다. 그래서 많은 소규모 홈페이지가 관리의 편의를 이유로 윈도우 기반에서 APM을 통해 웹 서비스를 제공하고 있습니다. 더욱이 웹 서비스를 제공하는 것이 목적이 아니라 PHP를 배우는 것이 목적인 경우, 윈도우용 APM은 매우 만족스러운 개발 환경이 아닐 수 없습니다. 이러한 이유로 많은 개발자가 윈도우용 APM 자동 설치 프로그램을 만들어 배포하고 있습니다. 대부분의 윈도우용 APM 자동 설치 프로그램이 무료로 제공되며 각 APM의 새로운 버전이 출시될 때마다 업그레이드된 설치 프로그램이 등장하곤 합니다. 이분들의 노력에 감사하는 마음을 가지면서 대표적인 윈도우용 APM 프로그램 중 하나인 AutoSet을 설치해 보도록 하겠습니다.

오토셋 홈페이지(http://www.autoset.org)에서 최신 버전의 AutoSet 프로그램을 다운받습니다. 다운받은 설치 파일을 실행하면 다음과 같은 설치 프로그램이 동작합니다.



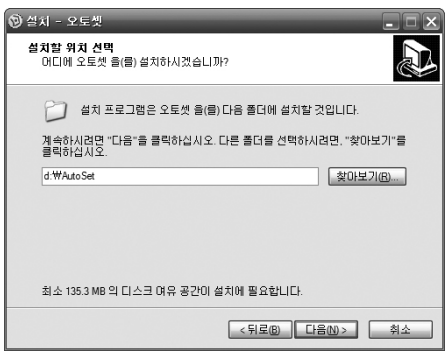
[그림 1-15] 오토셋 실행화면

만약 오토셋 프로그램을 설치하기 이전에 다른 윈도우용 APM 프로그램을 설치하였거나 현재 시스템에 웹 서버가 동작하고 있다면 반드시 제거한 후 설치하기 바랍니다.



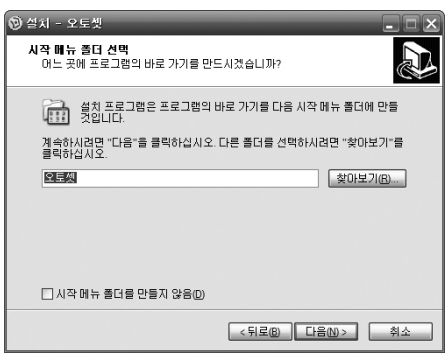
[그림 1-16] 오토셋 상세 정보

[그림 1-16]에서 보는 바와 같이 오토셋 버전의 상세한 정보(어떤 버전의 아파치 웹 서버가 포함되어 있는지 등)를 얻을 수 있습니다.



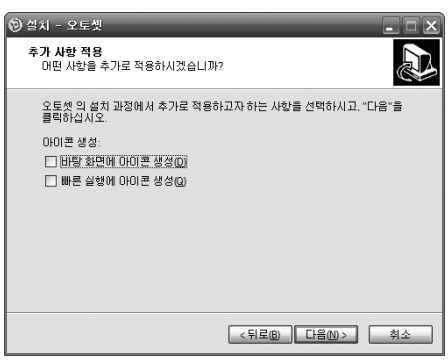
[그림 1-17] 오토셋을 설치할 디렉토리 설정

〈찾아보기〉를 통해 APM을 설치할 디렉토리를 설정합니다. 기본적으로는 C:\AutoSet 디렉토리에 설치됩니다. 될 수 있으면 운영체제가 있는 드라이브는 자제하는 것이 좋습니다.



[그림 1-18] 오토셋 프로그램 메뉴 등록 설정

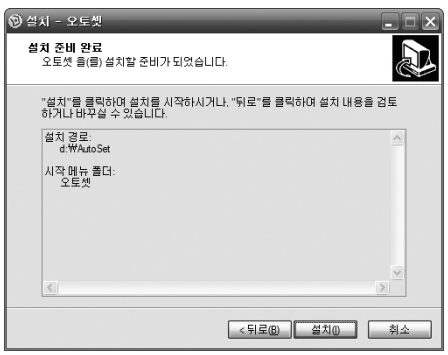
[시작]-[프로그램]에 등록되는 이름을 설정합니다.



[그림 1-19] 바로가기 아이콘 설정

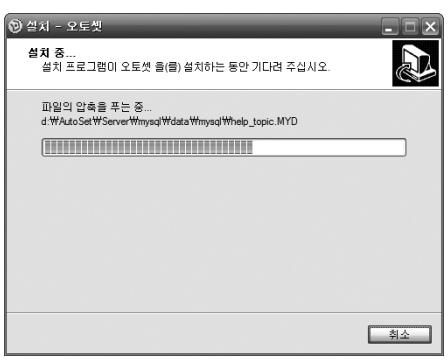
바탕화면과 빠른 실행줄에 아이콘을 생성할지를 설정합니다. 추후 APM을 자동으로 실행하도록 한

다면 굳이 아이콘을 생성할 이유는 없습니다. 그러나 매번 필요에 따라서 APM을 실행하고자 한다면 바로가기 아이콘을 만들어 두는 것이 편리합니다.



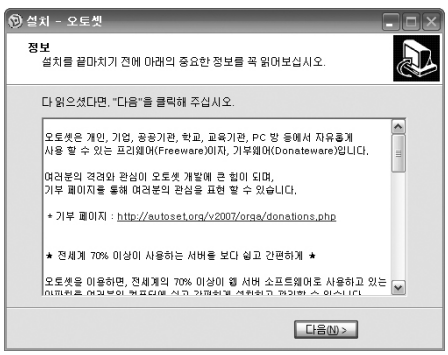
[그림 1-20] 오토셋 설치옵션 확인

이전 과정에서 선택한 옵션이 올바르게 출력되어 있는지 [그림 1-20]과 같이 확인합니다. 이제 설치 준비가 완료되었습니다. 〈설치〉 버튼을 클릭하면 본격적인 설치가 시작됩니다.



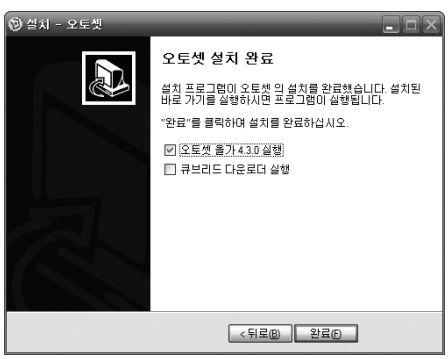
[그림 1-21] 오토셋 설치 중

본격적으로 윈도우 시스템에 APM을 설치합니다. 오토셋 버전에 따라 조금씩 다르지만 일반적으로 APM을 비롯하여 APM 운영에 도움을 주는 여러 가지 보조적인 프로그램이 함께 설치됩니다.



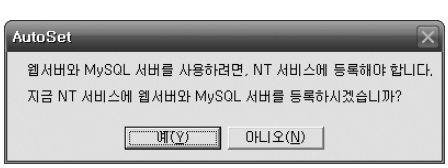
[그림 1-22] 오토셋 라이센스 정보

설치가 완료되면 [그림 1-22]와 같이 오토셋의 라이센스 정보가 나타납니다. 오토셋은 프리웨어이므로 개인, 기업, 공공기관의 구분없이 어느 곳에서나 자유롭게 사용할 수 있습니다. 오토셋 프로그램은 개인이 틈틈이 시간을 내어서 개발하는 프로그램이므로 더욱 빠른 업데이트를 원한다면 기부를 통해서 개발자에게 힘을 북돋아 줄 수 있습니다.



[그림 1-23] 오토셋 설치 완료

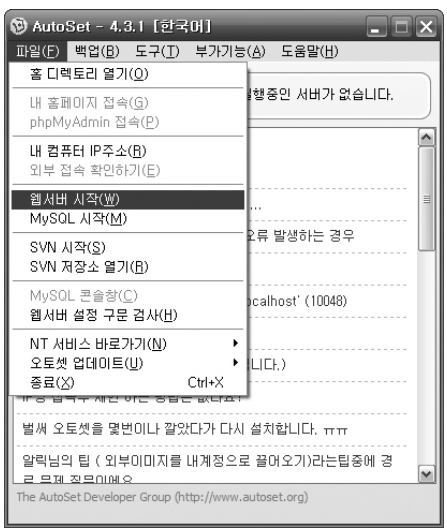
오토셋의 설치가 거의 완료되었습니다. 〈완료〉 버튼을 누르면 자동으로 오토셋 프로그램이 실행됩니다. 여기서 큐브리드는 관계형 데이터베이스 엔진에 객체지향 기능을 추가한 국산 무료 데이터베이스로 인터넷 서비스에 최적화된 데이터베이스를 목표로 하고 있습니다. 최근 국내에서는 MySQL 대신에 큐브리드를 이용하자는 움직임이 나타나고 있습니다. 만약 큐브리드에 관심이 있다면 체크하여 설치해보기 바랍니다. 또한 인터넷에서 큐브리드를 검색하면 여러 가지 유용한 문서를 제공받을 수 있습니다.



[그림 1-24] APM의 서비스 등록 여부 결정

오토셋은 NT 시스템에서 안정적으로 동작하는 프로그램입니다. 그래서 오토셋을 처음 실행하면 APM을 NT 서비스로 등록할 것인지를 [그림 1-24]와 같이 묻습니다. 만약 〈아니오〉를 선택한다면 필요할 때마다 매번 APM을 실행해 주어야 하고 〈예〉를 선택한다면 설치 이후 다시 APM을 실행해 줄 필요 없이 자동으로 실행됩니다.

이제 모든 설치가 완료되었습니다. 리눅스 기반의 APM 설치보다 훨씬 마음이 편한 것 같습니다. 이제 설치가 올발라졌는지 확인하기 위해 아파치 웹 서버와 MySQL 서버를 시작해 봅시다.

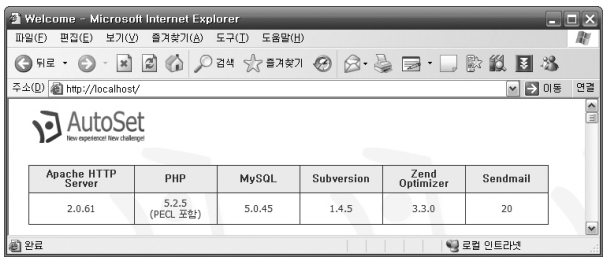


[그림 1-25] 오토셋을 이용하여 APM 실행

오토셋 프로그램을 이용하여 [그림 1-25]와 같이 "웹 서버 시작"과 "MySQL 시작"을 클릭하면 아파치 웹 서버와 MySQL 데이터베이스가 시작됩니다. 올바르게 시작되었으면 오토셋 프로그램의 상단에 [그림 1-26]과 같이 표시됩니다.

[그림 1-26] APM이 올바르게 동작하는 모습

이제 웹 브라우저를 띄워서 확인해 봅시다. 내 컴퓨터 시스템에 설치하였으므로 웹 브라우저의 주소창에 http://localhost를 입력합니다. 그 결과는 [그림 1-27]과 같으며 시스템에 설치된 프로그램의 각 버전 정보와 주요 기능과 설치 정보를 보여줍니다.



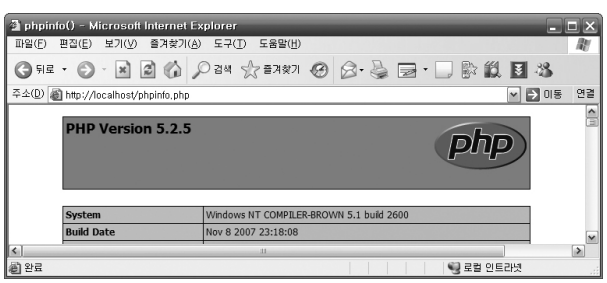
[그림 1-27] 오토셋의 초기 웹 페이지

여기서 잠깐

로컬 호스트

로컬 호스트라는 것은 자신의 컴퓨터를 의미하는 별명과 같은 것입니다. 즉, 웹 브라우저에 http://localhost라고 입력하면 웹 브라우저는 '내 컴퓨터로 접속하라는 말이구나'는 뜻으로 이해하여 인터넷을 사용하지 않고 바로 내 컴퓨터로 접속합니다. localhost를 IP로 나타내면 127.0.0.1과 같습니다. 웹 브라우저에 localhost 대신에 127.0.0.1을 입력해 보면 똑같은 결과를 얻을 수 있습니다. 그런데 많은 분이 이 부분에 대해서 자주 오해를 하곤 합니다. 바로 외부의 다른 컴퓨터에서도 localhost라고 입력했을 때 자신의 컴퓨터에 접속할 수 있을 것이라는 착각을 한다는 것입니다. localhost는 원격의 컴퓨터 입장에서는 원격 컴퓨터 자신이 localhost이기 때문에 내 컴퓨터에 접속할 수 없습니다. 즉, A와 B 컴퓨터가 있을 때 각각의 localhost는 A의 경우 A를, B의 경우에는 B를 의미한다는 것을 유념하기 바랍니다. IP로 127.0.0.1도 마찬가지입니다.

오토셋을 이용하여 APM을 설치했을 경우 홈 디렉토리는 설치 디렉토리의 public_html 디렉토리입니다. 즉, C:\AutoSet에 프로그램을 설치했다면 C:\AutoSet\public_html 디렉토리가 홈 디렉토리가 됩니다. 해당 디렉토리에는 phpinfo.php 파일이 존재하는데 이는 phpinfo() 함수를 출력하는 PHP 문서입니다. 따라서 웹 브라우저를 통해서 PHP 설치 정보를 확인해 보도록 합니다.



[그림 1-28] phpinfo() 함수 출력결과

Section 06

에디터 사용하기

앞서 PHP의 개발 환경을 구축했습니다. 이제 본격적으로 PHP 프로그램만 작성하면 되는데 PHP 프로그래밍을 하려면 어떤 툴이 필요할까요? 비주얼 스튜디오와 같은 값비싼 툴이 필요한 것일까요? 아닙니다. PHP 프로그램을 작성하려면 단지 괜찮은 텍스트 에디터만 있으면 됩니다. 텍스트 에디터란 말 그대로 글을 쓰고 수정하고 저장할 수 있는 프로그램을 말합니다. 가장 대표적인 텍스트 에디터가 여러분도 너무나 잘 알고 계시는 메모장입니다.

메모장으로 PHP 프로그램을 작성한다고?

가끔 PHP는 정말 대단하다는 생각이 듭니다. 무료 운영체제에 무료 웹 서버, 무료 데이터베이스를 사용할 수 있고 심지어 프로그램을 개발하기 위해서 무료인 메모장을 사용하면 되니 말입니다. (뭐 어떻게 보면 메모장은 유료 프로그램일지도 모르지만요 ^^) 리눅스 사용자라면 메모장이 아닌 vi 에디터를 떠올릴 수도 있을 겁니다. 메모장으로 프로그래밍이 가능은 하지만 실제로 누가 사용하겠느냐고 생각하는 분도 있을지 모르겠습니다. 그러나 믿어지지 않겠지만 많은 PHP 프로그래머들이 실제로 메모장을 사용하고 있습니다. 필자도 메모장을 더러 사용하는 편인데 주로 짧은 소스를 작성하거나 수정할 때 이용합니다. 하지만 소스 코드가 짧을 때는 고치기 쉽지만 소스가 길어지면 상당한 불편을 느끼게 됩니다. 그래서 여러분이 우려하는 바와 같이 메모장으로 PHP 프로그램을 개발하기에는 충분하지 않습니다. 왜냐하면 메모장은 말 그대로 잠깐 메모를 하기 위해 만들어진 프로그램이기 때문입니다. 그래서 여러 페이지 분량의 내용을 쓰고 수정하는 데는 적합하지 않습니다. 그래서 보다 편하게 프로그래밍을 하려면 텍스트 에디터를 하나 골라서 사용하는 것이 좋습니다.

어떤 텍스트 에디터가 좋은 에디터인가?

그렇다면 어떤 텍스트 에디터가 PHP 프로그래밍을 하기에 좋은 에디터일까요? 이 문제에 대한 대답은 한마디로 말하기가 정말 어렵습니다. 왜냐하면 프로그래머마다 취향이 다르기 때문입니다. 그 예로 세상에는 vi 에디터가 가장 좋은 에디터라고 말하는 사람들이 있습니다. 그러나 리눅스 초보자들이 vi 에디터를 처음 접하면 마치 문과대학 학생이 공대생이 사용하는 공학용 계산기의 끄는 법조차 알 수 없듯이, vi 에디터의 사용법을 몰라서 이리저리 헤매는 것을 어렵지 않게 볼 수 있습니다. 반면에 vi 에디터에 익숙한 리눅스 사용자들은 GUI로 무장된 다른 에디터들이 다소 번거롭고 불편하게 느껴질 수도 있습니다. 실제로 vi 에디터에 익숙해지면 키보드와 마우스를 같이 사용해야 하는 GUI 에디터가 더 번거롭게 느껴지기도 합니다. 왜냐하면 vi 에디터는 마우스를 전혀 사용하지 않고 모든 기능을 키보드만으로 처리할 수 있기 때문입니다. 이처럼 프로그래머마다 성향이 달라서

어느 에디터가 좋다고는 말씀드리기 곤란합니다. 그래서 가장 좋은 에디터는 손에 익은 에디터가 아닐까 생각합니다.

좋은 에디터의 조건

처음 PHP를 접하고 텍스트 에디터라고는 메모장밖에 모르는 일부 독자를 위해 필자 마음대로 정한 좋은 에디터의 조건 5가지를 알려주고자 합니다. PHP 프로그래밍에 좋은 에디터의 조건 5가지는 다음과 같습니다.

① 찾기와 찾아서 고치기 기능이 있는 에디터
② 구문 강조 기능이 있는 에디터
③ FTP 연결을 통한 원격 파일 생성 및 수정 기능이 있는 에디터
④ 실행취소와 다시 실행이 횟수에 상관없이 자유로운 에디터
⑤ 소스 앞에 몇 번째 줄인지 표시가 되는 에디터

첫 번째로 특정 단어를 찾아서 고치기 쉬운 에디터입니다. 웹 프로그래밍을 하다 보면 긴 소스 코드 중에서 특정 단어를 찾아서 고쳐야 하는 경우가 많고 그뿐만 아니라 전체 파일에서 특정 단어를 다른 단어로 모두 고쳐야 하는 경우도 많이 발생합니다. 이런 경우 단어 찾기나 찾아서 고치기(Replace) 기능이 없으면 일일이 찾아서 수정해줘야 합니다.

두 번째로 구문 강조 기능이 있는 에디터입니다. 구문 강조 기능이란 PHP에서 사용하는 키워드나 함수 등 중요한 단어들을 다른 색깔로 표시해주는 기능을 말합니다. 메모장처럼 흰색 바탕에 검은색 글씨로만 적혀 있으면 눈에 쉽게 띄지 않는 것들이 성질별로 다른 색으로 표시되는 에디터를 사용하면 더욱 쉽게 소스 코드를 이해할 수 있습니다. 심지어는 오류가 있는 소스 코드를 쉽게 고칠 수 있게 도와주기도 합니다.

세 번째로 FTP 연결을 통해서 원격의 파일을 직접 수정하거나 새로 작성할 수 있는 에디터입니다. 웹 프로그래밍의 특징으로 원격에 존재하는 파일을 수정하거나 원격지에 새로 파일을 생성해야 하는 경우가 매우 많습니다. 이때 FTP 기능을 지원하지 않는 에디터를 사용하는 경우 FTP 프로그램을 별도로 이용하여 파일을 다운받은 다음 에디터를 이용하여 수정한 후 다시 FTP 프로그램을 이용하여 업로드해야 하는 번거로운 일이 생겨나게 됩니다.

네 번째로 실행취소 및 다시 실행이 횟수에 상관없이 자유로운 에디터입니다. 프로그래밍을 하다 보면 작성했던 소스 코드를 원상태로 되돌려야 하거나 되돌렸던 소스 코드를 다시 최근 작성했던 코드로 되돌려야 하는 경우가 허다하게 발생합니다. 실행취소 기능을 제공하지 않는 경우 예전에 작성했던 소스 코드를 되새김질하며 새로 작성해야 할지도 모릅니다.

마지막으로 줄 번호 기능을 지원하는 에디터입니다. 프로그래밍을 하다가 에러가 발생하는 경우 에러 메시지로 해당 소스 코드의 줄 번호를 알려줍니다. 만약 줄 번호 기능이 없다면 위에서부터 일일이 한 줄씩 세어보아야 하기 때문에 불필요한 시간을 많이 허비하게 됩니다. 만약 줄 번호가 표시된다면 해당 소스 코드의 줄로 이동하여 쉽게 수정할 수 있습니다.

이 5가지 기능을 모두 가진 에디터라면 PHP 프로그래밍을 하는데 어떤 무리도 없으리라 생각합니다. 하지만 여기에 적어두지는 않았지만 무엇보다도 중요한 것은 여러 프로그램을 사용해보고 자신에게 잘 맞는 에디터를 찾는 것입니다.

가장 많은 사람이 쓰는 텍스트 에디터는 무엇일까?

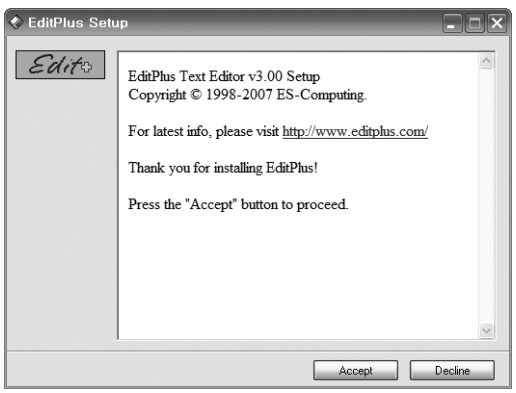
우리가 인터넷 쇼핑몰에서 물건을 구매할 때 좋은 물건인지 판단하는 가장 유용한 방법의 하나는 많은 사람이 구매하는 물품을 고르는 것입니다. 많은 사람이 관심을 두고 구매한다는 것은 구매하고 싶을 만큼 매력적이라는 의미가 될 수 있기 때문입니다. 물론 너무 싼 가격이라서 무턱대고 구매하고 보는 물건도 있지만 그런 물건은 구매 수기가 형편없기 마련이라 장기적으로는 대박 상품이 되지 못합니다. 이런 면에서 에디터를 선택할 때 가장 많은 사람이 사용하는 텍스트 에디터를 알고 있다면 도움이 될 수 있을 것입니다.

필자가 시장조사 기관에 근무하는 것도 아니고 웹 컨텐츠 검색 능력도 부족한 편이라 확실한 증거를 제시할 수는 없지만 전 세계 시장을 생각한다면 울트라에디트(UltraEdit)란 프로그램이 가장 많은 사람이 애용하는 프로그램이라고 할 수 있을 것 같습니다. 울트라에디트는 디자인적 측면에서는 그다지 좋은 평점을 주고 싶지는 않지만 텍스트 에디터로서의 기능은 매우 만족스러우며 특히 수십, 수백 MB 분량의 텍스트 문서를 처리하는 능력은 가히 타의 추종을 불허합니다. 국내에서도 울트라에디트의 인기는 대단하며 필자를 비롯하여 수많은 개발자가 울트라에디트를 사용하고 있습니다. 그러나 PHP 프로그래머에게 특히 사랑을 받는 국산 텍스트 에디터가 있으니 바로 에디트플러스(EditPlus)라는 프로그램입니다. 사실 개인적으로는 울트라에디트보다 에디트플러스에 점수를 더 주고 싶습니다.

에디트플러스 설치하기

개인적인 취향이지만 객관적으로도 충분히 추천받을 자격이 있는 텍스트 에디터이므로 에디트플러스의 사용법에 대해 설명하고자 합니다. 에디트플러스는 앞서 말씀드린 좋은 에디터의 조건을 모두 갖추고 있으며 디자인도 무난한 매우 훌륭한 텍스트 에디터입니다. 단, 유의할 것은 에디트플러스는 상용 프로그램이라는 것입니다. 하지만 30일간 무료로 사용할 수 있고 30일이 지나도 등록하라는 메시지만 뜰뿐 실제로는 무제한으로 사용할 수 있습니다.

에디트플러스를 설치하기 위해서 에디트플러스 홈페이지(http://www.editplus.com)에서 평가판 프로그램을 다운받습니다. 다운받은 설치 파일을 실행합니다.

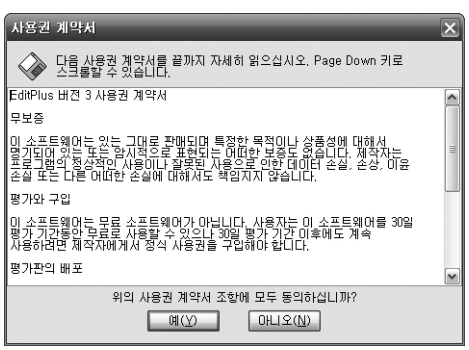


[그림 1-29] 에디트플러스 설치 프로그램

〈Accept〉 버튼을 클릭하여 설치를 시작합니다. 설치 프로그램의 압축이 해제되면서 본격적인 설치 프로그램으로 전환됩니다.

[그림 1-30] 에디트플러스 언어 설정

에디트플러스의 언어를 설정합니다. 여기서 선택한 언어에 따라 메뉴가 한글 혹은 영어로 표시됩니다. 필자는 자랑스러운 대한건아이므로 〈한글〉을 택하겠습니다 ^^. 에디트플러스는 한글과 영문 버전을 지원하며 설치 후에도 언어 설정을 변경할 수 있습니다.

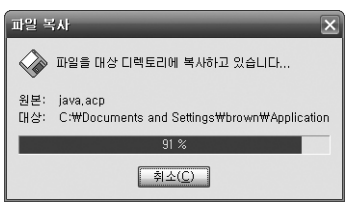


[그림 1-31] 에디트플러스의 라이센스

앞서 말씀드렸듯이 에디트플러스는 상용 프로그램입니다. 30일의 평가기간을 거치고 나서 계속해서 사용하고자 할 때는 더 좋은 에디터 개발을 위해서 정식 사용권을 구매하는 것을 고려해보기 바랍니다.

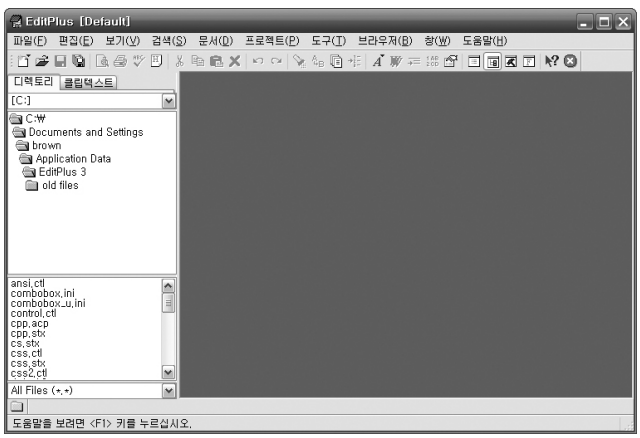
[그림 1-32] 에디트플러스 설치 디렉토리 설정

에디트플러스 프로그램을 설치할 디렉토리를 설정합니다. INI 파일 디렉토리의 경우 에디트플러스 환경 설정 파일이 저장될 위치이며 구문 디렉토리는 PHP나 C/C++ 그리고 자바와 같은 프로그래밍 언어들의 구문 강조 기능과 자동완성 기능과 같은 설정 파일이 존재하는 위치를 말합니다.



[그림 1-33] 에디트플러스 프로그램 설치

설치할 디렉토리를 설정하고 나면 해당 디렉토리로 에디트플러스 프로그램이 설치됩니다. 설치된 에디트플러스의 모습은 다음과 같습니다.

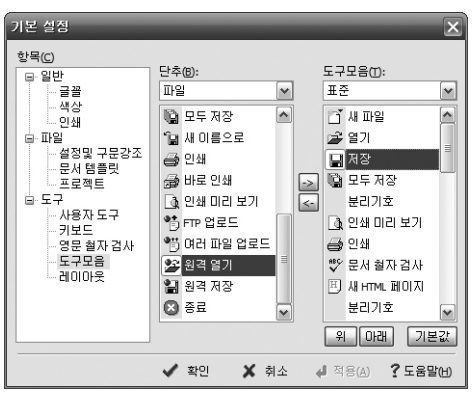


[그림 1-34] 에디트플러스 실행화면

에디트플러스 프로그램은 5가지 조건을 충족하는 기능 이외에도 매우 다양한 기능을 제공합니다. 이러한 기능들은 직접 하나둘씩 사용해가면서 익히도록 하고 여기서는 한 가지 매우 유용한 설정을 하나 해보도록 하겠습니다.

앞서 말씀드렸듯이 PHP 프로그래밍은 주로 원격 파일을 생성하거나 수정하므로 FTP 기능을 자주 사용하게 됩니다. 메뉴 설정을 바꾸어 상단 메뉴에 FTP 아이콘을 추가하도록 해보겠습니다.

메뉴에서 [도구]-[기본 설정]을 선택하면 에디트플러스의 환경 설정을 할 수 있습니다.



[그림 1-35] 에디트플러스 기본 설정

좌측의 항목에서 도구 모음을 선택하면 [그림 1-35]와 같이 메뉴의 도구 모음을 수정할 수 있는 화면이 나타납니다. 여기서 좌측 단추에서 '원격열기'를 선택하고 도구 모음에 사이에 끼워넣을 부분을 선택합니다. 그 후 중간 부분의 오른쪽 화살표(→)를 클릭하면 해당 위치로 이동합니다. 이와 마찬가지로 '원격저장'을 선택하여 도구 모음에 끼워 넣도록 합니다. 설정 후 〈확인〉 버튼을 누르고 나와보면 툴 바에 새로운 아이콘 두 개가 추가된 것을 알 수 있습니다.

[그림 1-36] 추가된 FTP 아이콘

이제 앞으로 원격 파일을 열어서 수정하고 싶을 때는 [그림 1-36]의 아이콘을 선택하면 됩니다.

이제 텍스트 에디터를 다 깔았는데 그다음은 어떻게 하지?

훌륭한 텍스트 에디터도 설치를 완료했습니다. 이제 새 문서를 열고 하얀 바탕 위에 주옥과 같은 소스 코드를 한 줄씩 작성해 나가면 됩니다. 그러나 저장할 때는 반드시 파일의 확장자를 php나 html로 저장해야 합니다. 아파치의 설정 부분에서 언급하였듯이 PHP 문서로 인식하는 것은 등록된 확장자뿐이기 때문에 설정해두지 않은 파일 확장자로 지정하는 경우 PHP로 인식되지 않습니다.

윈도우용 텍스트 에디터 이외에 리눅스에서 사용할 수 있는 편리한 에디터로는 대표적으로 두 가지가 있습니다. 에디터 계의 전설로 불리는 vi 에디터와 화려한 기술로 vi의 아성을 뒤쫓는 Emacs입니다. 리눅스 사용자라면 vi는 필수이므로 기본적인 기능은 반드시 알아두도록 합시다.

Section 07 PHP 개발 환경 테스트

PHP 프로그래밍을 하기 위한 환경이 모두 갖춰졌습니다. 프로그램을 작성할 에디터도 하나 구했고 작성한 문서를 테스트할 웹 서버도 설치했습니다. 이제는 직접 소스 코드를 입력해보고 PHP가 잘 동작하는지 확인해 봅시다.

우리는 윈도우와 리눅스 양쪽 모두에서 PHP 개발 환경을 구축하는 방법을 배웠습니다. 그러나 이제부터는 이 책에서 "APM 환경"이라고 하면 특별한 언급이 없으면 윈도우 기반의 APM 환경을 의미하는 것으로 정하겠습니다. 초보자들이 쉽게 접근하려면 윈도우 기반이 쉽게 느껴질 것이기 때문입니다. 그렇다고 윈도우 기반 APM이 리눅스 기반보다 많이 부족하거나 크게 다른 것이 아니라는 점을 명심해주길 바랍니다.

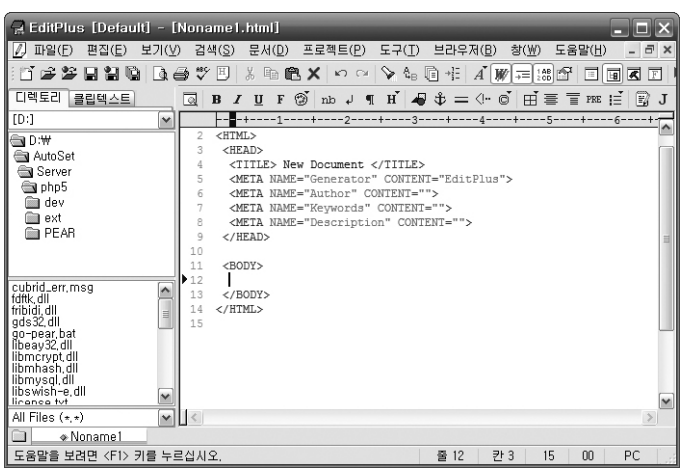
비스타가 출시된 지도 꽤 지난 시점이니 윈도우 98 및 ME 사용자는 죄송스럽지만 논외의 대상으로 생각하고 설명하겠습니다. 먼저 아파치 웹 서버가 실행되지 않았다면 웹 서버를 실행합니다. 오토셋 프로그램을 기준으로 말씀드리면 [파일]-[웹 서버 시작]을 선택하도록 합니다. 아파치 웹 서버

가 실행되었다면 웹 브라우저를 실행하고 localhost라고 입력해 봅시다.

설치한 지 얼마 지나지 않았지만 아직 잘 돌아가는 것을 확인했다면 디폴트 페이지를 수정하여 새로운 페이지로 한번 바꿔 보도록 합시다. 이 파일을 수정하려면 어떤 파일을 고쳐야 할까요? 이미 언급한 내용이지만 까마귀 고기를 자주 드시는 독자들을 위해 C:\AutoSet\public_html 디렉토리로 이동해 봅시다. 이 디렉토리는 아파치 웹 서버의 홈 디렉토리입니다. 만약 설치를 다른 곳에 하여 해당 디렉토리가 존재하지 않는다면 오토셋 프로그램을 이용하여 해당 디렉토리를 찾을 수 있습니다. [파일]-[홈 디렉토리 열기]를 이용하면 해당 디렉토리가 열립니다. 디렉토리 안을 살펴보면 index.html이라는 파일이 있는 것을 확인할 수 있습니다. 앞서 httpd.conf를 수정하면서 언급했듯이 URL의 끝이 파일이 아니라 디렉토리 형태로 접근하는 경우 DirectoryIndex 옵션에 지정된 파일을 먼저 찾습니다.

우리가 현재 보는 디폴트 웹 페이지가 실제로는 http://localhost/index.html의 경로를 가지지만 기본 설정 때문에 http://localhost/라는 디렉토리 형식으로 접근해도 자동으로 index.html이 인식됩니다. 이와 마찬가지로 http://localhost/test/index.html은 http://localhost/test/와 같은 결과를 나타냅니다.

오토셋 프로그램은 기본적으로 홈 디렉토리에 여러 개의 파일이 많이 있습니다. 홈 디렉토리 내의 모든 파일을 삭제하고 새 HTML 페이지를 하나 열어보도록 합니다.

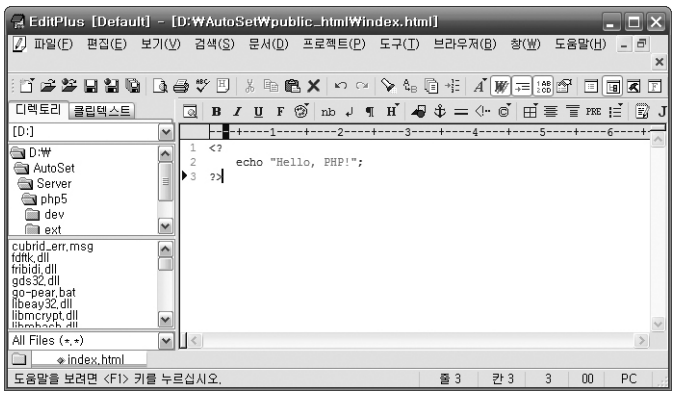


[그림 1-37] 불러온 index.html 파일

HTML 문서 안의 내용을 모두 지우고 다음과 같이 입력하도록 합시다. (내용을 모두 선택하는 방법은 Ctrl + A 입니다.)

```
<?
  echo "Hello, PHP!";
?>
```

이를 제대로 정확히 입력하면 다음과 같습니다.



[그림 1-38] 첫 번째 프로그램

우리는 최초의 PHP 프로그램을 작성했습니다. 우선은 글자를 틀리지 않고 정확하게 입력하는 것에만 신경을 쓰기 바랍니다. 아직 코드의 의미를 알 필요는 없습니다. 소스 코드의 의미는 다음에 알게 될 것이니 초등학교 때 받아쓰기를 하듯이 또박또박 꼼꼼히 옮겨 적도록 합시다. 무얼 하는 놈인지도 모르는 여러분의 최초의 PHP 프로그램은 다음과 같은 결과를 나타냅니다.

[그림 1-39] 첫 번째 프로그램 실행결과

Section 08

Hello, PHP!

PHP 세상에 오신 걸 환영합니다.

필자는 지금 버선발로 독자 여러분에게 마중 나가고 싶은 마음입니다. 지면이라는 제약이 없다면 좀 더 화려한 환영식을 해주고 싶지만 흑백 지면의 한계로 이렇게밖에 반기지 못함을 이해해주기

바랍니다. 그러나 한편으로는 이제 막 PHP에 입문하려는 여러분이 이 책에 기대가 큰 만큼 막중한 부담이 어깨를 짓누르는 듯합니다.

이제 여러분은 PHPer가 되었습니다. 이제 막 시작하려는데 무슨 PHPer냐고 말하는 독자들도 있으리라 생각합니다. 그러나 PHP 개발 환경을 갖추고 첫 번째 프로그램도 작성해 보았으니 여러분은 누가 뭐라 해도 PHP 프로그래머, 즉 PHPer 입니다. PHPer라는 말은 "피에이치피어"라고 읽어 주길 부탁합니다. 간혹 "피에이치퍼"라고 읽는 사람들이 있는데 어감이 전자의 경우가 더 좋아서 그렇게 부르길 권하고 싶습니다. 하지만 근래에는 PHPer란 말도 잘 안 쓰이는 것 같기도 합니다.

기분이 어떠신가요? 가슴이 설레고 어서 PHP를 배우고 싶어 안달이 나시나요? 아니면 어려울 것 같아 조금은 두려우신가요? 가슴이 설렌다면 정말 좋은 마음가짐을 가지는 것이며 어쩌면 조만간 정말 훌륭한 PHP 프로그래머가 될지도 모르겠습니다. 만약 두렵다면 전혀 두려워할 것이 못 된다는 것을 조금씩 이 책을 통해서 차례차례 증명해 보이겠습니다. PHP를 공부하는데 가장 중요한 것은 무엇보다도 자신감! PHP를 정복할 수 있다는 자신감을 갖는 것입니다.

Q PHP는 많이 어려운가요?

PHP는 어렵고 난해한 프로그래밍 언어가 아닙니다. 공과대학이나 컴퓨터학과에서 신입생 때 배우는 C 언어보다 10배 아니 100배는 쉽습니다. C 언어를 공부하다가 '나는 프로그래밍과 안 맞는 사람인가 보다.'라고 생각하는 사람이 있다면 오히려 PHP를 먼저 공부해보길 권해 드립니다. 더욱 쉬운 PHP를 통해 프로그래밍에 대한 감각을 키우고 나면 C 언어도 어렵지 않음을 느낄 수 있게 될 것입니다.

Q PHP가 좋다던데 어떻게 공부해야 하나요? 또 어떤 책이 좋아요?

PHP를 처음 시작하려고 하는 사람들이 필자에게 가장 많이 묻는 말 중 하나입니다. 그러나 필자의 생각으로는 공부방법이나 책 어느 것도 그리 중요하지 않다고 생각합니다. 필자는 한때 PHP 홍보대사로 불릴 만큼 여기저기 심지어 프로그래밍과 관련 없는 문과 학생에게까지 PHP를 공부해보길 권하고 다녔습니다. 그런데 필자의 말발에 속아(?) 넘어간 많은 사람이 그저 책만 사놓고 몇 페이지 보다가는 "어려워서 못하겠다!"라고 포기하는 것을 자주 보았습니다. 이런 사람들은 PHP가 정말 어려워서가 아니라 시작할 때부터 마음에 새로운 것에 대한 두려움을 가지고 시작했기 때문이라고 생각합니다. 이렇게 쉽게 포기하는 사람들을 만날 때마다 "자신감을 갖고 한번 무작정 도전해 보세요. PHP는 정말 쉬워요!"라고 말을 하면 대부분의 사람은 "원래 어느 정도 터득한 사람은 항상 저렇게 말하곤 한다."라며 고개를 저어대곤 했습니다.

물론 컴맹이 PHP를 바로 시작하는 것은 불가능에 가까울지도 모릅니다. 메모장도 잘 다루지 못하는데 에디트플러스라는 복잡한 기능의 텍스트 에디터를 써야 하고 이러한 기본적인 툴도 잘 다루지 못하는데 게다가 프로그래밍이라니, 말이 되지 않는 소리로 들릴 수 있습니다. 그러나 컴퓨터를 좋

아하고 컴퓨터와 친하게 지내는 사람이라면 충분히 소질이 있다고 생각합니다.

PHP는 누군가 가르쳐주는 것을 익힌다고 되는 것이 아닙니다. PHP는 자신이 스스로 공부하는 것입니다. 수업을 한 시간 듣는 것보다 직접 프로그램을 30분 짜보는 것이 훨씬 낫습니다. 혼자서 책도 보고 직접 프로그램도 작성해보고 잘 모르는 것이 있으면 주변에 물어보기도 하고 그것도 아니면 인터넷 게시판에 도움을 요청하면서 무럭무럭 성장해 갈 수 있습니다. 그런데 가끔 질문을 보면 하나에서 열까지 모두 가르쳐주기를 바라는 분들을 쉽게 만날 수 있습니다.

Q "게시판을 만들고 싶어요. 게시판 만드는 법 좀 메일로 보내주세요."

Q "이 소스에 에러가 있어요. 고쳐서 메일로 보내주세요."

Q "리포트 때문에 급한데 이러저러한 프로그램을 짜서 메일로 보내주시면 안 될까요?"

좀 냉정하지만 이런 질문은 거절합니다. 아니 실제로 이런 것들은 질문이라고 할 수도 없습니다. 필자뿐만이 아니라 대부분의 선배 개발자들이 가장 싫어하는 질문 중 하나가 위와 같은 질문입니다. 마치 감나무 밑에 누워서 감 떨어지길 기다리는 사람들 같습니다. 필자가 PHP를 처음 시작했을 때는 발간된 책도 많지 않았고 또한 필자가 너무 모르는 게 많아서 무작정 사람들에게 물어보는 경우가 많았습니다. 궁금한 것이 무척 많을 때여서 "이게 뭐예요? 저게 뭐예요?" 하는 식으로 많은 질문을 했습니다. 그런데 메일도 보내보고 게시판에 글도 써봤지만 답변을 잘 안 해주는 경우가 많았습니다. 그래서 오기로 그나마 몇 권 출간된 책들을 뒤적거려보고 인터넷도 뒤지고 하면서 나름대로 해답을 찾곤 했습니다. 그래도 모르는 것이 있으면 어쩔 수 없이 다시 질문을 했습니다.

Q "이걸 구현하기 위해서 이렇게 해보았는데 이러한 오류가 발생합니다. 혹시 이런 이유 때문인가요?"

그런데 이렇게 질문하니까 신기하게도 아주 친절한 말투로 질문한 지 얼마 지나지 않아 답이 오곤 했습니다. 필자가 어느 정도 자신감이 생겨서 누군가의 질문에 답을 할 수 있는 위치가 되었을 때 누군가가 "이게 뭐예요?" 하고 물었습니다. 처음에는 내가 누군가를 가르쳐 줄 수 있다는 기분에 상세히 알려주곤 했는데 시간이 지나니까 그러한 질문들은 나도 모르게 무시하게 되었습니다. 공부는 누군가 가르쳐주는 것을 외우고 습득하는 것이 아니라 어떤 문제를 스스로 해결해 나가는 과정 속에서 진정한 깨달음을 얻는 것입니다. 충분히 검색을 통해서 알 수 있는 것들을 찾아보지도 않고 무턱대고 질문을 올리는 사람들에게 PHP의 정복은 머나먼 여정이 될 수밖에 없습니다. 언젠가 필자의 친구 중 한 명이 갑자기 PHP를 한다고 온종일 꼬박 컴퓨터 앞에 앉아 있는 것을 보았습니다. 옆에서 보고 있기가 뭐해서 도와줘야지 하고 친구에게 갔는데 자기가 물어볼 때까지는 어떤 도움도 주지 말라고 당부를 하는 것이었습니다. 그래서 그냥 옆에서 친구가 하는 것을 물끄러미 보고만 있었는데 밤이 늦어서야 친구가 물었습니다.

"도저히 왜 안 되는지 모르겠다. 답 말고 힌트 좀 줘봐."

그런데 그 친구 끝까지 고집인 것이 해답은 싫으니까 힌트만 달라는 것이었습니다. 그 친구는 이틀 만에 정말 말 그대로 광소(狂笑)를 터트렸습니다. 3일째 되던 날부터 인터넷에서 자료를 찾아서 MySQL 공부를 하더니 금세 MySQL을 이용한 북 마크 프로그램이라면서 보여주었습니다. 다음번에는 방명록을 만들겠노라 하면서 말입니다. 그 친구는 천재가 아닙니다. 그 친구는 그저 남들보다 PHP를 하고 싶은 열정이 많았을 뿐입니다. 문제가 발생했을 때 남에게 의지하기보다는 스스로 해결할 수 있는 능력을 기를 줄 알았기 때문에 짧은 시간에 눈에 띄는 성과를 얻을 수 있었던 것입니다.

PHP를 모두 배우는 데에는 정말 많은 시간이 소요됩니다. 한 달, 두 달 가지고는 어림도 없습니다. 그러나 웬만한 프로그램 하나 작성하는 데는 그리 많은 시간이 필요하지 않습니다. 그러나 누워서 받아먹기만 하는 사람은 스스로 공부하는 사람보다 몇 배 또는 몇 십 배 차이가 나게 되어 있습니다. 실제로 PHP뿐만이 아니라 다른 언어나 스크립트도 마음가짐이 제일 중요합니다.

2000년경 필자의 강좌 홈페이지에 쓰던 게시판을 만들었던 이기현이란 친구가 있습니다. 필자의 PHP 홍보에 힘입어 PHP를 시작하긴 했으나 PHP가 어려울 것 같아서 손도 못 대던 친구였습니다. 필자가 계속해서 "PHP 공부 좀 해라! 해라!" 해서 마지못해 시작하더니 며칠 있다가 "PHP 정말 쉽네…." 라는 말을 했습니다. 그 친구가 PHP를 배운지 두 세 달쯤 밖에 안되었을 즈음, 정말 그럴듯한 PHP 게시판을 만드는 것을 보고 놀라움을 금치 못했던 적이 있었습니다. 이처럼 여러분도 자신감과 열정을 가지고 있다면 분명히 좋은 결과가 있으리라 생각합니다.

Chapter

02

PHP 문법

이 장에서는 컴퓨터에 계산을 시키거나 어떤 명령을 수행시키기 위해 컴퓨터와 대화하는 방법에 대해서 알아봅니다. PHP 프로그래밍 언어는 컴퓨터와 대화할 수 있는 컴퓨터 언어이지만, 우리가 대화하려면 외국어를 배우듯이 컴퓨터와 말하는 방법을 하나씩 배워야 합니다. 이것이 PHP 문법입니다. 외국어라는 말에 너무 두려워하지 않아도 됩니다. 컴퓨터는 매우 간단하고 명확한 언어를 사용하기 때문에 문법이 많지도, 그리 어렵지도 않습니다. 이 장에서는 문장을 구성하는 문법에 대해서 다루고, 지금 당장은 필요 없지만 알면 크게 도움되는 함수는 4장에서 살펴보겠습니다. 이 장은 가장 기본적인 과정이므로 만약 이해가 잘 안 된다면 이해될 때까지 여러 번 반복해서 보기 바랍니다.

Section 01

컴퓨터와 언어

언어(Language)는 자신의 생각이나 느낌을 표현하기 위한 문자나 음성 등의 수단과 체계를 의미합니다. 이 말은 단순히 아무렇게나 긁적거려둔, 알 수 없는, 혹은 체계가 없는 문자나 음성은 언어가 될 수 없다는 말로 해석할 수 있습니다. 언어란 아무렇게나 조합된 것이 아닌, 그 언어를 사용하는 사람들이 서로 오랜 시간 동안 만들어온 규칙과 체계를 갖춘 문자와 음성을 의미합니다.

2004년 최고의 인기를 얻었던 KBS 드라마 〈미안하다, 사랑한다〉를 보고 어느 누군가가 "미사"라고 짧게 명명하기 시작했습니다.

친구 A : "너 어제 '미사'봤어?"
친구 B : "응! 너무 감동이었어"
친구 C : "너희 카톨릭 신자야?"

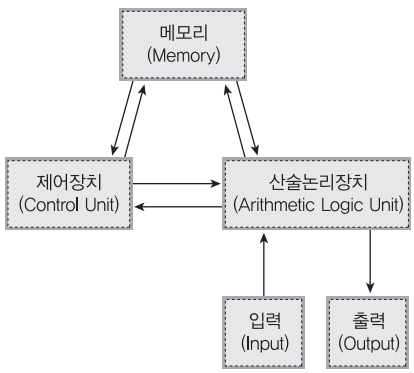
친구 A와 B는 서로 의사소통을 하는데 아무런 문제가 없었습니다. 그러나 친구 C는 비록 〈미안하다, 사랑한다〉라는 드라마를 알고 있었을지 몰라도 이를 줄여 '미사'라고 부르고 있음을 알지 못하였기 때문에 제대로 의사소통이 이루어지지 않았습니다. 이처럼 언어는 일종의 약속이기 때문에 올바른 의사소통을 위해서는 해당 언어를 사용하는 대부분의 사람이 어떤 말에 대해 같은 의미를 떠올릴 수 있도록 합의가 이루어져야 합니다.

기계어와 어셈블리어

어린 학생들은 종종 컴퓨터가 영어를 할 줄 안다고 생각합니다. 왜냐하면 컴퓨터 언어들이 대부분 알파벳을 사용하고 많은 영어 단어가 실제로 컴퓨터 언어에 사용되고 있기 때문입니다. 그러나 한국인의 주 언어가 한국어 듯이 컴퓨터의 주 언어는 기계어(Machine Language)입니다. 기계어는 단어 자체에서 풍기는 이미지 그대로 기계가 사용하는 언어입니다. 그래서 컴퓨터는 이 기계어를 곧바로 이해할 수 있습니다. 기계어가 사용하는 기호는 딱 두 개밖에 없습니다. 전기가 흐르느냐? 흐르지 않느냐? (이 두 가지 기호를 인간들의 편의를 위해 전기가 흐르는 경우를 "1"로 흐르지 않는 경우를 "0"으로 정했습니다.) 이러한 두 기호의 조합을 통해서 컴퓨터는 단어를 구성하기도 하고 문장을 만들기도 합니다.

컴퓨터의 발명 초창기에는 인간과 컴퓨터가 대화하기 위해서 인간이 컴퓨터의 언어인 기계어를 배웠습니다. 그래서 최초의 컴퓨터인 에니악(ENIAC)과 대화하기 위해서 컴퓨터 기술자들은 6천여 개의 전기 케이블 배선을 사용했습니다. 그러나 새로운 계산을 위해서 매번 전기배선을 바꿔야 하

는 등의 문제로 천재 수학자인 폰 노이만(John von Neumann)은 현재 컴퓨터의 기본 구조로 자리 잡은 폰 노이만 구조(Von Neumann Architecture)를 개발합니다. 폰 노이만 구조는 모든 프로그램과 데이터를 실행 전에 메모리에 저장하는 내장형 프로그램(stored program) 방식입니다. 폰 노이만이 개발한 EDVAC은 이 구조를 통해서 전기배선 대신에 구멍이 뚫린 카드를 사용하여 컴퓨터와 대화를 합니다.



[그림 2-1] 폰 노이만 구조

ENIAC이나 EDVAC 모두 컴퓨터에 명령을 하려고 전기 배선을 연결하거나 카드에 구멍을 뚫어서 사용한 것은 기계어를 표현하는 방법의 차이일 뿐입니다. 기본적으로 0과 1의 모음으로 이루어진 기계어를 배선이 연결되었는지 연결되지 않았는지를 통해서 0과 1을 구분하거나 카드에 구멍이 뚫려 있는지 아니면 뚫리지 않았는지를 통해서 0과 1을 구분한 것입니다.

```
00004d0 0000 0000 ffff ffff 0000 0000 ffff ffff
00004e0 0000 0000 0000 0000 940c 0804 0000 0000
00004f0 0000 0000 825a 0804 0000 0000 4700 4343
0000500 203a 4728 554e 2029 2e33 2e32 2033 3032
0000510 3330 3530 3230 2820 6552 2064 6148 2074
0000520 694c 756e 2078 2e33 2e32 2d33 3335 0029
0000530 4700 4343 203a 4728 554e 2029 2e33 2e32
0000540 2033 3032 3330 3530 3230 2820 6552 2064
0000550 6148 2074 694c 756e 2078 2e33 2e32 2d33
0000560 3335 0029 4700 4343 203a 4728 554e 2029
0000570 2e33 2e32 2033 3032 3330 3530 3230 2820
```

[그림 2-2] 16진수로 표현된 기계어

초창기 컴퓨터 프로그래머는 너무나 자상하고 인자한 나머지 프로그래머 자신이 컴퓨터가 된 것 마냥 프로그램을 작성했습니다. 그래서 그 당시엔 컴퓨터 프로그래머가 될 수 있는 사람은 극소수밖에 없었습니다. 당시만 해도 컴퓨터를 실제로 본 사람이 전 세계 인구의 0.01%도 안 될 때였으니 어쩌면 당연한 일인지도 모릅니다. 인간이 기계의 입장에서 기계어를 제대로 이해하고 올바르게 프로그램을 작성하는 것은 너무나 어렵고 효율이 떨어지는 일입니다. 그래서 컴퓨터 학자들은 더 나

은 방법을 연구하였고 이에 어셈블리어(Assembly Language)가 탄생했습니다.

어셈블리어는 기계어와 크게 다르지 않습니다. 기존의 숫자로 이루어진(binary) 명령어를 사람이 알아보기 쉽게 영문으로 바꾼 것뿐입니다. 사실 매우 간단한 아이디어지만 이러한 발상의 전환만으로도 프로그램 작성 효율이 크게 증가했습니다. 혹여 일부 독자들은 그것만으로 무슨 효율이 증가하겠느냐고 생각할지 모르겠으나 사람이 숫자로 이루어진 코드를 작성하는 것과 영문으로 된 코드를 작성하는 것은 하늘과 땅의 차이입니다. 영문으로 된 코드를 작성한다는 것은 컴퓨터의 입장에서 프로그램을 작성하던 것을 인간의 입장에서 프로그램하고자 하는 의지가 숨어 있습니다. 그러나 기계어보다는 편리할진 모르지만 여전히 어셈블리어로 프로그램을 작성하기는 쉽지 않습니다.

```
        .file   "a.c"
        .text
.globl main
        .type   main,@function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        movl    $0, %eax
        subl    %eax, %esp
        movl    $0, %eax
        leave
        ret
.Lfe1:
```

[그림 2-3] x86용 어셈블리어

고급 프로그래밍 언어

외국인과 의사소통을 하고 싶을 때 어떻게 하면 될까요? 외국어를 배워서 그 사람과 대화하는 방법이 있습니다. 다른 방법으로는 외국인으로 하여금 내가 말하는 언어를 습득하게끔 하여 나의 모국어로 대화하는 방법입니다. 그리고 두 언어를 모두 할 수 있는 통역사를 통해서 대화하는 방법이 있을 수 있습니다.

컴퓨터와 대화하는 것도 이와 다르지 않습니다. 앞서 기계어와 어셈블리어는 첫 번째 경우처럼 인간이 컴퓨터가 알아듣는 말을 배우는 것과 유사합니다. 그러나 우리가 10년을 배워도 영어에 미숙한 것처럼 기계어와 어셈블리어를 배우는 것은 어려운 일입니다. 그렇다면 컴퓨터를 학습시켜 컴퓨터가 사람의 말을 알아들을 수 있다면 어떨까요? 그야말로 누구나 쉽게 컴퓨터와 대화할 수 있는 시대가 오게 될 것입니다. 그러나 이것은 이상일 뿐 현실은 그렇지 못합니다. 자연 언어 처리(인공지능의 한 분야로 컴퓨터가 학습을 통해서 인간의 언어를 이해할 수 있게 만드는 방법)에 대한 연구가 계속되고 있지만 아직 괄목할 만한 연구성과는 나오고 있지 않습니다. 그래서 우리는 마지막 방

법인 통역사를 통한 의사소통 방법을 사용하고 있습니다.

그런데 통역사를 고용한다고 해서 컴퓨터에 인간의 말을 모두 이해시킬 수 있는 것은 아닙니다. 왜냐하면 통역사가 기계어와 인간의 언어에 능숙한 "사람"이 아니라 인간의 언어에 서툰 "컴퓨터 프로그램"이기 때문입니다. 이 통역사는 항상 확실하고 명확한 언어만을 이해할 수 있습니다. 예를 들어 두루뭉술하거나 어법에 맞지 않는 표현은 올바르게 이해하지 못합니다. 그래서 인간이 한 걸음 양보해서 통역사가 쉽게 알아듣게끔 정확하고 간결한 언어를 개발하였는데 이러한 언어를 고급 프로그래밍 언어라고 합니다.

고급 프로그래밍 언어란 어셈블리어와 같은 언어와 구분 짓기 위해 붙여진 이름으로 어셈블리어처럼 컴퓨터 입장에서 프로그램을 작성하는 언어를 저급 언어, 저급 언어와 달리 보다 인간이 이해하기 쉽게 만들어진 언어 혹은 인간의 언어를 명확하고 간결하게 하여 컴퓨터가 이해할 수 있게 하는 언어를 고급 언어라고 합니다.

고급 프로그래밍 언어로 만들어진 문장은 통역사를 통해서 컴퓨터가 이해할 수 있는 기계어로 바뀝니다. 일반적으로 고급 프로그래밍 언어를 통역해주는 통역사에는 컴파일러(Compiler)와 인터프리터(Interpreter)가 있습니다.

원론적 의미에서 컴파일러는 A라는 언어로 만들어진 코드를 B라는 언어로 변환하는 프로그램입니다. 즉, 어떤 언어든지 다른 언어로 변환할 수 있다면 일종의 컴파일러라고 할 수 있습니다. 컴파일러를 사용하는 대표적인 프로그래밍 언어로는 C, C++, 자바, 델파이 등이 있습니다. 그러나 컴파일러 대부분은 원론적 의미를 떠나서 대게 고급 언어로 이루어진 코드를 어셈블리어로 변환해주는 컴파일러와 어셈블리어를 컴퓨터가 이해할 수 있는 기계어로 바꾸어주는 어셈블러(Assembler)로 구성되어 있습니다. 그래서 C와 같은 고급 언어(최근 C 언어는 저급 언어로 분류되기도 합니다.)로 작성된 프로그램을 실행하려면 컴파일러가 어셈블리어를 생성하고 어셈블러가 생성된 어셈블리어를 컴퓨터가 알 수 있는 명령어로 변환하는 절차를 가지게 됩니다. 특별히 자바는 컴파일러가 어셈블리어로 변환하지 않고 바이트 코드라는 플랫폼에 독립적인 중간 코드를 생성합니다.

반면 Basic, Perl, PHP 그리고 기타 많은 스크립트 언어들은 인터프리터를 사용합니다. 인터프리터는 컴파일러와 다르게 프로그램을 실행할 때마다 한 줄씩 코드를 해석하고 컴퓨터가 알 수 있는 기계어로 변환합니다. 매번 프로그램을 실행할 때마다 코드를 해석하면 컴파일된 프로그램보다 느릴 것이 자명합니다. 최소한 코드를 해석하고 기계어로 변환하는 시간만큼은 더 소요될 테니까요. 그런데 왜 PHP는 인터프리터 방식을 사용하는 것일까요? 여러 가지 이유가 있을 수 있으나 필자가 생각하는 가장 큰 이유는 소스 코드의 유지보수 때문입니다. 웹 프로그램의 특성상 소스 코드는 원격의 컴퓨터(서버)에 존재하기 때문에 이를 수정하려면 소스를 수정한 후 반드시 업로드하는 과정을 거쳐야 합니다. 이때 매번 컴파일하여 프로그램을 업로드하고 웹 브라우저를 통해 실행결과를 확인해보는 것은 여간 번거로운 일이 아닙니다. 그래서 PHP의 경우 소스 코드를 직접 서버에 업로드하여 작업하는데 FTP를 지원하는 에디터 프로그램을 이용하면 마치 내 컴퓨터에 있는 코드를 수

정하는 것처럼 저장할 때마다 바로 반영됩니다. 그러나 앞서 언급하였듯이 이 방식은 컴파일된 프로그램 방식보다 느려서 JSP, ASP.NET과 같은 다른 웹 프로그래밍 언어에서는 인터프리터 방식과 컴파일러 방식을 혼용하는 방법을 사용하여 속도를 높이고 있습니다. 최초로 실행된 경우에는 컴파일 작업을 하고 두 번째 접근부터는 이미 컴파일된 파일을 재사용하는 방식입니다. 만약 소스코드가 수정되었다면 다음번 접속에서는 다시 한번 컴파일 작업이 실행됩니다.

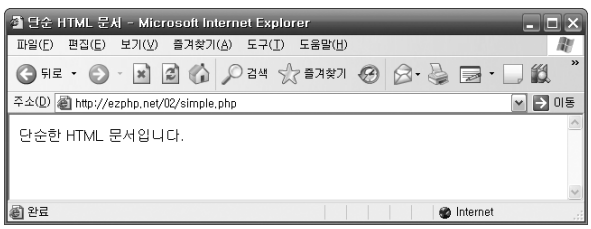
Section 02

PHP 프로그래밍 언어

프로그래밍 언어에 대한 개괄적인 내용을 배웠으니 이제 우리가 정말 배우고자 하는 PHP로 넘어가 봅시다. PHP는 웹 사이트를 보다 편리하게 구축할 수 있게 하려고 만들어진, 서버 측에서 실행되는 스크립트 언어입니다. 정의를 쓰려다 보니 어색하면서 긴 문장이 되었는데 의미는 간단합니다. PHP를 실행하려면 서버가 필요하고, 클라이언트(웹 브라우저)의 요청에 따라 서버가 PHP를 해석하여 실행하고 나서 그 결과를 다시 클라이언트에게 보여주는 것을 의미합니다. 특히 여기서 서버는 웹 서버를 지칭하며 PHP는 웹을 위한 프로그래밍 언어임을 의미합니다. 그래서 PHP는 웹의 표준 문서양식인 HTML을 지원합니다. 그래서 단순한 HTML 문서파일을 PHP 문서파일 확장자인 .php로 저장해도 아무 지장 없이 사용할 수 있습니다.

```
<HTML>
 <HEAD>
  <TITLE>단순 HTML 문서</TITLE>
 </HEAD>
 <BODY>
  단순한 HTML 문서입니다.
 </BODY>
</HTML>
```

위의 소스를 .html이 아닌 .php로 저장하여 웹 서버에 저장하고 웹 브라우저로 실행해보면 다음과 같이 아무 이상 없이 출력됨을 알 수 있습니다.



[그림 2-4] 단순 HTML 실행결과

PHP 문서는 반드시 PHP 코드가 들어가야만 하는 것이 아니라 이처럼 HTML만으로 이루어져 있거나 반대로 HTML 없이 PHP 코드만 존재해도 올바르게 동작합니다.

HTML 모드와 PHP 모드

PHP를 공부하려고 마음먹은 사람이라면 간단한 HTML 정도는 이미 알고 있으리라 생각합니다. 잘은 모르더라도 〈FONT〉, 〈IMG〉, 〈BR〉, 〈TABLE〉 등과 같은 유명한 태그들은 분명히 알고 계실 겁니다. PHP가 웹을 위한 언어이며 웹 브라우저를 통해서 결과를 확인하기 때문에 PHP의 결과는 언제나 HTML과 같은 웹 표준 언어라야 합니다. 왜냐하면 결과를 보여줄 웹 브라우저는 HTML과 같은 웹 표준 언어만을 이해할 수 있기 때문입니다.(물론 자바 스크립트나 VBScript와 같은 스크립트를 제2외국어로 알고 있기는 합니다.) 그래서 PHP 프로그램은 HTML과 PHP 코드가 공존할 수밖에 없습니다. 이러한 이유로 소스 코드 중에서 어느 부분이 HTML 부분인지 또한 어느 부분이 PHP 부분인지 구분할 필요가 있습니다.

PHP와 HTML을 구분하기 위해서 PHP는 두 가지 모드로 동작을 하는데, 바로 HTML 모드와 PHP 모드입니다. 기본적으로는 HTML 모드로 동작하다가 PHP 코드를 만나면 PHP 모드로 전환하여 PHP 부분을 해석하고 실행합니다. 그리고 PHP 코드 부분이 끝나면 다시 HTML 모드로 전환되는 방식입니다. 그러면 간단한 HTML 문서에 PHP 코드를 삽입한, 다음과 같은 PHP 문서를 상상해 봅시다.

```
<HTML>
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
print '여기는 PHP 부분입니다.';
print 라는 단어는 인쇄하다, 출력하다 등의 뜻입니다.
</HTML>
```

위 코드를 보고 어느 부분이 HTML이고 어느 부분이 PHP 부분인지 알아맞춰 보십시오. 아직 문법을 모르기 때문에 쉽게 구분하지 못할 수도 있지만 일단은 문법은 고려하지 말고 한번 구분해 보기 바랍니다.

```
<HTML>
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
print '여기는 PHP 부분입니다.';
print 라는 단어는 인쇄하다, 출력하다 등의 뜻입니다.
</HTML>
```

필자의 의도는 위와 같았습니다. 필자가 이 소스 코드를 작성하면서 2번 줄만 PHP 부분이고 나머지는 단순한 HTML 부분으로 생각했습니다. 아마도 많은 분이 어렵지 않게 잘 구분했을 것으로 생각합니다. 그 이유는 '여기는 PHP 부분입니다.'라는 말이 있기 때문입니다. 여러분은 이 말을 이해할 수 있었기 때문에 쉽게 해당 부분이 PHP 부분일 것으로 예상할 수 있었습니다. 그러나 여러분이 정답을 맞힐 수 있었던 것은 사실 우연일 뿐입니다. 왜냐구요? 그렇다면 다음의 소스 코드를 다시 한번 HTML과 PHP로 구분해 보십시오.

```
<HTML>
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
print 'PHP 예제';
<BR>
print '안녕하세요. PHP 세상에 오신 것을 환영합니다.';
</HTML>
```

많은 분이 위의 3, 5번 줄을 PHP 구문으로 생각하였을 것입니다. 왜냐하면 앞선 문제 때문에 대충 PHP의 모양을 눈치 챌 수 있었기 때문입니다. 그러나 아쉽게도 오답입니다. 왜냐하면 필자가 소스 코드를 작성할 때는 2번 줄만 PHP 코드이고 나머지는 전부 HTML로 생각했기 때문입니다.

필자는 다음과 같은 출력결과를 예상하고 프로그램을 작성했습니다.

```
PHP 예제
print '안녕하세요. PHP 세상에 오신 것을 환영합니다.';
```

만약 3, 5번 줄이 PHP 구문이었다면 다음과 같은 결과를 출력했을 것입니다.

```
PHP 예제
안녕하세요. PHP 세상에 오신 것을 환영합니다.
```

여러분 중 일부는 정답을 맞혔을 것입니다. 그러나 정답을 맞혔다고 좋아할 필요는 없습니다. 물론 틀렸다고 슬퍼할 이유도 없습니다. 여기에 제시된 소스 코드는 여러 가지 의미로 오해할 수 있도록 애매하게 작성했기 때문입니다. 프로그래머의 생각을 미루어 짐작하는 수밖에 없는 이러한 코드는 HTML 부분과 PHP 부분을 쉽게 판단할 수 없게 만듭니다.

컴퓨터는 정확하게 구분할 수 있을까?

천만의 말씀입니다. 컴퓨터도 여러분과 마찬가지로 소스 코드를 보고서 PHP 부분과 HTML 부분을 구분할 수 없습니다. 컴퓨터는 생각만큼 똑똑하지 못하기 때문입니다. 대부분은 사람이 판단하지 못하는 문제를 컴퓨터도 판단할 수 없습니다. 심지어는 사람이 쉽게 판단하는 문제도 컴퓨터는 절대 판단할 수 없는 경우도 많습니다. 물론 사람에게는 어렵지만 컴퓨터는 쉽게 판단하는 때도 많이 있습니다. 그러나 이것은 대부분 컴퓨터의 빠른 계산 능력 때문에 짧은 시간이 걸린 것일 뿐이지 사람이 풀 수 없는 문제를 컴퓨터가 해결하는 것은 아닙니다. 왜냐하면 컴퓨터가 판단한다는 것은 사람의 두뇌로 작성한 프로그램에 의해서 가능한 것이기 때문이다. 그래서 앞의 문제와 같이 애매모호한 문장은 사람과 컴퓨터 모두 올바르게 해석할 수 없습니다. 따라서 컴퓨터 프로그램은 항상 어느 누가 보더라도 똑같이 이해할 수 있어야 합니다. 같은 소스 코드를 보고 여러 사람이 다르게 생각한다면 그것은 잘못된 코드입니다.

PHP는 어느 누가 보더라도 HTML 부분과 PHP 부분을 쉽게 구분할 수 있도록 다음과 같은 4종류의 방법을 제공하고 있습니다.

순위	시작 태그	끝 태그	비고
1	〈?	?〉	일반적인 방법
2	〈?php	?〉	XML과 구분하기 위한 방법
3	〈script language="php"〉	〈/script〉	스크립트식 방법
4	〈%	%〉	ASP식 방법

[표 2-1] HTML 모드에서 PHP 모드로 빠져나오는 방법

[표 2-1]은 필자가 판단하기에 자주 사용되는 방식을 순서대로 나열한 것입니다. 그런데 HTML 모드와 PHP 모드가 있다고 했는데 표 제목은 "HTML 모드에서 PHP 모드로 빠져나오는 방법"이라고 이름을 지었습니다. 왜 빠져나오는 방법이라고 했을까요?

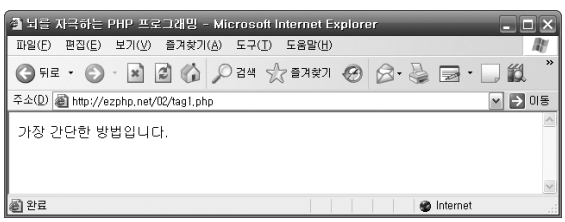
눈치가 빠른 분들이라면 이미 알아채셨을 거로 생각합니다. 빠져나오는 방법이라고 한 이유는 앞서 언급했듯이 PHP 문서가 기본적으로 HTML 모드로 동작하기 때문입니다. 아마도 기본 모드를 무엇으로 정할지 PHP를 개발한 분들이 많이 고민했으리라 생각합니다. 결론적으로 HTML 손을 들어주었는데, 그 이유는 이미 웹에 널리 퍼져 있는 기존의 HTML 문서를 쉽게 PHP 문서로 바꿀 수 있게 하기 위해서가 아닐까 생각해 봅니다. 기존의 HTML 문서에 특정 기능을 추가하고 싶다면 새로 문서를 작성하지 않고 기존 문서에 간단히 PHP 코드를 삽입하기만 해도 올바르게 동작하기 때문입니다.

PHP는 [표2-1]과 같은 PHP 시작 태그들이 나오면 "앗! 이제 내 차례네~" 하고 열심히 PHP 코드 부분을 해석하고 처리합니다. 그런 다음 모든 처리가 끝나고 PHP의 끝을 알리는 태그가 나오면 PHP는 다시 HTML 모드로 전환하고, 다음 시작 태그가 나올 때까지 휴식을 취합니다.

[그림 2-5] HTML 모드 vs. PHP 모드

그럼 각 태그별로 실제 어떻게 사용하는지 알아봅시다.

```
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
<?
    echo '가장 간단한 방법입니다.';
?>
```



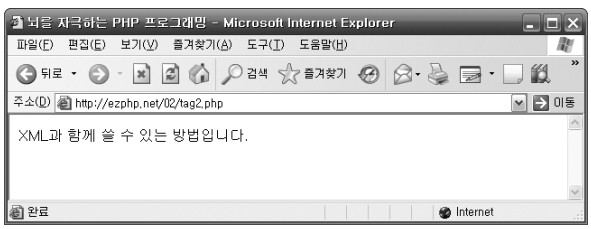
[그림 2-6] 〈? ?〉 태그의 사용결과

이 방법은 가장 간단한 방법이어서 가장 많은 사람이 애용하는 방식입니다. 그러나 이 방법은 XML(eXtensible Markup Language)이라는 녀석이 〈?xml 모양의 태그를 사용하기 때문에 XML과 PHP를 같이 사용하는 경우에는 사용할 수가 없습니다. 왜냐하면 〈?xml 태그를 해석할 때 PHP는 '〈?' 모양을 보고 바로 PHP 부분이라고 생각할 테지만 실제로는 PHP 부분이 아니라 XML 부분이기 때문입니다. 앞서 말씀드렸듯이 컴퓨터 프로그램은 모호한 부분을 가지고 있어서는 안 됩니다. 그래서 많은 책과 많은 선배 프로그래머들이 이 태그의 사용을 권하지 않습니다. 그러나 XML과 동시에 사용하지 않는다면 굳이 편리한 이 방법을 쓰지 않을 이유가 없습니다. 앞으로 등장하는 코드 대부분은 이 방법을 사용합니다.

```
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
<?php
    echo 'XML과 함께 쓸 수 있는 방법입니다.';
?>
```

여기서 잠깐

php.ini 파일에 short_open_tag라는 설정 부분이 있습니다. 기본적으로 이 설정은 On으로 되어 있지만 서버 관리자나 설치 프로그램 등에 의해 Off로 되어 있는 경우가 가끔 있습니다. 이런 때에는 이 방식을 사용할 수 없으므로 만약 이 태그가 적용되지 않는다면 php.ini 파일을 수정합니다.

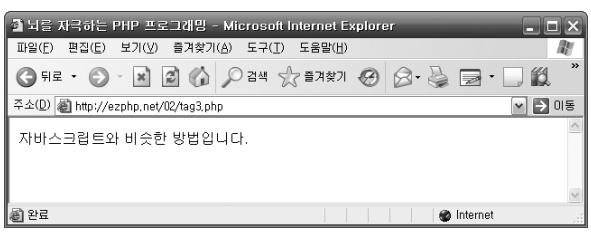


[그림 2-7] 〈?php ?〉 태그의 사용결과

이 방법은 XML과의 충돌방지를 고려한 방법입니다. 첫 번째 방법에 php를 덧붙여서 XML을 비롯한 비슷한 방식의 문서 태그와 구분할 수 있게 만든 것입니다. 이 방법 또한 많은 프로그래머가 애용하고 있으나 역시 단순한 게 최고인지라 첫 번째 방식이 더 많이 쓰입니다. 하지만 XML과 함께 사용할 때는 꼭 이 방법을 써야 함을 기억하기 바랍니다.

첫 번째와 두 번째 방법을 제외한 나머지 방법들은 실제로는 거의 사용하지 않는 방식입니다. 따라서 가볍게 '이런 것이 있구나!'하고 넘어가세요.

```
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
<script language="php">
    echo '자바스크립트와 비슷한 방법입니다.';
</script>
```



[그림 2-8] 스크립트 형식 태그의 사용결과

이 방법은 자바스크립트나 VBScript를 HTML에 추가하고자 할 때 사용하는 방식과 유사합니다. 자바스크립트 등을 많이 사용하는 사람에게는 익숙한 모양일지 모르지만 대부분의 경우 너무 번거로워서 실제로는 사용되지 않습니다. 그렇다면 쓸데없이 왜 이런 것을 지원할까요? 그 이유는 웹 페이지 제작에 많이 사용하는 웹 에디터 때문입니다. 위지윅(WISIWIG, what you see is what you get) 방식의 웹 에디터 중에서 PHP를 지원하지 않는 경우가 있습니다. 최신 버전의 웹 에디터들은 대부분 PHP를 지원해서 PHP 코드를 올바르게 받아들이지만 오래된 버전이나 PHP를 여전히 지원하지 않는 다른 웹 에디터들은 〈? 나 〈?php 같은 태그를 만나면 잘못된 태그로 인식하고 해당 PHP 코드를 지워버리는 경우가 생기는 등 문제가 발생할 가능성이 있습니다. 이 경우 이 방법을 사용하면 PHP를 지원하지 않는 웹 에디터에서도 PHP 코드를 자바스크립트처럼 인식하기 때문에 코드에 아무런 영향을 주지 않습니다. 하지만 근래의 웹 에디터는 대부분 PHP를 지원하므로 이 방

법을 권하지 않습니다.

```
<TITLE>뇌를 자극하는 PHP 프로그래밍</TITLE>
<%
    echo 'ASP식 방법입니다.';
%>
```

이 태그는 마이크로소프트 사가 개발한 서버 측 스크립트인 ASP(Active Server Page)식 방식을 그대로 빌려 만든 방법입니다. 이는 ASP 프로그래머로 하여금 코드 작성에 편의를 주고자 만들어진 것으로 필자가 판단하기에는 기존 웹 프로그래밍 시장을 선점하고 있던 ASP 프로그래머를 PHP로 옮겨 타게 하기 위한 물밑 작업의 하나라 생각합니다. 그러나 ASP 프로그래머에게는 친숙할지 모르나 PHP 프로그래머라면 '누가 이 방법을 쓰겠나?'하는 방법이 아닐까 합니다. 또한 이 방법을 사용하려면 특별히 따로 설정을 해주어야만 합니다. 일반적인 방법으로 설치된 경우에는 다음과 같은 결과가 나옵니다.

[그림 2-9] ASP식 방법이 적용되지 않는 경우

이런 경우 php.ini 파일에서 asp_tags = Off를 asp_tags = On으로 바꾸어 주고 아파치 웹 서버를 다시 시작하면 다음과 같은 결과를 얻을 수 있습니다.

[그림 2-10] ASP식 방법이 적용된 경우

잠시 쉬어가는 의미에서 간단한 퀴즈를 맞혀봅시다. [그림 2-9]를 보면 asp_tags = Off인 경우 〈%와 %〉 태그를 사용했을 때, 문서의 제목은 변경되었는데 본문 부분에는 아무것도 출력되지 않았습니다. 〈%와 %〉 사이에 있던 부분은 왜 출력되지 않았을까요?

이유는 〈% ~~~~ %〉 여기까지가 HTML 태그라고 웹 브라우저가 생각했기 때문입니다. PHP는 〈% 태그를 허용하지 않았기 때문에 HTML 부분이라고 생각하고 HTML로 동작하게 하였고 웹 브

라우저는 <를 보고 '여기부터 다음 >까지가 HTML 태그구나!' 하고 생각합니다. 그런데 태그라고 생각했던 것이 유효하지 않은 태그이므로 웹 브라우저는 잘못된 태그로 판단하고 무시해버려서 아무것도 출력이 안된 것입니다.

우리는 이제 HTML 모드와 PHP 모드를 자유자재로 넘나들 수 있게 되었습니다. HTML 모드에서 해야 할 일들은 이 책을 읽기 전에 미리 공부하였을 거라 믿고(만약에 그렇지 못하다면 HTML 태그 부분을 반드시 공부하기 바랍니다.) 이제부터 본격적으로 PHP라는 언어를 배워보도록 합시다.

PHP에서 문장 구분하기

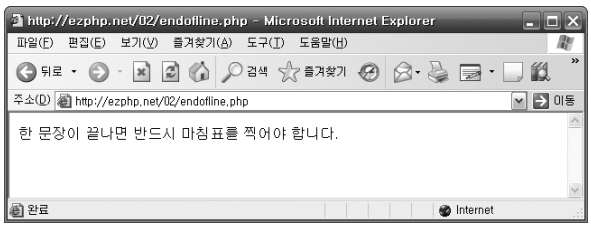
우리가 글을 쓸 때 문장의 마지막에 마침표(.)를 찍듯이, PHP도 한 문장이 끝날 때 마침표를 찍어야 합니다. PHP에게 "이게 한 문장이야~"라고 친절히 알려주고자 하는 것입니다. 일반적으로 글을 쓸 때 마침표를 찍지 않아도 사람들은 충분히 알아볼 수 있지만 PHP는 그렇지 못합니다. 만약 있어야 할 곳에 마침표가 없으면 PHP는 마침표가 있어야 하는 자리라는걸 눈치 채지 못하고 마침표가 나올 때까지 기다리고 비록 여러 문장이 지나갔더라도 다음번 마침표가 나와야 지금까지를 한 문장이라고 해석을 합니다. 실제로는 여러 문장이지만 마침표가 없어서 한 문장이라고 생각하는 것입니다.

다른 많은 프로그래밍 언어에서도 PHP처럼 문장이 끝나는 것을 명시적으로 표현해 주기를 바랍니다. 이는 코드의 가독성과 명확성을 높여주기 위한 것으로 마침표를 통해서 컴파일러나 인터프리터는 문맥을 파악하지 않고도 쉽게 문장을 구분할 수 있습니다. 그러나 마침표를 잊어버리는 경우가 많아서 간혹 PHP는 프로그래머가 의도한 바와 다르게 PHP 문서를 해석하는 경우가 생깁니다. BASIC과 같은 몇몇 프로그래밍 언어들은 마침표가 존재하지 않습니다. 이러한 언어들은 문장을 구분하기 위해 줄 구분을 합니다. 즉, 엔터를 통해서 구분되면 새로운 문장이라고 판단하는 것입니다. 이 방법은 귀찮게 마침표를 찍어주지 않아도 되므로 편리하지만 한 줄에 여러 개의 짧은 문장을 쓰는 등의 일을 할 수 없고 긴 문장을 한 줄에 모두 표시해야 하므로 때로는 가로로 매우 긴 프로그램 코드가 생길 수 있습니다.

PHP는 명시적으로 마침표를 찍어야 합니다. 그렇다면 PHP에서 마침표는 무엇일까요? 말 그대로 마침표(.)일까요? 아닙니다. PHP 언어에서 한 문장이 끝남을 알리고 문장과 문장을 구분 짓는 기호는 세미콜론(;)입니다. 이 세미콜론은 PHP뿐만이 아니라 C 언어와 자바 등과 같은 많은 유명 언어들에서도 사용되고 있습니다.

다음은 올바른 문장의 사용법입니다. 마침표를 잘 찍었는지 주의하면서 살펴봅시다.

```
<?
    echo '한 문장이 끝나면 반드시 마침표를 찍어야 합니다.';
?>
```

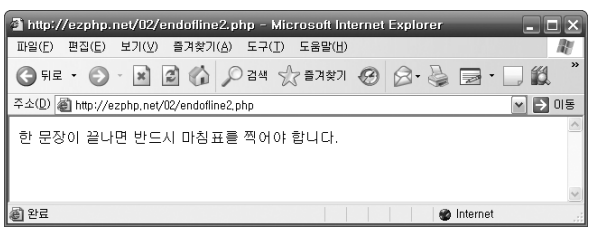


[그림 2-11] 문장의 마침표 세미콜론(;)

그렇다면 만약에 마침표(;)를 찍지 않으면 어떻게 될까요? 한번 다음과 같이 실험을 해봅시다.

```
<?
    echo '한 문장이 끝나면 반드시 마침표를 찍어야 합니다.'
?>
```

위의 예제에서 두 번째 줄 마지막의 세미콜론을 제거해 보겠습니다. 마침표를 찍지 않았으니 아마도 에러가 날 것입니다.



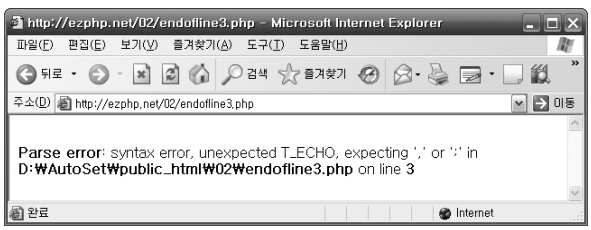
[그림 2-12] 세미콜론을 잊은 경우

앗! 뭔가 이상하다!

분명히 마침표를 찍지 않았음에도 두 예제의 결과가 같게 나왔습니다. 마침표를 찍어야 한다고 그렇게 말씀드렸는데 말입니다. 이거 완전 거짓말쟁이가 되어 버렸군요. 마침표를 찍지 않았음에도 올바른 결과가 출력된 이유는 문장 뒤에 PHP 모드의 끝을 알리는 태그인 ?〉가 있어서 입니다. PHP는 여러 문장이 있을 때 마침표가 없으면 문장과 문장을 서로 구분할 수 없지만 문장이 하나뿐이거나 마지막 문장(?〉 태그 바로 앞의 문장)의 경우에는 PHP 모드의 끝을 알리는 태그를 통해서 문장이 끝난 것을 알 수 있습니다. 그러나 이것은 PHP가 프로그래머의 실수를 감춰주는 것일 뿐 올바른 문법이 아닙니다. 마지막 문장에는 마침표를 찍지 않아도 된다는 것 때문에 위와 같이 세미콜론을 적어두지 않았다가 다음에 코드를 덧붙이거나 하는 경우 에러를 발생할 여지를 만들게 됩니다.

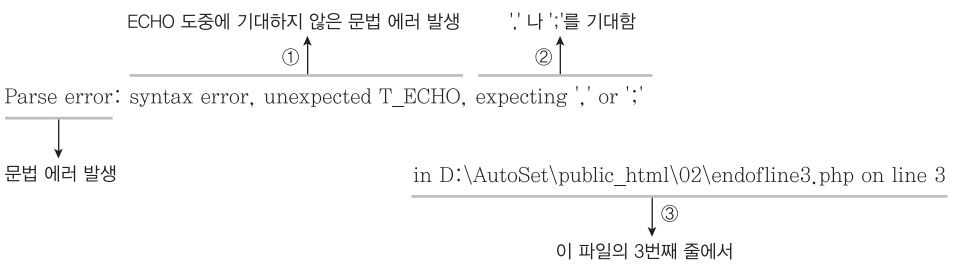
```
<?
    echo '한 문장이 끝나면 반드시 마침표를 찍어야 합니다.'
    echo '만약 마침표를 찍지 않으면 에러가 발생합니다.'
?>
```

앞의 예제에 위와 같이 한 문장을 덧붙여 보겠습니다. 이번에도 마침표를 찍지 않고 문장을 추가해 보았습니다. 이번에는 분명히 에러가 날 것입니다. 만약에 이번에도 에러가 발생하지 않는다면 필자는 책 장사 접어야 할 것 같네요.



[그림 2-13] 여러 문장이 있을 때 마침표를 잊은 경우

우리가 우려한 대로 마침표를 찍지 않아서 위와 같이 에러가 발생했습니다. 영어로 된 에러 메시지라고 너무 겁먹을 필요 없습니다. 에러 메시지는 프로그래머에게 "너 프로그래밍을 어떻게 하는 거야?", "너! 그렇게밖에 못해?"라고 말하는 것이 아닙니다. 오히려 친절하고 인자한 빨간 펜 선생님입니다. 비록 우리말로 자세히 말해주지는 못하지만 PHP의 친절한 마음을 받아들여 자세히 알아보도록 합시다.



[그림 2-14] 에러 메시지의 분석

Parse error는 이 문장이 PHP 문법에 맞지 않다고 알려주는 에러입니다. 즉, 어법에 맞지 않게 글을 썼다고 알려주는 것입니다. 이는 문법에만 맞추면 금방 해결되는 문제이므로 에러 중에서 가장 고치기 쉬운 에러라고 할 수 있습니다.

① syntax error, unexpected T_ECHO

echo 구조를 이용하여 출력을 하는 도중 예기치 않게 문법 에러가 발생했다는 뜻입니다.

② expecting ',' or ';'

콤마(,)나 세미콜론(;)이 나올 줄 알았는데 나오지 않았다는군요.

③ in D\AutoSet\public_html\02\endofline3.php on line 3

해당 파일의 3번째 줄에서 에러가 발생하였다는 메시지입니다.

에러 메시지를 정리해보면 다음과 같습니다.

해당 파일의 3번째 줄에서 echo 출력을 하던 도중 예기치 않은 문법 에러가 발생했습니다. 콤마(,)나 세미콜론(;)이 나오길 기대하고 있습니다.

에러 메시지를 보았으니 왜 에러가 발생하였는지 이유를 알아봅시다. 우리가 이미 알고 있듯이 에러는 문장의 끝에 세미콜론을 적어두지 않아서입니다. 좀 더 정확하게 말씀드리면 3번째 줄에 세미콜론이 없기 때문입니다.

그렇다면 왜 2번째 줄에서 에러가 발생하지 않고 3번째 줄에서 에러가 발생했을까요? PHP는 세미콜론을 기준으로 문장을 판단하면서 2번째 줄과 3번째 줄을 하나의 문장으로 판단합니다. 그런데 2번째 줄까지는 문법상 문제없이 처리되었고 이제 세미콜론만 나오면 되는데 2번째 줄에는 세미콜론이 없어서 3번째 줄에서 세미콜론을 찾습니다. 그런데 세미콜론이 나와야 할 자리에 다시 echo 문이 나오니까 3번째 줄에 echo 구조가 아닌 세미콜론이 나와야 한다고 알려주는 것입니다.

2번 줄과 3번 줄은 줄이 다르잖아요? 어떻게 한 문장인가요?

우리는 글을 쓸 때 한 줄을 다 채우지 않고 다음 줄로 줄 바꿈이 되면 대개 새로운 문장이라고 생각합니다. 시와 같이 문법적 예외를 두는 경우를 제외하곤 말이죠. 그러나 PHP는 줄 바꿈이 되어 있어도 늘 같은 줄이라고 생각합니다. 왜냐하면 앞서 말씀드렸듯이 세미콜론이 한 문장이 끝났음을 나타내는 기호이므로 세미콜론이 나오기 전까지는 다른 문장이 아니라고 생각하는 것입니다. 심지어 문장의 중간에 100줄의 공백이 있어도 PHP는 하나의 문장으로 판단합니다. 즉, PHP에서 줄 바꿈과 공백은 단어의 구분 이외에는 아무런 의미가 없습니다. 역으로 생각하면 세미콜론으로 연결해서 한 줄로 이루어진 굉장한 프로그램을 작성할 수도 있습니다.

```
<?
   echo '문장에서
   들여쓰기와
   줄바꿈은
   아무런 의미가 없습니다.';
?>
```

이 예제를 보면 무언가 잘못되어서 에러를 발생시킬 것 같은 생각이 듭니다. 하지만 실제로는 다음과 같은 올바른 결과를 출력합니다.

[그림 2-15] 들여쓰기와 줄 바꿈의 의미

그런데 다른 많은 프로그래머가 작성한 소스 코드를 보면 항상 들여쓰기를 비롯하여 줄 바꿈이 되어 있는 것을 알 수 있습니다. PHP는 의미 없게 생각하지만 프로그래머가 보기 편하게 줄 바꿈과 들여쓰기를 하는 것입니다. 이렇게 해두면 소스 코드를 작성하거나 나중에 수정하게 될 때 훨씬 효율적이면서 편하게 프로그램을 작성할 수 있습니다. 이제 에러의 원인도 알아냈으니 올바르게 코드를 수정해 봅시다.

```
<?
    echo '한 문장이 끝나면 반드시 마침표를 찍어야 합니다.';
    echo '만약 마침표를 찍지 않으면 에러가 발생합니다.';
?>
```

비록 2번째 줄에는 세미콜론을 붙이지 않아도 문법적으로 문제는 없지만 모든 문장의 끝에 마침표를 찍는 습관을 길러 두시기 바랍니다. 결과는 다음과 같습니다.

[그림 2-16] 마침표를 모두 적용한 결과

PHP에서 주석 달기

주석이라는 것은 코드에 덧붙여 놓은 해당 코드에 대한 설명입니다. 프로그래머가 소스 코드를 작성하면서 소스 유지 및 관리를 위해서 추가해 놓는 것으로 단지 코드에 대한 설명이기 때문에 PHP는 굳이 힘들여 이 부분을 해석하려 하지 않습니다. 그래서 주석 속에 소스 코드를 작성해 놓는다 하더라도 PHP는 주석이라 생각하여 그 코드를 무시합니다. PHP는 기본적으로 C와 Perl과 같은 형식의 주석을 제공합니다.

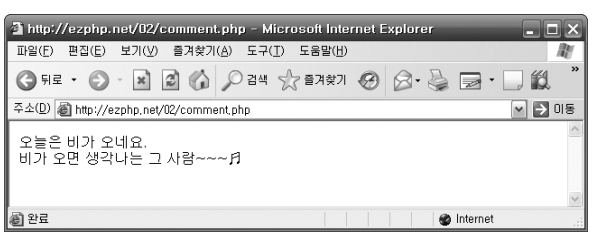
주석 방법	설명
//	한 줄 주석 처리
#	한 줄 주석 처리
/* */	여러 줄 주석 처리

[표2-2] PHP의 주석

PHP는 표에서 보이듯이 한 줄 주석과 여러 줄 주석을 제공하고 있습니다. 여기서 주의할 점은 한 줄의 의미가 세미콜론으로 구분되는 한 문장을 뜻하는 것이 아니라 말 그대로 해당 줄을 의미합니다. 한 줄 주석에는 두 가지 방법이 있으며 일반적으로는 **//**를 사용하는데 주석 기호 이후 줄 바꿈이 되기 전까지의 모든 문자를 주석 처리합니다. 만약에 두 줄 이상의 다량의 코드에 대해 주석 처리를 하고 싶다면 다중 주석 기호인 **/* */**를 이용하여 주석 처리를 할 수 있습니다. 다중 주석 기호는 PHP 시작과 끝 태그와 마찬가지로 주석 시작 기호(**/***)와 주석 종료 기호(***/**)로 쌍을 이루고 있습니다.

실제로 주석을 어떻게 사용하는지 한번 확인해 봅시다.

```
<?
  //2008년 2월 1일
  echo '오늘은 비가 오네요.<BR>';
  echo '비가 오면 생각나는 그 사람~~~♬<BR>'; #노래에서 연륜이.
  /* 여러 줄 주석을 하면
     줄마다 한 줄 주석을 달 필요가 없어요. */
?>
```



[그림 2-17] 주석의 예

결과를 보시면 아시겠지만 주석 처리된 부분은 결과에 반영되지 않습니다. 이처럼 주석을 다는 이유는 코드에 대해 설명해놓기 위해서입니다. 한 번 만에 프로그램을 짜고 '다시는 이 소스 코드를 보지 않으리라!'라고 생각하거나 극명하게 간단한 코드여서 "딱 보면 압니다~"라는 경우에는 주석이 필요 없습니다. 그러나 나중에 소스 코드를 재사용하거나 수정을 하려면 주석이 매우 큰 도움을 줍니다. 주석이 없으면 코드를 다시 볼 때 이해하기 어려울 뿐만 아니라 그 당시 무슨 생각을 하고 이 코드를 작성했는지 알 수 없어서 프로그램을 이해하는데 많은 어려움이 따릅니다.

주석은 PHP 문서를 해석하는데 아무런 영향을 끼치지 않기에 주석이 아무리 길더라도 성능상의 차

이는 거의 없습니다. 그러니 코드의 기능이 무엇인지 어떤 생각으로 코드를 작성하는지에 대해 자세하게 주석을 다는 버릇을 들이는 것이 매우 중요합니다.

```
<?
 /*
      프로그램명 : Hello World 출력 프로그램
      작 성 자 : 조명진
      작 성 일 : 2008년 2월 1일
 */
      echo 'Hello World';
?>
```

대체로 프로그램을 작성하면서 파일의 제일 위에는 위와 같은 주석을 많이 달아 놓습니다. 이러한 주석 덕분에 해당 파일이 어떤 기능을 하는지, 누가 작성했는지, 언제 작성되었고 언제 수정되었는지에 대한 정보를 알 수 있어서 코드를 유지하고 보수하는 데 많은 도움이 됩니다.

다음과 같이 주석을 처리할 때 결과가 어떻게 될까요?

```
/* 여러 줄 주석 안에  /* 또 여러 줄 주석이 있으면 */ 어떻게 될까요? */
```

정답은 에러가 발생한다! 입니다. 여러 줄 주석 안에 한 줄 주석을 넣는 것은 전혀 문제가 없습니다. 문제는 위처럼 여러 줄 주석에 또 여러 줄 주석이 있는 경우인데요, 이럴 때

```
/* 여러 줄 주석 안에  /* 또 여러 줄 주석이 있으면 */
```

까지가 주석 처리가 되고 나머지

```
어떻게 될까요? */
```

는 주석이 아닌 PHP 구문이라고 생각합니다. 그 이유는 PHP가 **/*** 기호를 만나면 지금부터 여러 줄 주석이라고 생각하고 다음번 ***/** 기호가 나올 때까지를 주석이라고 생각하기 때문입니다. 그래서 중간에 여러 줄 주석이 또 있다고 생각지 않고 **있으면**에서 주석이 끝나고 나머지는 PHP 구문이라고 생각합니다. 이때 PHP 구문이라고 생각한 **어떻게 될까요? */** 부분이 PHP 문법상 맞지 않으므로 에러가 발생하는 것입니다.

식별자(identifier)

이번에는 우리 모두 작명가가 되는 시간을 갖도록 하겠습니다. PHP에서는 변수니 함수니 하는 것들이 있는데 이들은 모두 각자 구별이 가능한 이름을 가지고 있습니다. 이들의 이름은 어떠한 규칙에 따라 짓는데 이렇게 만들어진 이름을 식별자라고 합니다. 식별자는 식별자라는 말에서도 알 수 있듯이 다른 무엇과 서로 식별을 할 수 있게 해주는 이름입니다.

새로운 이름을 만들 때 가장 많이 고민하는 것은

① 예쁜가?(멋진가?)

② 이름의 뜻은 좋은가?

③ 발음이 이상하거나 어렵지 않은가?

등일 것입니다.

그러면 이름을 하나 떠올려 봅시다. 방금 인터넷을 통해 여러 이름을 찾아봤더니 "부시 전 미국 대통령"의 딸인 "Barbara Bush"가 검색되었습니다. "Barbara"라 예쁜 이름인 것 같군요. 그런데 제가 영어를 잘 못해서 그런지 몰라도 "Barbara"가 원어 발음으로 듣기엔 "밟아라 부시"처럼 들리네요. 딸이 아버지를…. 이거 부시 대통령이 딸 이름은 잘못 지은 것 같습니다.

우리나라의 예를 들어보면 나라를 다스리는 큰 인물이 되라는 뜻에서 치국(治國)이란 근사한 이름을 지어줬는데 성이 김(金)인 경우와 비슷합니다. 이름이 김치국이라… 실제로 이런 이름을 가지신 탤런트도 계신다죠.

이처럼 이름을 지을 때는 한 가지만 생각해서는 안 되고 여러 가지 조건에 맞는지 잘 생각해야 한다는 것을 알 수 있습니다. 이와 마찬가지로 프로그래밍할 때도 이름을 지을 때 여러 가지 조건을 생각하며 이름을 지어야 합니다. 그렇다면 사람 이름을 짓는 것만큼이나 이름 짓기가 어렵다는 말인데, 보통 사람은 평생 이름을 지어봐야 자식 손자 합쳐서 서넛을 넘지 않을 텐데 프로그래머는 하루에도 10개 20개를 생각해 내야 하니 한숨부터 나옵니다. 하지만 프로그램에서의 이름은 그 역할을 미리 아는 경우가 대부분입니다. 마치 영화 〈늑대와 춤을〉에서 '주먹쥐고 일어서'처럼 이미 그 사람의 특징을 잘 알고 있다면 이름을 짓기가 훨씬 쉬워질 것입니다.

사람의 이름을 지을 때와 마찬가지로 식별자를 만들 때 가장 많이 고민하는 것은 다음과 같습니다.

① 역할에 맞는 이름인가?

② 중복된 이름은 없는가?

③ 식별자 규칙을 따르는가?

역할에 맞는 이름인가?

첫 번째로 식별자를 만들 때는 '이름이 충분히 그 역할을 대변하고 있는가?'를 고려해야 합니다. 이 경우의 가장 일반적인 예가 sum이나 count 같은 것입니다. 덧셈의 결과를 저장하게 될 무언가가 있다면(뒤에서 배우게 될 변수) 그 이름을 sum이라고 지을 수 있겠고 만약 숫자를 하나 둘 셋 이렇게 세어 나가는 변수가 있다면 이 또한 이름을 count라고 지으면 좋을 것입니다. 누구나가 쉽게 '이 sum은 덧셈의 결과를 저장하는 변수겠구나!'하고 생각할 수 있을 것입니다. 이러한 이유로 식별자를 결정할 때 가장 중요한 것이 바로 "역할을 충분히 표현할 수 있는 이름으로 정하라!"입니다.

중복된 이름은 없는가?

이름을 짓는 이유와 마찬가지로 프로그램 내에서도 서로 다른 역할을 가진 것들을 구분 지으려고 식별자를 만듭니다. 그래서 같은 이름을 갖는 또 다른 식별자가 있다면 그 둘을 구분 짓지 못하기 때문에 오류를 일으킬 여지가 있습니다.

식별자 규칙을 따르는가?

우리의 이름에 **+철수, 김-수, 김철***와 같이 특수문자를 포함하는 이름을 지을 수 없듯이 식별자에도 몇 가지 규칙이 있습니다.

① **문자와 밑줄(_)로 시작한다.**
우리가 이름을 지을 때 성에 내가 원하는 아무 글자나 쓸 수 없듯이 식별자의 첫 번째 글자는 반드시 문자나 밑줄로 시작해야 합니다. 즉, 숫자를 쓰거나 밑줄이 아닌 다른 특수기호는 첫 번째 글자를 쓸 수 없다는 뜻입니다.

② **두 번째 글자부터는 숫자, 문자, 밑줄이 가능하다.**
식별자의 두 번째 글자부터는 첫 번째에 비해서 자유롭습니다. 문자나 밑줄뿐 아니라 숫자까지 사용할 수 있습니다.

③ **밑줄 이외의 기호는 사용할 수 없다.**
두 번째 글자부터는 숫자, 문자, 밑줄이 가능합니다. 그러나 그 외의 기호는 사용할 수 없습니다. 또한 식별자 중간에 공백이 있어서도 안 됩니다.

④ **문자는 유니코드(한글포함)도 가능하다.**
다른 프로그래밍 언어와 다르게 PHP는 국제 문자 부호 체계인 유니코드를 지원합니다. 즉, 영문 알파벳 뿐 아니라 한글이나 히라가나 등으로 식별자를 만들어도 상관없다는 뜻입니다. 간혹 재미삼아 한글로 된 프로그램을 작성하기도 하는데 될 수 있으면 식별자는 영문 알파벳을 사용하는 것이 바람직합니다.

⑤ **변수는 대소문자를 서로 구분된다.**
PHP는 기본적으로 대소문자를 구분하지 않습니다. PHP 키워드나 함수 등에서는 대소문자를 섞어서 사용하여도 같은 것으로 간주합니다. 예를 들면 brown이란 이름의 식별자와 BroWn이란 이름의 식별자는 동일한 것으로 간주합니다. 그러나 만약 이러한 식별자가 다음에 배울 변수에 사용된다면 대소문자를 구분하여 다른 것으로 간주합니다.

변수

일반적으로 변수란 여러 가지 값을 가질 수 있는 즉, 여러 가지 값으로 변할 수 있는 수를 의미합니

다. 프로그래밍에서의 변수도 이와 의미상 크게 다르지 않습니다. PHP에서 변수란 여러 가지 값을 기록해 둘 수 있는 메모리 공간에 이름을 붙여 둔 것을 의미합니다.

변수를 쉽게 이해하려면 겉에 이름이 적힌 복주머니를 떠올리면 됩니다. 설날에 세배하고 받은 돈을 통장이나 책상 깊숙이 숨겨 두기 전까지는 대개 복주머니에 넣어 두고 보관하곤 합니다. 세배를 할 때마다 복주머니에는 더 많은 돈이 들어가고 그 돈으로 눈여겨 두었던 물건을 사고 나면 복주머니 속의 돈은 다시 줄어들게 됩니다. 이처럼 주머니 속에 있는 돈이 늘어났다 줄어들었다 하듯이 변수의 값도 다양하게 변할 수 있습니다. 즉, 복주머니 겉에 쓰인 이름은 변수의 이름을 뜻하고 복주머니 속에 들어 있는 것이 변수의 값을 의미합니다.

PHP에서 변수를 **$name**과 같은 모양으로 표시합니다. 변수임을 표시하는 기호 $와 변수의 이름이 되는 식별자를 붙여 둔 형식입니다. 즉, **$name**은 name이라는 이름의 변수임을 뜻합니다. 따라서 변수를 사용할 때는 반드시 앞에 $ 기호를 붙여야 합니다.

변수는 주머니와 같이 여러 가지 것들을 담을 수 있습니다. 변수에 담을 수 있는 것에는 진리값, 정수, 부동소수점수, 문자열, 배열, 객체, 리소스, 널과 같이 총 8가지의 종류가 있습니다. 이러한 8가지 종류를 변수형(Type)이라고 합니다. 조그마한 주머니에 큰 수박을 넣을 수 없고 큰 수박을 넣는 주머니에 작은 완두콩 하나를 넣는다면 주머니의 공간을 낭비하게 될 것입니다. 그래서 귤을 담을 수 있는 주머니와 사과를 담을 수 있는 주머니 그리고 수박을 담을 수 있는 주머니 등으로 주머니의 크기를 여러 가지로 구분한 것이 변수형입니다.

변수형이란 것이 생긴 이유는 한정된 메모리 때문입니다. 웹 서버에 100MB의 메모리가 있다고 했을 때 모든 값을 다 넣을 수 있게 하려고 변수 한 개마다 메모리 1MB씩 할당해 준다고 하면, 변수 100개만 만들어 쓰면 메모리를 모두 사용하게 됩니다. 단지 숫자 몇 자리를 저장하려고 하는데 1MB의 공간이 필요할 리 없습니다. 그래서 메모리를 보다 효율적으로 사용하기 위해서 변수를 여러 가지 종류로 나눈 것입니다.

각 변수형에 대해 자세히 알아봅시다.

진리값(boolean)

boolean형은 진리값, 즉 참(TRUE)과 거짓(FALSE) 두 가지 값만을 가지기 위한 변수형입니다. PHP에서 거짓은 0을 뜻하고 0이 아닌 다른 모든 수가 참을 뜻합니다. 따라서 0이냐 아니냐만 판단하면 되므로 1bit의 메모리 공간이 필요합니다. 그러나 메모리 관리에 있어서 너무 작은 공간에 대한 처리가 오히려 부담되기 때문에 메모리 운영을 위한 최소 단위로 1Byte를 사용하고 있습니다. 따라서 진리값은 실제로는 1bit만 필요하지만 메모리의 지나친 공간 절약보다 효율성을 우선시하는 취지로 1Byte의 공간을 사용합니다.

여기서 잠깐

☑ bit? Byte?

bit는 컴퓨터에서 가장 작은 단위로 0과 1, 두 가지를 저장할 수 있는 크기의 용량을 말합니다. (여기서 0과 1 두 개 모두를 저장하는 것이 아니라 둘 중 하나의 값을 저장할 수 있는 것을 말합니다.) 8bit = 1Byte이므로 1Byte는 2의 8승 가지, 즉 256가지를 표현할 수 있는 크기입니다. 보통의 경우 숫자는 메모리를 4Byte 차지하는데, 4Byte는 2의 32승 가지를 표현할 수 있으므로 정수를 사용한다면 0~4294967295 또는 -2147483648~2147483647 사이의 숫자를 저장할 수 있습니다.

정수(Integer)

정수형은 수학적인 의미의 정수를 표시하기 위한 변수형입니다. 십진수와 음수는 다들 알겠지만 8진수와 16진수는 표현법이 새롭습니다. 8진수는 0으로 시작하고 0~7까지의 숫자만을 사용하여 표현하며, 16진수는 0x로 시작하고 0~9와 A~F까지의 총 16가지의 문자로 표현합니다.

```
$a = 1234; # 십진수
$a = -123; # 음수
$a = 0123; # 8진수 (십진수의 83과 같다.)
$a = 0x12; # 16 진수 (십진수의 18과 같다.)
```

일상생활에서는 2진수와 8진수 그리고 16진수를 사용할 일이 거의 없지만 프로그래밍을 하다 보면 이러한 것들을 자주 만나게 됩니다. 왜냐하면 사람은 10진수를 사용하지만 컴퓨터는 2진수를 사용하기 때문입니다. 항간에는 이런 말이 있습니다.

"세상에는 10가지 종류의 사람이 있다. 2진수를 아는 사람과 모르는 사람"

10가지 종류의 사람이 있다고 했는데 두 종류의 사람만 이야기하고 끝이나 버렸습니다. 이게 뭔가 하는 분이 대부분일 거라 생각합니다. 이런 오해가 생기는 이유는 10가지가 2진수로 2가지를 의미하기 때문입니다. 즉, 세상엔 두 종류의 사람이 있는데 2진수를 아는 사람과 그렇지 못한 사람이라는 말입니다. 이런 말이 있을 정도로 프로그래머는 2진수와 8진수 그리고 16진수와 친해져야 할 필요가 있습니다.

여기서 잠깐

PHP는 정수를 표시하기 위해서 4Byte의 메모리 공간을 부여합니다. 위에서도 언급하였듯이 4Byte 공간은 총 2^{32}가지의 표현이 가능합니다. 정수는 부호가 없이 0 이상의 값을 표현할 때는 0~4294967295 사이의 수를, 부호가 있는 정수 값을 표현할 때는 -2147483648~2147483647 사이의 수를 가질 수 있습니다. 총 32개의 bit 중에서 첫 번째 bit를 부호 bit으로 사용하여 0일 때 양수, 1일 때 음수로 표현하기 때문에 실제 숫자는 31개의 bit으로 표현하게 되어 위와 같은 범위를 갖게 됩니다.

부동소수점수(floating point numbers)

부동소수점수는 실수를 표현하면서 부동소수점(floating point) 방식을 사용하는 수를 의미합니다. 부동소수점(浮動小數點) 방식이란 소수점의 위치가 고정되지 않고 떠다니듯 움직인다는 뜻입니다. 부동소수점 방식을 이해하기에 앞서 실수를 나타내는 또 다른 방식인 고정소수점에 대해 먼저 알아볼 필요가 있습니다. 고정소수점 방식은 소수점의 위치를 고정해두는 방식이며 실수를 표현할 때 정수부와 소수부의 범위를 고정하는 방식입니다.

PHP에서는 실수를 표현하기 위해서 8바이트의 공간을 사용합니다. 만약 고정 소수점 방식을 사용한다면 개념적으로 정수부를 나타내려고 4바이트의 공간을 할당하고 소수부를 나타내고자 또한 4바이트의 공간을 할당하는 방식을 생각해 볼 수 있습니다.

정수부

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

소수부

그러나 고정소수점을 사용할 때 큰 수에 대한 표현의 한계가 생깁니다. 정수부를 4바이트 할당하게 되면 부호가 있는 수일 때 2,147,483,647보다 큰 수를 표현할 수 없습니다. 그래서 고정소수점을 사용하지 않고 부동소수점을 사용합니다.

$$(가수) \times (밑수)^{(지수)}$$

부동소수점 방식은 정밀도를 나타내는 유효 값과 자릿수를 나타내는 지수로 표현됩니다. 지수로 자릿수를 표현하기 때문에 1.8×10^{308}까지의 수를 표현할 수 있습니다. 그러나 소수점 아래의 자릿수가 긴 경우 이를 정밀하게 표현하지 못하는 단점이 있습니다. PHP의 경우 소수점 이하 14자리까지의 정밀도를 제공합니다.

```
$a = 1.234;
$a = 1.2e3;
```

PHP에서 부동소수점의 표현은 위와 같습니다. 두 번째의 경우에는 e3은 10의 3승인 1000이 곱해진다는 뜻입니다. 즉, 1200이란 값을 의미합니다.

그런데 부동소수점은 계산 시에 주의해야 할 점이 있습니다. 0.1과 같이 우리가 보기엔 간단한 수도 부동소수점으로는 정확히 표현할 수 없기 때문입니다. 그 이유는 컴퓨터가 0.1을 2진수로 계산하기 때문입니다. 2진수로 0.1을 표현하려면 1/16 + 1/32 + 1/256 + ... 과 같이 합산을 해야 하나 이러한 방식으로는 절대 0.1을 표현할 수가 없습니다. 그래서 0.1을 10번 더해도 1이 되지 않고 0.9999999999999…. 가 됩니다. 따라서 부동소수점수들이 서로 같은지 비교하는 등의 작업은 언

제나 오류의 소지를 가지므로 사용하지 않도록 합니다.

문자열(String)

PHP에서 일련의 문자들을 표현하고자 할 때 문자열 형식을 사용합니다. 문자열 형식은 앞선 논리값이나 정수 등과 달리 메모리 공간의 제약이 없습니다. 즉, 문자열의 길이에 상관없이 제한 없는 문자열을 지원합니다.

PHP에서 문자열을 표시하는 방법에는 큰따옴표("), 작은따옴표('), 히어닥 문법과 나우닥 문법이 있습니다.

① **큰따옴표**

"나는 문자열이다."와 같이 큰따옴표 사이에 문자열을 써넣는 방법입니다. 큰따옴표를 사용하게 되면 따옴표로 둘러싸인 문자열을 해석하는 절차를 가집니다. 그래서 내부에 특수문자나 변수의 사용이 가능합니다.

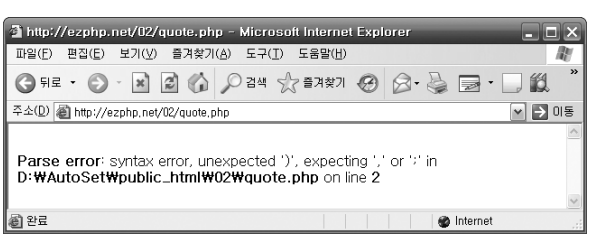
큰따옴표를 이용한 문자열 표현에서는 다음과 같은 특수문자의 사용이 허용됩니다.

특수문자	의미
\n	개행문자(줄 바꿈)
\r	캐리지 리턴
\\	역 슬래시
\t	수평 탭
\"	따옴표
\$	달러 표시

[표 2-3] 문자열에서 사용할 수 있는 특수문자

이러한 특수문자를 지원하는 이유는 가장 먼저 따옴표 내부에 따옴표를 사용할 수 있도록 하기 위해서입니다.

```
<?
    echo "따옴표 안에 따옴표(")를 넣어 보자.";
?>
```



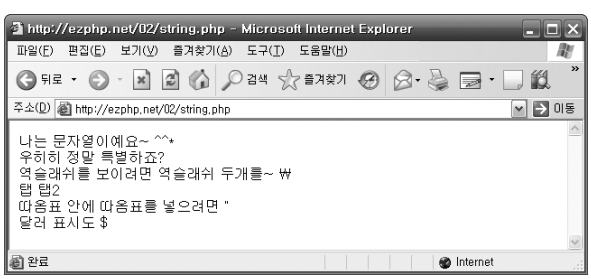
[그림 2-18] 문자열 속에 따옴표가 있는 경우

위와 같이 문자열 속에 따옴표가 있으면 에러가 발생합니다. 왜냐하면 따옴표가 쌍을 이루게 되면서 따옴표로 둘러싸인 부분이 문자열이라고 인식하는데 문자열 안에 따옴표가 나와서 "**따옴표 안에 따옴표(**" 이만큼이 문자열이라고 인식합니다. 그래서 뒤에 나오는 **)를 넣어 보자.**" 부분을 해석할 수 없어서 에러가 발생하는 것입니다.

그래서 문자열 속에 따옴표를 표현할 수 있게 하려고 역슬래시(\)를 이용하기로 하였습니다. 역슬래시 다음에 따옴표가 나온다면 따옴표가 문자열을 구분 짓는 기호가 아니라 문자열 내부에 들어가는 문자로 인식되도록 하기 위해서입니다. 역슬래시를 특별한 기호로 사용하다 보니 역슬래시 자체를 문자열에 넣고싶을 때에도 문제가 발생합니다. 그래서 이 경우에도 역슬래시를 이중으로 사용하면 역슬래시를 문자열 속에 포함할 수 있도록 정했습니다. 이러한 방법에 따라서 따옴표 내부에 변수가 들어오는 경우 때문에 변수명 자체나 $ 기호를 표현하려고 역슬래시를 사용하도록 했습니다. 그 외에 개행문자나 캐리지 리턴 그리고 수평 탭은 출력의 모양을 조절하기 위한 특수기호입니다.

예로 보여 드리겠습니다.

```
<?
     echo "나는 문자열이예요~ ^^*<BR>\n";
     echo "우히히 정말 특별하죠? <BR>\r";
     echo "역슬래쉬를 보이려면 역슬래쉬 두개를~ \\<BR>\n";
     echo "탭\t 탭2<BR>\n";
     echo "따옴표 안에 따옴표를 넣으려면 \" <BR>\n";
     echo "달러 표시도 \$ <BR>\n";
?>
```



[그림 2-19] 문자열의 회피 결과

각 줄과 출력된 결과를 비교해보면 어떤 식으로 특수문자가 동작하는지 알 수 있습니다. 2번째 줄에 역 슬래시, 5번째 줄에 따옴표 그리고 6번째 줄에 달러($) 기호가 출력된 것을 확인할 수 있습니다. 그런데 개행문자와 캐리지 리턴, 그리고 탭은 어떻게 된 건지 보이지가 않습니다. 이 특수문자들은 웹 브라우저에서 변화되는 것들이 아니라 출력되는 HTML에 영향을 주는 기능이며 HTML 소스를 보기 좋게 정렬하는 등의 기능을 하려는 것입니다. 이를 확인하기 위해서 웹 브라우저에서 소스 보기를 통해서 HTML 소스를 열어 봅시다.

[그림 2-20] 출력된 HTML 소스

HTML 소스를 보면 확실히 개행문자가 뭔지 보이죠? HTML 소스 코드가 한 줄씩 줄 바꿈되어 있는 것이 바로 개행문자의 흔적입니다. 캐리지 리턴이 개행문자와 같은 결과를 나타내는데 실제로 캐리지 리턴은 커서를 제일 처음으로 보내는 역할을 합니다. 그리고 3번째 줄을 보면 탭과 탭2 사이에 공간이 있는 것을 알 수 있습니다. 바로 이 공간이 탭이 적용된 것을 나타냅니다.

프로그램을 작성하면서 소스가 잘 정리되어 있으면 작성한 소스를 이해하기도 쉽고 버그 수정도 편리하다는 장점이 있습니다. PHP 프로그래머는 PHP 소스 코드뿐만 아니라 그 결과가 되는 HTML 소스 코드도 정확히 출력되는지 확인해야 하므로 보기 좋게 정리하는 것이 매우 중요합니다. 그러니 여러분도 조금 귀찮더라도 될 수 있으면 PHP 소스뿐만 아니라 HTML 소스도 개행문자 등을 통해서 보기 좋게 잘 정리하기 바랍니다.

이외에도 큰따옴표 내에서는 변수 사용이 가능합니다. 예를 들어 $name 이라는 변수에 "홍길동"이라는 문자열이 저장되어 있다면, "**나의 이름은 $name 입니다.**"는 "**나의 이름은 홍길동입니다.**"와 같은 효과를 가지게 됩니다. 만약 "나의 이름은 $name 입니다." 자체를 출력하고 싶다면 앞서 배운 특수 문자를 사용하여 다음과 같이 바꾸어야 합니다.

```
"나의 이름은 \$name 입니다."
```

② 작은따옴표

'나는 문자열이다.'와 같이 작은따옴표 안에 문자열을 써넣는 방법입니다. 작은따옴표를 이용한 방법은 큰따옴표를 이용할 때와 차이가 있는데, 작은따옴표 내부에 쓰인 특수문자와 변수가 허용되지 않는다는 것입니다.

```
echo '작은따옴표 안에서는\n 특수문자와 변수가 $name 동작을 안 해요.';
```

작은따옴표를 사용한 문자열은 기본적으로 작은따옴표 자체를 출력하기 위한 \'와 역슬래시 자체를 출력하기 위한 \\만이 해석될 뿐 다른 어떤 것도 해석되지 않습니다. 따라서 변수나 개행문자 등을 사용하면 작성한 그대로 출력됩니다. 작은따옴표를 사용한 문자열은 이처럼 문자열을 해석하는 절차가 적기 때문에 단순한 문자열을 출력하고자 할 때 가장 빠르고 간편하게 사용할 수 있습니다. 반면에 단순 문자열을 큰따옴표로 사용하는 경우 불필요한 문자열 해석 절차를 가지므로 효율적이지 못합니다. 그러나 성능의 차이가 눈에 띌 만큼 크지 않기 때문에 프로그램 작성의 편의적 입장에서 이럴 때에도 큰따옴표를 사용하곤 합니다.

③ **히어닥 문법(Heredoc)**

히어닥 문법은 문자열의 시작과 종료를 알려 주는 방식을 이용하여 문자열을 보다 편리하게 표현하는 방법입니다. 문자열의 시작을 알릴 때는 〈〈〈 다음에 문자열에 대한 식별자를 표시하고 문자열이 종료될 때는 그 식별자를 다시 표시하는 방법입니다.

```
$str = <<<EOD
문자열이라네~
나도 문자열이라네~
EOD;
```

이처럼 EOD라는 식별자를 사용하여 문자열을 표현했고 그 결과는 큰따옴표를 사용한 것과 같습니다. 큰따옴표에 비해 장점이 있다면 문자열 내부에 따옴표를 추가하고자 할 때 굳이 역슬래시를 쓸 필요가 없다는 것입니다. 왜냐하면 문자열의 시작과 끝을 구분하기 위해서 따옴표를 사용하지 않기 때문입니다.

④ **나우닥 문법(Nowdoc)**

PHP 버전 5.3.0부터 나우닥 문법이 추가되었습니다. 나우닥 문법은 히어닥 문법과 유사하나 다른 점은 식별자에 작은따옴표가 붙는다는 표기법의 차이와 변수 등의 문자열 해석을 하지 않는다는 차이가 있습니다. 즉, heredoc은 따옴표 방식과 유사하고 nowdoc은 작은따옴표 방식과 유사한 것입니다.

```
$str = <<<EOT
내 이름은 "$name"입니다.
문자열이 해석되지 않습니다.
EOT;
```

⑤ **변수의 해석(Variable Parsing)**

따옴표와 히어닥 문법에서는 문자열 내에 변수가 있을 때 이를 해석해서 처리한다고 말씀드렸습니다. 이때 문자열 내에 아무렇게나 변수를 넣었다간 잘못된 결과를 일으킬 수 있습니다. 문자열에 변수를 추가하는 방법에는 두 가지 방법이 있습니다.

"문자열 내에 변수를 넣는 가장 쉬운 방법 : $money"

"문자열 내에 변수를 넣는 가장 안전한 방법 : {$money}"

기본적인 변수는 첫 번째 방법처럼 그냥 사용해도 문제없이 동작합니다. 그러나 변수$money가 1000원 단위의 금액이어서 뒤에 '000'을 덧붙여 주고 싶을 때는 어떻게 해야 할까요? 간혹 아무런 생각 없이 $money000라고 쓰는 경우가 있습니다. 이 경우 PHP는 가장 긴 식별자를 찾기 때문에 $money000라는 변수를 찾고 이 변수는 존재하지 않기 때문에 올바른 결과를 얻을 수 없게 됩니다. 이런 때에는 간단히 변수와 출력하고자 하는 문자열 사이에 공백을 만들어 해결할 수 있으나 $money 000의 결과는 1000 000과 같이 중간에 공백이 생기기 때문에 원하는 출력 방법이 아닙니다. 이럴 때를 대비하여 보다 안전한 방식인 중괄호를 사용할 수 있습니다. {$money}000과 같이 변수명이 어디까지인지를 중괄호를 통해서 명확히 알려 주어 PHP가 혼동하지 않게 할 수 있습니다.

리소스와 NULL

리소스는 파일이나 데이터베이스 그리고 이미지 작업과 같은 외부적인 자원을 나타내는 데 사용하는 변수형입니다. 리소스 타입은 이와 같이 특별한 경우에 사용되는 용도로 만들어진 것으로 다른 값으로 변환하거나 하는 등 일체의 작업이 불가능합니다. 또한 리소스는 자동으로 해제되기 때문에 수동으로 메모리를 해제하지 않아도 됩니다.

NULL은 아무 값도 갖지 않는 변수를 말합니다. 변수에 아직 아무런 값도 할당하지 않았을 때나 상수로 NULL을 직접 할당했을 때 그리고 unset 함수로 변수를 제거했을 때 해당 값을 NULL이라고 합니다.

PHP에서는 변수의 데이터형에 대해 매우 유연한 정책을 시행하고 있습니다. 다른 프로그래밍 언어에서는 변수를 "정수를 저장하는 용도로 사용하겠다!"라고 먼저 선언을 하고 해당 변수에는 정수만을 저장합니다. 이에 반해서 PHP는 변수의 선언이 없고 변수에 할당하는 값의 형식에 따라서 데이터형이 자동으로 결정됩니다. 그래서 정수를 저장하던 변수에 부동소수점수를 저장해도 올바르게 동작하고 심지어 문자열을 정수형으로 변환할 수도 있습니다. 이처럼 데이터형의 변환이 자유로워서 프로그래머는 데이터형에 얽매이지 않기 때문에 매우 편하게 프로그램을 작성할 수 있습니다. C와 같은 언어에서는 데이터형이 다른 경우 경고 메시지나 에러 메시지를 출력하기 때문에 해당 용도로 사용하거나 아니면 용도가 변경될 때마다 데이터형을 변환시켜야 하는 불편함이 있습니다.

변수의 종류와 외부로부터의 변수

변수에는 우리가 이름을 짓고 용도에 따라 만들어 사용할 수 있는 사용자 변수와 미리 정의된 변수 그리고 외부로부터 넘어온 변수가 있습니다. 사용자 변수는 앞에서 식별자에 $ 기호를 붙여서 필요에 따라 사용할 수 있었습니다. 반면 미리 정의된 변수는 서버 종류, 버전 등이나 기타 환경에 따라 변하는 변수들로 이미 이름이 지어진 변수들입니다. 이러한 변수들은 프로그램의 어디서든 해당 이름을 통해서 접근할 수 있습니다. 그래서 슈퍼 전역 변수라고 불립니다. 슈퍼 전역 변수는 웹 서버의 설정값이나 서버 시스템의 환경 그리고 사용자의 입력과 관련한 변수들이 지정되어 있으며 이 변수들은 우리가 인위적으로 값을 수정하거나 일반 변수로 사용할 수 없습니다.

$_GET	HTTP GET 방식으로 넘어온 변수
$_POST	HTTP POST 방식으로 넘어온 변수
$_COOKIE	HTTP 쿠키 변수
$_FILES	업로드 시 파일 정보 변수
$_SESSION	세션 변수
$_SERVER	웹 서버와 PHP 환경에서의 환경 변수

$_ENV	서버 시스템의 환경 변수

[표 2-4] PHP 슈퍼 전역 변수

슈퍼 전역 변수는 모두 배열로 되어 있기 때문에 $_ENV['OS']와 같은 방법으로 변수에 접근할 수 있습니다. 각 슈퍼 전역 변수는 앞으로 다시 등장할 예정이므로 이런 것들이 있다는 정도만 알고 넘어가기 바랍니다. 만약 여러 가지 슈퍼 전역 변수에 대해 알고 싶다면 phpinfo() 함수를 실행해서 시스템의 여러 가지 환경 변수들을 볼 수 있습니다.

마지막으로 외부에서 정의되어 넘어온 변수가 있습니다. 외부로부터의 변수라고 하니 마치 외계인과 같은 느낌이 드는군요. 이런 변수에는 HTML 폼을 통해 전달된 변수와 쿠키로 전달된 변수가 있습니다. 이것은 모두 사용자의 브라우저에서 작성된 값이 웹 서버에 전달된 변수입니다. 그래서 외부로부터의 변수라고 불립니다. 이 두 변수에 대해서는 뒤에서 각각 자세히 배우겠습니다.

변수의 범위와 상수

앞에서 슈퍼 전역 변수가 어떤 것들인지 가볍게 알아보았습니다. 슈퍼 전역 변수는 앞에 뭔가 대단한 듯한 슈퍼란 말이 붙기는 했지만 기본적으로는 전역 변수라는 의미입니다. PHP에는 지역 변수라느니 전역 변수라느니 하는 것이 있는데 말 그대로 지역 변수는 어떠한 일부분 영역에서만 적용되는 변수이고, 전역 변수는 전국 방방곡곡, 어느 영역이든 사용 가능한 변수입니다.

지역 변수와 전역 변수는 어깨 형님들로 따지면 지역구와 전국구에 비교할 수 있습니다. 목포에 회칼이라 불리는 지역구 형님이 있고 종로에 쌍칼이란 유명한 전국구 형님이 있다고 합시다. 각자 그 지역에서는 이름만 들먹여도 다들 알만큼 유명한 형님들이라고 합시다. 그런데 회칼이 서울로 상경해서 "나 회칼이야~"라고 한들 누가 알아줄까요? 하지만 전국구인 쌍칼은 목포에 가서 "나 쌍칼이다."라고 하면 다들 벌벌 떨겠죠.

그런데 하루는 전국구 쌍칼이 부산에 가서 "나 쌍칼이다."라고 했는데 거짓말하지 말라며 되레 화를 냅니다. 왜 그런가 했더니 부산에도 쌍칼이 있었던 것이었습니다. 상대방은 부산의 쌍칼 얼굴을 아는데 자기가 쌍칼이라 그러니까 거짓말을 한다고 생각한 것입니다. 황당한 경험을 한 쌍칼은 전국구 연합에 전국구만의 표식을 만들자고 건의합니다. 지방순례를 할 때엔 그 표식을 가지고 가서 자신이 전국구임을 확인시키자고 말이죠.

이때의 전국구 표식이 바로 global입니다.

```
<?
  $a = 1; //전국구 쌍칼

        function Busan()
```

```
        {
        echo $a; //지역구에서 쌍칼을 찾음
        }

    Busan();
?>
```

결과는 아무것도 출력되지 않습니다. 우리는 아직 함수라는 것을 배우지 않았기 때문에 소스 코드가 다소 생소할 수 있을 것입니다. 함수는 다음과 같은 형식을 갖습니다.

```
function 함수이름( 파라미터 ) {

}
```

이렇게 함수를 정의하면 정의된 함수는 함수이름(파라미터); 식으로 실행할 수 있습니다. 그런데 함수는 자신만의 세계를 만들어 영역을 형성합니다. 그래서 부산과 같은 지역구가 만들어집니다. 함수 내부에서 새롭게 사용하는 변수들은 모두 지역 변수가 됩니다.

전국구 $a가 Busan 함수 영역으로 들어오면 지역구에서는 $a가 지역구에 있는지 찾습니다. 그런데 부산 지역에는 $a가 없으므로 누군지 모르니까 무시합니다.

그래서 뭘 만들었죠? 전국구만의 표식!!!

```
<?
    $a = 1; //전국구 쌍칼

    function Busan()
    {
        global $a; //나 전국구 쌍칼이야~
        echo $a;
    }

    Busan();
?>
```

이 예제의 결과는 1이 출력됩니다. Busan 함수 내에 global 키워드를 통해서 변수 $a를 전역 변수임을 알려 주어서 이와 같은 결과를 나타낸 것입니다. 전국구 표식을 보여 주니 다들 "형님~ 형님~" 그러는군요.

정적(static) 변수

함수는 함수만의 작업 영역을 갖고 있다고 말씀드렸습니다. 그런데 이 작업 영역은 함수를 호출할

때 새로 만들어졌다가 함수의 처리가 완료되면 작업 영역을 없애 버립니다. 마치 해결사처럼 필요할 때 등장했다가 일 처리를 다하고 나면 흔적도 없이 사라지는 것과 같습니다. 그래서 함수 내부에 새로 선언한 지역 변수는 모두 흔적도 없이 사라집니다. 다시 함수를 호출해도 처음부터 새롭게 지역 변수가 만들어지기 때문에 기존에 가지고 있던 값을 유지할 수가 없습니다.

그런데 어떤 경우에는 함수가 다시 호출될 때 지역 변수의 값이 그대로 유지되어야 할 때가 있습니다. 이럴 때 사용하는 것이 **static** 키워드입니다. static 키워드를 통해서 선언된 변수를 정적 변수라고 합니다.

```
function Test()
{
   echo $a;
   $a = "제발 출력해줘~";
}
```

이 함수는 몇 번을 호출하더라도 아무것도 출력되지 않습니다. 처음 호출되었을 때는 변수 $a에 아무 값도 할당되지 않아서 그렇다고 하더라도 곧바로 문자열이 할당되기 때문에 두 번째 호출부터는 출력되어야 할 텐데 말입니다. 앞에서 말씀드렸듯이 함수가 종료됨과 동시에 내부의 지역 변수들이 모두 사라져 버리기 때문에 아무리 값을 저장해주더라도 이전 값을 유지할 수 없습니다.

```
function Test()
{
   static $a = 0;
   echo $a;
    $a++;
}
```

여기에 static 키워드를 사용하여 지역 변수인 $a를 선언해 주면 0이 출력되고 두 번째는 1, 세 번째는 2와 같이 계속 호출할 때마다 1씩 증가하는 결과를 얻을 수 있습니다. 이렇게 static 키워드를 통해서 변수를 선언하면 변수는 소멸하지 않고 계속 그 값을 유지하게 됩니다. 그러나 이 변수가 전역적으로 사용될 수 있는 것은 아닙니다. 이 정적 변수는 단지 그 지역에서만 유효한 값을 유지합니다.

상수

상수는 아주 간단합니다. 말 그대로 항상 똑같은 값을 가지는 변하지 않는 수입니다. 그래서 딱 한 번만 설정하면 전역 변수처럼 사용할 수 있습니다. 또한 변하지 않는 수이기 때문에 한번 설정된 값은 변경할 수 없습니다. 상수는 대소문자 구별을 하지만 관례로 대문자로 표기합니다. 또한 변수처럼 $ 기호를 사용하지 않고 반드시 define() 함수를 이용하여 상수를 정의합니다.

```
<?
  define ("HELLO","안녕하세요");
  echo HELLO;
?>
```

이 예제는 **안녕하세요**라는 문구를 출력합니다. define() 함수는 다음과 같은 방법으로 사용할 수 있습니다.

```
define("상수명","상수에 저장될 값");
```

위의 예제에서 보듯이 define 이후의 HELLO는 **안녕하세요**를 나타냅니다. 그렇다고 모든 HELLO까지 **안녕하세요**로 바뀌는 것은 아닙니다.

```
echo "HELLO";
```

위와 같이 문자열 속에 HELLO가 있다면 이는 **안녕하세요**로 출력되지 않고 글자 그대로 **HELLO**가 출력됩니다.

상수도 변수와 마찬가지로 미리 정의된 상수가 있습니다. 예를 들면 PHP_VERSION(현재 PHP 버전), PHP_OS(현 시스템의 운영체제), M_PI(원주율) 등이 있습니다. 이외에 정의된 상수를 알고 싶다면 다음 웹 페이지를 방문하기 바랍니다.

http://docs.php.net/manual/kr/reserved.constants.php

연산자(operator)

연산자는 숫자나 문자열과 같은 어떤 값과 변수 등을 사용하여 더하거나 곱하고 어느 값이 더 크고 작은지를 비교하는 등의 연산을 하는 기호를 말합니다. PHP는 다양한 연산자를 제공하는데 비슷한 부류로 분류해보면 다음과 같습니다.

① 대입 연산자
② 산술 연산자
③ 증감 연산자
④ 비교 연산자
⑤ 논리 연산자
⑥ 문자열 연산자
⑦ 배열 연산자
⑧ 비트 연산자

⑨ 실행 연산자

⑩ 에러 제어 연산자

대입 연산자

일반적으로 "=" 기호는 같다라는 의미라고 생각합니다. 왜냐하면 우리는 어려서부터 "A = B"라고 하면 "A는 B이다." 혹은 "A equal B"라고 읽고 "A와 B는 같다."라는 의미로 배웠기 때문입니다. PHP에서는 우리가 아는 상식과 다르게 해석이 됩니다. PHP에서 "A = B"의 의미는 "B의 값을 A에다가 대입한다."라는 뜻입니다. 그래서 "=" 기호를 대입 연산자라고 합니다. 그리고 같다는 의미로는 "==" 기호를 사용하고 있습니다.

오른쪽에 있는 값을 왼쪽의 값에 대입하니까 결과적으로는 A와 B가 같아집니다. 그러나 의미는 천지 차이입니다. 특히 프로그램에서는 치명적인 문제를 발생시킬 수 있습니다. 아직 제어 구조와 비교 연산자를 배우지 않아서 자세히 설명 드릴 수는 없지만 아래의 문장이 가장 대표적으로 대입 연산자를 같다는 의미로 잘못 사용한 예입니다.

```
if ( $a = 1 ) { echo "성공!"; }
```

if라는 제어 구조를 앞으로 배우게 될 텐데, 간단히 설명드리면 만약 괄호 안이 참이면 **성공!**이라는 글자를 출력합니다. 프로그래머는 $a와 1이 같은 경우에 참이라고 생각하면서 프로그램을 작성했지만 실제로는 '같다'의 의미가 아닌 '대입'이 되기 때문에 $a에는 1이 대입되고 그 결과 아래의 문장과 같은 의미로 변경됩니다.

```
if ( 1 ) { echo "성공!"; }
```

1은 앞에서도 말씀드렸듯이 0이 아닌 수이므로 참을 의미합니다. 따라서 $a 값이 어떤 값을 갖든지 상관없이 이 문장은 언제나 **성공!**이란 글자를 출력합니다.

대입 연산자는 매우 간단해서 '같다'라는 의미가 아니라는 것만 머릿속에 꼭 기억하면 됩니다.

산술 연산자

산술 연산은 흔히 우리가 접하는 더하기, 빼기, 곱하기, 나누기를 말합니다. 어려운 말로 가감승제(加減乘除)입니다. 여기에 덧붙여 나머지 연산이 있습니다. 나머지 연산은 말 그대로 나누기를 하고 남은 나머지를 반환해주는 연산입니다.

연산자	이름	예
+	덧셈	$a + $b
−	뺄셈	$a − $b

*	곱셈	$a * $b
/	나눗셈	$a / $b
%	나머지	$a % $b

[표 2-5] 산술 연산자

산술 연산자에 대해서는 더는 설명이 필요 없으리라 생각합니다. 만약 헷갈리면 초등학교 다니는 동생한테 물어 보세요. : Ð

증감 연산자

증감 연산자는 선(先) 처리 연산자와 후(後) 처리 연산자가 있습니다. 먼저 증감하고 값을 반환하는 것을 선 처리 연산자라 하고, 값을 우선 반환한 후 증감을 하는 것을 후처리 연산자라 합니다. 증감 연산자를 사용하면 1씩 증가하거나 1씩 감소합니다.

예	이름	의미
++ $a	선 증가 연산자	$a를 1 증가 후 $a를 반환
$a ++	후 증가 연산자	$a를 반환 후 $a 에 1 증가
– – $a	선 감소 연산자	$a를 1 감소 후 $a를 반환
$a – –	후 감소 연산자	$a를 반환 후 $a 에 1 감소

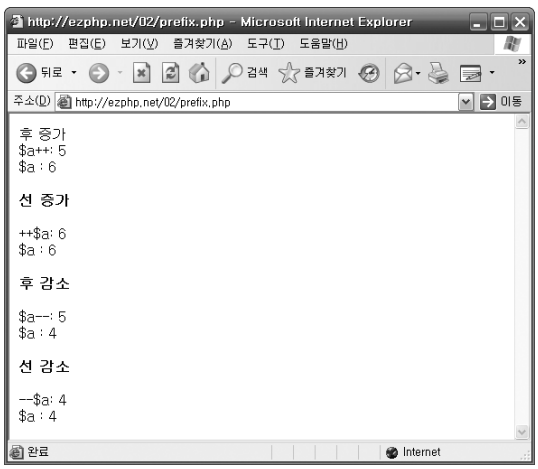
[표 2-6] 증감 연산자

예제를 통해 쉽게 이해할 수 있습니다.

```
<?
  echo "후 증가<br>";
  $a = 5;
  echo " \$a++: " . $a++ . "<br>\n";
  echo " \$a : " . $a . "<br>\n";

  echo "<h4>선 증가</h4>";
  $a = 5;
  echo " ++\$a: " . ++$a . "<br>\n";
  echo " \$a : " . $a . "<br>\n";

  echo "<h4>후 감소</h4>";
  $a = 5;
  echo " \$a--: " . $a-- . "<br>\n";
  echo " \$a : " . $a . "<br>\n";

  echo "<h4>선 감소</h4>";
  $a = 5;
  echo " --\$a: " . --$a . "<br>\n";
```

```
    echo " \$a : " . $a . "<br>\n";
?>
```

이 예제로 기본값 5가 할당된 변수 $a가 증감 연산자를 통해서 어떻게 값이 변화하고 출력되는지 알 수 있습니다. 소스 코드와 그 결과를 하나씩 비교해가면서 보면 증감 연산자가 어떻게 동작하는지 쉽게 이해할 수 있을 것입니다.



[그림 2-21] 증감 연산자 처리결과

비교 연산자

비교 연산자는 두 값을 비교하는 연산자를 말합니다. 두 값을 비교하여 참인지 거짓인지를 판단합니다.

연산자	예	의미
〈	$a 〈 $b	$a가 $b보다 작으면 참
〉	$a 〉 $b	$a가 $b보다 크면 참
〈=	$a 〈= $b	$a가 $b보다 작거나 같으면 참
〉=	$a 〉= $b	$a가 $b보다 크거나 같으면 참
==	$a == $b	$a와 $b의 값이 같으면 참
!=	$a != $b	$a와 $b의 값이 다르면 참
〈〉	$a 〈〉 $b	$a와 $b의 값이 다르면 참
===	$a === $b	$a와 $b가 같으면 참
!==	$a !== $b	$a와 $b가 같지 않으면 참

[표 2-7] 비교 연산자

비교 연산자 중에서 === 연산자와 !== 연산자에서 같다의 의미는 두 값이 같고 그 데이터형이 같을 경우를 말합니다. 예를 들면 0 == NULL은 참이지만 0 === NULL은 거짓입니다. 왜냐하면 값은 같지만 데이터형이 하나는 정수형이고 하나는 NULL 형식이기 때문입니다.

논리 연산자

논리 연산자는 논리 조건들에 대해 참 거짓을 판단하는 연산자입니다. 일반적으로 하나 이상의 논리적 조건을 통해서 하나의 결론을 얻고자 많이 사용합니다. 예를 들면 변수 $a가 0보다 크고 100보다는 작은 범위 내에 있는지를 알아보고자 다음과 같은 두 수식을 사용할 수 있습니다.

```
$a > 0
$a < 100
```

이 경우 두 조건을 모두 참으로 만족시켜야 참이라고 할 수 있습니다. 그래서 이 두 조건을 모두 만족한다는 의미에서 다음과 같은 논리 연산자를 사용할 수 있습니다.

```
$a > 0 && $a < 100
```

이처럼 참 거짓으로 이루어진 연산에 사용하는 연산자가 바로 논리 연산자입니다.

예	이름	결과
$a and $b	AND	두 변수 모두 참일 때 TRUE
$a or $b	OR	두 변수 중 적어도 하나가 참이면 TRUE
$a xor $b	XOR	둘 중 하나만 참이면 TRUE, 둘의 진리값이 같으면 FALSE
!$a	NOT	변수가 거짓일 때 TRUE, 참이면 FALSE
$a && $b	AND	두 변수 모두 참일 때 TRUE
$a || $b	OR	두 변수 중 적어도 하나가 참이면 TRUE

[표 2-8] 논리 연산자

문자열 연산자

문자열 연산자는 문자열을 서로 이어주는 역할을 합니다. 문자열을 서로 결합하고자 한다면 '.' 기호를 사용할 수 있습니다.

```
$a = " 안녕 ";
$b = " 하세요";
```

위와 같이 두 문자열이 있을 때 이 두 문자열을 연결하고 싶으면 다음과 같이 문자열 연산자를 사용할 수 있습니다.

```
$c = $a . $b;
```

변수와 변수 사이에 점을 찍어주는 것만으로 문자열은 서로 연결됩니다. 만약 두 문자열이 합쳐진 문자열 $c에 한마디를 더 추가하고자 한다면 다음과 같은 방식을 쓸 수 있습니다.

```
$c .= " 반가워요~";
```

$c .= "";의 의미는 $c = $c . ""; 와 같습니다. 그래서 결과적으로 **$c = 안녕 하세요 반가워요~;** 가 됩니다.

배열 연산자

배열을 서로 더하려면 + 기호를 이용할 수 있습니다. + 기호를 사용하면 왼쪽의 배열에 오른쪽 배열을 더하게 됩니다.

```
<?
  $a = array("a" => "사과", "b" => "바나나");
  $b = array("a" =>"배", "b" => "딸기", "c" => "포도");

  $c = $a + $b;
?>
```

이 예제의 결과는 다음 문장과 같습니다.

```
$c = array("a" =>"사과", "b" => "바나나", "c" => "포도");
```

예제에서 $a 배열에 $b 배열을 덧붙이게 되는데 이미 $a 배열에 a와 b가 있으므로 같은 키를 가지는 값은 덮어쓰지 않고 중복되지 않는 것만 추가됩니다.

비트 연산자

비트 연산은 숫자를 2진수로 생각하여 각각의 비트별로 계산하는 연산입니다.

	이름	결과
$a & $b	AND	양쪽의 비트가 모두 1인 자리를 1로, 나머지는 0으로 세팅
$a | $b	OR	둘 중 적어도 하나의 비트가 1이면 그 자리를 1로, 모두 0이면 0으로 세팅
$a ^ $b	XOR	비트가 같으면 그 자리를 0으로, 다르면 1로 세팅
~ $a	NOT	모든 비트를 0은 1로, 1은 0으로 세팅
$a 〈〈 $b	Shift left	모든 비트를 왼쪽으로 $b 자리씩 이동(한번 이동은 2를 곱한 것과 같음)
$a 〉〉 $b	Shift right	모든 비트를 오른쪽으로 $b 자리씩 이동(한번 이동은 2를 나눈 것과 같음)

[표 2-9] 비트 연산자

실제 예를 통해서 이해해 보도록 합시다.

예	설명	결과
12 & 5	1100과 0101에서 모두 1인 자리만 1, 따라서 0100	4 (0100)
12 | 5	1100과 0101에서 하나만 1이어도 1, 따라서 1101	13 (1101)
12 ^ 5	1100과 0101에서 같으면 0 다르면 1, 따라서 1010	10 (1010)
~ 12	1100을 반대로, 따라서 0011	3 (0011)
12 〈〈 2	1100을 왼쪽으로 두 자리 이동, 따라서 110000	48 (110000)
12 〉〉 2	1100을 오른쪽으로 두 자리 이동, 따라서 0011	3 (0011)

[표 2-10] 비트 연산의 실제 예

비트 연산은 우리가 쉽게 접하지 못하기 때문에 이해하기가 쉽지 않습니다. 그러나 위의 예제를 여러 번 살펴보면 그렇게 어렵지는 않습니다. 그런데 PHP에서는 비트 연산을 사용할 경우가 극히 드뭅니다. 따라서 비트 연산의 원리 정도만 이해하고 넘어가도록 합시다.

실행 연산자

실행 연산자(`)를 이용하여 서버에 쉘(Shell) 명령을 실행할 수 있습니다. 실행 연산자는 백틱(backticks)으로 작은따옴표가 아닙니다. 백틱키는 키보드의 Esc 키 바로 아래의 키를 말합니다.

```
<?
  $output = `ls -al`;
  echo "<pre>$output</pre>";
?>
```

이 연산자는 shell_exec() 함수를 사용하는 것과 같습니다. 위의 예는 리눅스 시스템에서 파일 목록 보기의 명령을 쉘을 통해 실행하고 그 결과를 출력하는 것입니다. ls 명령은 리눅스 전용이기 때문에 윈도우 환경에서는 동작하지 않습니다. 만약 윈도우 환경 사용자라면 다음과 같은 방법으로 위의 예제와 유사한 결과를 얻을 수 있습니다.

```
<?
  $output = `dir`
  echo "<pre>$output</pre>";
?>
```

에러 제어 연산자

PHP 표현식에서 에러가 발생할 수 있는 곳에 @ 기호를 붙이므로 해서 에러를 출력하지 않게 하는 연산자입니다.

```
<? @include "a.php"; ?>
```

만약 a.php 파일이 없으면 include가 실패하여 에러를 표시하지만 에러 제어 연산자가 있기 때문에 에러를 표시하지 않고 넘어가게 됩니다. 에러가 발생해도 프로그램의 실행에는 상관없는 경우, 이 연산자를 사용하면 됩니다.

연산자 우선순위

10가지 종류의 연산자에 대해 알아보았습니다. 이러한 많은 연산자는 서로 복합적으로 섞어서 많이 사용하므로 연산자 간에 어느 연산자가 먼저 처리되어야 할지를 결정해야 할 때가 생깁니다. 이는 우리가 사칙연산을 할 때 덧셈과 뺄셈보다 곱하기와 나누기를 먼저 계산하는 것과 같습니다.

$1 + 2 \times 3 \neq 9$

$1 + 2 \times 3 = 7$

위의 예처럼 앞에서부터 순서대로 계산하면 9라는 값을 얻게 되지만 곱셈이 덧셈보다 우선순위가 높아서 제일 먼저 곱셈을 계산해야 합니다. 그래서 결과는 9가 아니라 7이 됩니다. 이처럼 여러 개의 연산자가 섞여 있으면 우선순위를 알아야만 올바른 결과를 얻을 수 있습니다. 그래서 PHP에서는 연산자 간의 우선순위를 표로 제공하고 있습니다. 다음 표에서 아래일수록 우선순위가 높은 연산자입니다.

연산자 (아래일수록 우선순위가 높음)
,
or
xor
and
print
= += -= *= /= .= %= &= |= ^= 〈〈= 〉〉=
? :
||
&&
|
^
&
== != === !==
〈 〈= 〉 〉=
〈〈 〉〉
+ - .

* / %
! ~ ++ -- (int) (float) (string) (array) (object) @
[
new

[표 2-11] 연산자 우선순위

그런데 표를 보니 연산자가 너무 많아서 우선순위를 외우기가 너무 힘이 들 것 같습니다. 사실대로 말씀드리면, 이 표를 외운다고 해서 유능한 프로그래머가 되는 것도 아닙니다. 그래서 대부분의 선배 프로그래머들은 우선순위를 외우는 것보다 괄호를 사용하길 권합니다. 복잡한 수식에서 우선순위를 모두 고려하여 수식을 완성하는 것은 실수를 유발할 가능성이 크고 또한 코드를 이해하는데 심각한 문제를 일으킬 수 있습니다. 그래서 명확한 순서를 제외한 나머지 부분을 괄호로 묶어서 연산 순서를 확실히 결정짓는 것입니다.

제어 구조

제어 구조는 PHP의 흐름을 제어하는 구조입니다. 즉, 위에서 아래로 차례로 실행되는 흐름에서 다시 위로 되돌아가게 한다거나 아니면 어느 부분을 건너뛴다거나 하는 프로그램의 흐름을 제어하는 것입니다.

프로그램의 구조는 크게 세 부분으로 나눌 수 있습니다. 첫째, 수식을 처리하는 계산 부분입니다. 둘째, 여러 가지 조건에 따라서 어느 것을 처리할지를 판단하는 부분입니다. 마지막으로 이러한 일련의 과정을 반복하는 부분입니다. 이 세 부분이 섞이고 조화를 이뤄서 하나의 프로그램을 완성합니다. 제어 구조는 두 번째와 세 번째 부분을 다루는데 사용합니다.

if

레오나르도 디카프리오와 톰 행크스가 주연한 스티븐 스필버그 감독의 〈Catch me if you can〉이란 영화가 있습니다. 미국에 실존했던 희대의 사기꾼 이야기인데 영화 제목의 의미는 '잡을 수 있으면 나 잡아봐라.' 정도로 해석할 수 있겠습니다. 이 영화의 제목처럼 if는 '만약 ~하면'이란 뜻입니다. PHP에서도 마찬가지로 if 문은 '만약 A와 B가 같으면, C에 1을 더하라.'와 같은 용도로 사용됩니다. if 문이 어떤 형태를 보이는지 알아봅시다.

```
if ( 표현식 ) 명령문
```

표현식에는 주로 참이나 거짓으로 판단할 수 있는 수식이 사용되며 만약 표현식 부분이 참이라면

다음 명령문을 실행합니다. 반면에 표현식이 참이 아니라 거짓이라면 명령문은 처리되지 않습니다. 그래서 if 문은 어떤 조건을 판단하여 코드를 수행할 것인지 그렇지 않을 것인지를 결정하는 데 사용합니다. 예를 통해서 자세히 알아봅시다.

```
<?
    $a = 3;
    $b = 1;

    if ( $a > $b )  echo "$a 은(는) $b 보다 크다.";
?>
```

위의 예제는 두 변수의 값을 비교하여 변수 $a의 값이 $b의 값보다 더 크면 뒤따르는 echo 문을 실행합니다. 이 경우 변수 $a가 $b보다 크므로 결과는 다음과 같이 출력됩니다.

```
3 은(는) 1 보다 크다.
```

만약 변수 $a = 1이고 $b = 3이었다면 어떤 말도 출력되지 않을 것입니다.

위의 예제에서는 하나의 echo 문만을 수행하였습니다. 그러나 많은 경우에서 하나 이상의 구문을 수행해야 할 경우가 있습니다. 지금 우리가 아는 지식으로는 어떤 조건에 따라 두 개의 구문을 수행하려면 if 문을 두 번 사용해야 합니다.

```
if ( $a > $b ) echo "$a 은(는) $b 보다 크다.";
if ( $a > $b ) echo "$a 와(과) $b 의 차는 " . ($a - $b) . "이다.";
```

그러나 두 개의 구문이 아니라 10개 혹은 100개의 구문이라면 위와 같은 방식으로는 한계가 있습니다. 또한 한 번 비교했던 표현식($a 〉 $b)을 계속해서 비교하는 것은 어린 아이에게 "1+1은 얼마지?"하고 계속해서 묻는 것과 같습니다. 아마 아무리 착한 아이라도 세 번 이상 물으면 화낼 겁니다. 그래서 PHP에서는 여러 개의 구문을 하나의 그룹으로 묶어 주는 기능이 있습니다.

```
if ( $a > $b )
{
      echo "$a 은(는) $b 보다 크다.";
      echo "$a 와(과) $b 의 차는 " . ($a - $b) . "이다.";
}
```

위의 예처럼 중 괄호로 여러 개의 구문을 하나로 묶어 주면 PHP는 이 묶음을 처리합니다.

else

else 구문은 if 문에서 표현식의 결과가 FALSE이면 즉, 거짓인 경우에 처리될 부분을 정의합니다.

그러므로 else 구문은 if 문 없이 사용할 수 없습니다.

```
<?
    $a = 3;
    $b = 5;

    if ( $a > $b ) echo "$a 은(는) $b 보다 크다.";
    else echo "$a 은(는) $b 보다 작다.";
?>
```

else 구문을 이용하면 if 구문의 표현식이 참인 경우와 거짓인 경우 모두를 처리할 수 있습니다. 따라서 이 예제의 결과는 다음과 같이 출력됩니다.

```
3 은(는) 5 보다 작다.
```

if-else 구문 속에는 다시 if-else 구문이 들어갈 수 있습니다. 이를 통해서 다단계에 걸친 조건문 판단이 가능하고 여러 가지 경우에서 특정 조건에 맞는 코드를 수행할 수 있게 됩니다.

```
<?
    $a = 3;
    $b = 5;

    if ( $a > 0 && $b > 0) {
          if ( $a > $b ) echo "$a 은(는) $b 보다 크다.";
          else echo "$a 은(는) $b 보다 작다.";
   }
   else echo "두 변수 중 적어도 하나가 자연수가 아닙니다.";
?>
```

위의 예제는 변수 $a와 $b가 0보다 큰 자연수일 때 그 크기를 비교하는 코드이며 이처럼 if-else 구문은 중복적으로 사용할 수 있습니다.

elseif

초등학교에 다닐 때 누구나 한 번쯤 '수우미양가'로 평가된 등급 점수를 받아 보았을 것입니다. 이 평가방법은 90점 이상을 받으면 빼어나다는 의미로 수(秀), 80점 이상 90점 미만의 점수를 받으면 뛰어나다는 의미로 우(優), 70점 이상 80점 미만의 경우엔 아름답다는 의미로 미(美), 60점 이상 70점 미만의 경우엔 좋다는 의미로 양(良) 그리고 60점 미만의 점수를 받으면 가능성이 있다는 의미로 가(可)를 주는 방식입니다. 컴퓨터 프로그래밍에서 이 '수우미양가'를 판단하려면 다음과 같은 여러 단계에 걸친 조건 판단이 필요합니다.

① 90점 이상인가?

② 80점 이상 90점 미만인가?

③ 70점 이상 80점 미만인가?

④ 60점 이상 70점 미만인가?

⑤ 60점 미만인가?

만약 85점을 받았다고 하면, 사람은 금방 80점 이상 90점 미만이므로 우라고 판단하지만 컴퓨터는 위의 5단계에 걸쳐서 순서대로 판단해야만 합니다. 그래서 90점 이상인지를 판단하는 첫 번째 조건에서 "아니오."라는 답을 얻게 되고 두 번째 조건인 80점 이상 90점 미만인가를 묻는 조건에서 "예"라는 답을 얻게 됩니다. 두 번째 조건을 통해서 85점이 우가 됨을 알았으니 더 이상 3,4,5번째 조건은 수행할 필요가 없겠죠? 이처럼 여러 가지 조건을 순차적으로 비교하여 해당 조건에 맞을 때 그에 따른 코드를 수행하도록 하는 것이 elseif 구문입니다. 실제로 elseif 구문은 else if와 같습니다. 즉, else 다음에 바로 if 문이 오는 경우를 줄여서 elseif라고 한 것입니다. elseif 구문은 앞선 if 구문이 거짓이었고 다시 다른 조건을 판단하고자 할 때 사용할 수 있습니다. 이 구문을 통해서 if 구문을 무한히 확장할 수 있습니다.

```
<?
    $a = 3;
    $b = 3;

    if ( $a > $b ) echo "$a 은(는) $b 보다 크다.";
    elseif ($a == $b) echo "$a 은(는) $b 과(와) 같다.";
    else echo "$a 은(는) $b 보다 작다.";
?>
```

이 예제를 보면 아시겠지만 elseif 구문을 통해 기존 예제에 있던 조건문에 하나의 조건을 더 추가할 수 있습니다. 첫 번째 조건에서는 변수 $a가 $b보다 크지 않기 때문에 거짓이 되어 두 번째 조건인 서로 같은지 여부를 판단합니다. 변수 $a와 $b는 3으로 서로 같아서 참이 되고 결과는 다음과 같이 출력됩니다.

```
3 은(는) 3 과(와) 같다.
```

이를 응용하여 '수우미양가' 평가를 코드로 작성해 보면 다음과 같습니다.

```
<?
    $math = 85;

    if ( $math >= 90 ) echo "수";
    elseif ($math >= 80 && $math < 90) echo "우";
    elseif ($math >= 70 && $math < 80) echo "미";
```

```
    elseif ($math >= 60 && $math < 70) echo "양";
    else echo "가";
?>
```

그런데 이 코드는 다음과 같이 간소화될 수 있습니다.

```
<?
    $math = 85;

    if ( $math >= 90 ) echo "수";
    elseif ($math >= 80) echo "우";
    elseif ($math >= 70) echo "미";
    elseif ($math >= 60) echo "양";
    else echo "가";
?>
```

90점 미만, 80점 미만 등과 같은 항목을 지웠는데요. 그 이유는 앞선 조건이 거짓인 경우에만 다음 elseif 구문으로 넘어오기 때문에 다시금 조건을 검사할 이유가 없기 때문입니다. 즉, 첫 번째 조건에서 85점은 90점 이상이 아니므로 두 번째 조건 검사를 받습니다. 그런데 두 번째 조건 검사를 받을 수 있는 점수는 당연히 90점 미만임이 분명합니다. 만약 90점 이상이 존재한다면 첫 번째 조건을 통해서 걸러지지 않았다는 의미가 되기 때문입니다. 무의미한 조건 판단으로 성능을 저하시키지 않으려면 위와 같이 코딩하는 것이 좋습니다.

while

우리의 속담에는 '다람쥐 쳇바퀴 돌듯 한다.'는 말이 있습니다. 학창 시절뿐만이 아니라 사회에 나와서도 우리는 이와 비슷한 삶을 살아갑니다. 컴퓨터 프로그램도 마찬가지입니다. 컴퓨터 프로그램의 구조에서도 언급하였듯이 반복은 컴퓨터 프로그램의 중요한 구성요소 중 하나입니다.

while 문은 반복되는 일을 수행하고자 할 때 사용할 수 있는 가장 기본적인 형태의 제어 구조입니다. 이런 것을 루프(Loop)라고 말합니다. while 문은 다음과 같은 기본 구조를 가집니다.

```
while ( 표현식 ) 명령문;
```

while 문은 표현식이 참이면 명령문을 수행합니다. 표현식은 한 번만 검사하는 것이 아니라 한 번 반복할 때마다 참인지를 판단하고 참이면 명령문을 수행합니다. 만약 표현식이 거짓으로 판단되면 더는 반복하는 것을 멈추고 루프를 빠져나옵니다.

```
<?
  $i = 1;
  while ($i <= 10) {
```

```
    echo $i++;
  }
?>
```

위의 예제는 1부터 10까지를 순서대로 출력해주는 코드입니다. 변수 $i는 1부터 시작하기 때문에 10보다 작은 값으로 참이 되어 루프가 돌게 됩니다. 매번 명령문을 수행할 때마다 1씩 증가하므로 언젠가 변수 $i가 11이 되면 루프가 더는 수행되지 않게 될 것입니다. 여기서 주의할 점은 2부터가 아니라 1부터 출력된다는 것입니다. 증감 연산자에서 배웠듯이 변수 뒤에 "++"가 있으면 먼저 출력을 하고 그다음에 증가시키기 때문에 처음의 값은 1이 됩니다.

그런데 표현식이 늘 참이면 어떻게 될까요? 간혹 프로그래머의 의도나 실수 때문에 while 문의 표현식이 항상 참이 되는 경우가 있습니다. 이렇게 되면 루프는 무한히 반복되는데 이 경우를 무한루프라고 말합니다. 인간의 삶에서도 마찬가지지만 무한히 반복되는 삶은 사람뿐만 아니라 컴퓨터도 지치게 합니다. 다음과 같은 코드를 한번 살펴봅시다.

```
<?
  $i = 1;
  while (1) {
    echo $i++;
  }
?>
```

이 예제 프로그램은 while 문의 표현식이 항상 참이기 때문에 무한히 반복해서 숫자를 출력합니다. 그런데 이렇게 무한루프에 빠지게 되면 컴퓨터는 계속 반복하여 이 작업을 수행합니다. 컴퓨터는 최선을 다해서 매우 빠르게 루프를 반복해서 돌기 때문에 결국 얼마 있지 않아서 컴퓨터는 벅벅거리기 시작하고 웹 브라우저의 화면은 하얗게 멈춰 버리게 될 것입니다.

do-while

do-while 구문은 표현식을 검사하고 명령문을 수행하는 while과 달리 일단 명령문을 수행하고 계속해서 명령문을 수행할지를 결정하는 루프 제어 구조입니다. do-while 문에서 가장 유념할 점은 반드시 한 번은 수행된다는 것입니다. do-while 구문의 기본 구조는 다음과 같습니다.

```
do 명령문 while ( 표현식 );
```

명령문이 무조건 한 번은 실행되고 난 후 표현식을 검사한다는 차이를 제외하곤 while 문과 같습니다.

```
<?
    $i = 1;
```

```
    do {
        echo $i++;
    } while ( $i <= 10 );
?>
```

이 예제는 앞의 while 문의 예제와 같이 1부터 10까지 출력하는 코드입니다. while 구문과 차이를 확인하기 위해서 다음과 같이 변수 $i 값을 100으로 초기화해 봅시다.

```
<?
    $i = 100;
    do {
        echo $i++;
    } while ( $i <= 10 );
?>
```

기존의 while 문의 경우에는 이 같은 경우 루프에 들어가기 전에 표현식을 검사하기 때문에 어떠한 값도 출력되지 않습니다. 그러나 do-while 문에서는 우선 명령문을 실행하기 때문에 결과는 100이 출력됩니다.

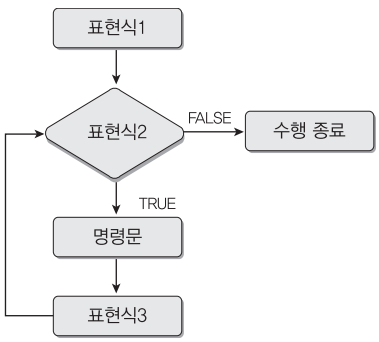
for

for 문은 특정 횟수만큼 해당 작업을 반복하고자 할 때 사용하는 제어 구조입니다. while 문과 for 문은 다른 표현으로 같은 작업을 수행할 수 있습니다. for 문의 기본 구조는 다음과 같습니다.

```
for ( 표현식1; 표현식2; 표현식3 ) 명령문
```

for 문은 제어 구조 중에서 가장 복잡한 형식을 갖습니다. for 문을 만나면 우선 표현식1을 수행합니다. 표현식1은 주로 for 문에서 사용될 변수들의 초기값을 설정하는 데 사용합니다. 그다음에는 표현식2를 통해서 참인지를 평가합니다. 만약 표현식2가 참이라면 명령문이 수행되고 명령문이 모두 수행되고 난 후에 표현식3이 수행됩니다. 표현식3은 주로 루프의 반복 횟수를 카운팅하는데 사용됩니다. 표현식3까지 처리가 되었다면 그다음은 다시금 표현식2를 통해서 반복 여부를 판단합니다. 만약 표현식2가 거짓이라면 루프를 빠져나옵니다.

for 문의 수행 절차를 정리해보면 다음과 같습니다.



[그림 2-22] for 문의 처리 순서도

예제를 통해서 for 문의 사용법을 알아봅시다.

```
<?
    for ( $i =1; $i <= 10; $i++ )
   {
        echo $i;
   }
?>
```

이 예제는 1부터 10까지 출력하는 코드입니다. 기존에 while 문을 통해서 구현한 코드와 비교해보면 숫자를 카운팅하는 변수 $i가 for 문의 구조 내에 표현되어 있기 때문에 명령문에는 간결하게 변수 $i를 출력하고 있습니다. 이처럼 특정 횟수만큼의 반복을 수행해야 할 때 for 문은 매우 유용하게 사용될 수 있습니다.

break

break는 while, do-while, for 등의 루프 제어 구조와 뒤에서 배울 switch 문의 수행을 멈추고 빠져나올 때 사용하는 제어문입니다. while 문처럼 루프를 반복해서 수행하면 특정 조건에서 루프의 수행을 중단해야 할 때가 있습니다.

while 문에서 무한루프의 예제를 다룬 적이 있습니다. 그 예제를 이용하여 다음과 같은 조건에서 무한루프를 빠져나오게 할 수 있습니다.

```
<?
    $i = 1;
    while (1) {
       if ($i > 10) break;
        echo $i++;
   }
?>
```

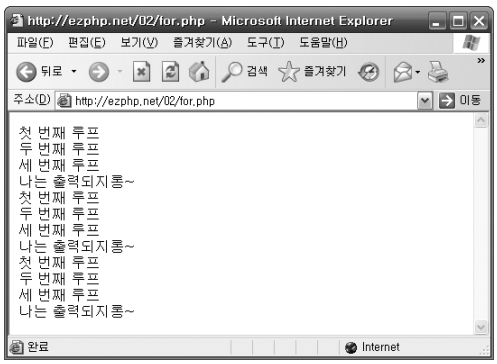
이 예제는 while 문의 표현식이 항상 참이기 때문에 계속해서 루프를 반복합니다. 그런데 명령문 블록 내부에 if 문을 사용하여 변수 $i가 10보다 크면 while 문의 수행을 종료하고 현재의 루프에서 빠져나오게 하기 때문에 더는 루프가 반복되지 않습니다. 이 예제의 결과는 1부터 10까지 출력하는데 주의할 점은 break 문을 만나면 break 문 이후의 명령문(echo 문)은 처리되지 않는다는 것입니다.

for 문이나 다른 루프에서도 이처럼 if 문을 사용하여 특정 조건에서 루프를 빠져나오게 할 수 있습니다. 때로는 다음 예제와 같이 루프가 중첩된 경우가 있습니다.

```
1 <?
2   for ( $i =0; $i < 3; $i++ )
3   {
4     echo "첫 번째 루프<br>";
5     while (1)
6     {
7       echo "두 번째 루프<br>";
8       while (1)
9       {
10        echo "세 번째 루프<br>";
11        break 2;          ①
12      }
13      echo "나도 출력되고 싶은데잉~<br>";   ②
14    }
15    echo "나는 출력되지롱~<br>";
16  }
17 ?>
```

이 예제는 3개의 루프가 중첩되어 있습니다. 세 번째 루프 속에 break 문이 있는데 break 뒤에 숫자가 쓰여 있습니다. 이 숫자는 중첩된 루프에서 얼마만큼의 깊이를 빠져 나오는 지 표시하는 것입니다. 숫자가 없는 경우에는 현재의 루프만을 빠져나오지만 2와 같이 숫자가 지정되면 2단계에 걸쳐서 루프를 빠져나옵니다. 그래서 세 번째 루프를 우선 빠져나오고 그다음에 두 번째 루프를 빠져나옵니다. 그러나 이미 두 단계에 걸쳐서 루프를 빠져나왔기 때문에 첫 번째 루프를 빠져나오지는 못합니다.

위와 같이 break 2를 통해서 매번 두 번째와 세 번째 루프가 종료되기 때문에 12번째 줄의 echo 문은 절대로 출력되지 않습니다. 그렇지만 첫 번째 루프의 영향으로 세 번의 반복이 발생하므로 이와 같은 일련의 작업이 세 번에 걸쳐서 나타나게 됩니다. 따라서 이 예제의 결과는 다음과 같이 출력됩니다.



[그림 2-23] break 문의 사용 예

만약 예제의 코드를 break 3으로 변경한다면 세 개의 모든 루프를 빠져나오게 되므로 반복은 일어나지 않습니다. 또한 **나는 출력되지롱~**의 문구도 출력되지 않습니다.

continue

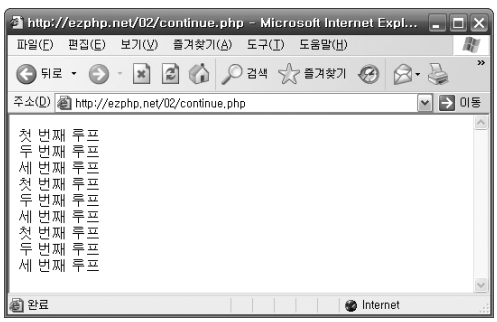
continue는 루프 구조에서 현재 루프의 남은 명령문을 생략하고 루프의 처음, 즉 표현식을 평가하는 부분으로 이동시켜주는 제어문입니다. break는 현재 루프의 남은 부분을 생략하고 루프 밖으로 빠져나오는 것에 반해서 continue는 루프를 빠져나오지 않는다는 차이점이 있습니다.

```
1 <?
2   for ( $i =0; $i < 3; $i++ )
3   {
4     echo "첫 번째 루프<br>";
5     while (1)
6     {
7       echo "두 번째 루프<br>";
8       while (1)
9       {
10        echo "세 번째 루프<br>";
11        continue 3;
12      }
13      echo "나도 출력되고 싶은데잉~<br>";
14    }
15    echo "나도 나오게 해줘 제발~ ㅜㅜ<br>";
16  }
17?>
```

③
②
①

이 예제는 break의 예제와 비슷한 유형으로 break와 continue의 차이를 보여주는 코드입니다. continue 또한 숫자가 없으면 현재 루프의 시작 부분으로 이동하고 만약 숫자가 있으면 해당 깊이만큼의 루프 시작 부분으로 이동합니다.

11번째 줄에서 "continue 3"을 만나고 3단계에 걸쳐서 루프의 시작 부분으로 되돌아가는 작업이 이루어집니다. 여기서 유의할 점은 첫 번째 루프인 for 문으로 되돌아왔을 때입니다. for 문으로 되돌아오면 for 구조의 세 번째 항목인 표현식3(여기서는 $i++)이 수행됩니다. 즉, 두 번째 항목인 참 거짓 평가보다 먼저 표현식3이 수행된다는 것입니다. 이는 다르게 생각하면 continue는 마치 해당 루프 블록의 마지막 부분으로 건너뛰는 것으로 생각할 수 있습니다. 루프의 마지막 부분에 도달했으니 표현식3을 수행하고 새롭게 루프를 반복해서 수행할지를 결정하는 것입니다. 이 예제의 결과는 다음과 같습니다.



[그림 2-24] continue 문의 사용 예

continue에 의해 항상 for 문의 처음으로 되돌아가기 때문에 13, 15번째 줄의 echo 문은 출력되지 않습니다.

switch

커피 자판기에서 판매하는 커피는 보통 일반형과 고급형 커피로 나눠지고 각각은 설탕과 프림의 첨가 여부에 따라서 밀크 커피, 프림 커피, 블랙 커피로 나누어 집니다. 이렇게 자판기에 총 6개의 선택 버튼이 있다고 하면 구매자가 선택 버튼을 눌렀을 때 자판기는 정확히 그 버튼에 해당하는 커피를 만들어야 합니다.

1번 버튼 – 고급형 밀크 커피 (고급형 커피 + 프림 + 설탕)

2번 버튼 – 고급형 프림 커피 (고급형 커피 + 프림)

2번 버튼 – 고급형 블랙 커피 (고급형 커피 + 설탕)

3번 버튼 – 일반형 밀크 커피 (일반형 커피 + 프림 + 설탕)

5번 버튼 – 일반형 프림 커피 (일반형 커피 + 프림)

6번 버튼 – 일반형 블랙 커피 (일반형 커피 + 설탕)

여기서 구매자가 선택한 버튼의 값을 변수라고 했을 때 입력된 변수에 따라 각각 해당하는 커피

를 만들어야 합니다. 이처럼 어떤 변수에 대해 그 값에 따라 각각 특정한 일을 처리하고자 할 때 switch 문이 유용하게 사용될 수 있습니다.

switch 문의 기본 구조는 다음과 같습니다.

```
switch ( 표현식 )
{
  case 값1:
  명령문;
  break;

  case 값2:
  명령문;
  break;

  default:
  명령문;
  break;
}
```

일부에서는 switch 문은 switch-case 문이라고 말합니다. 그 이유는 switch 문의 기본 구조에서 알 수 있듯이 switch와 case가 항상 함께 존재하기 때문입니다. switch 문을 만나면 표현식의 결과값 혹은 변수값에 해당하는 case 값을 찾습니다. 이 값은 서로 정확히 일치해야 하며 해당하는 case 값이 있다면 해당 명령문이 수행되고 break 문을 통해서 switch 구조 밖으로 빠져나옵니다. 만약 해당하는 case 값이 없다면 default 값이 수행됩니다. default 항목은 필수 항목이 아니므로 반드시 추가할 필요는 없지만 표현식을 통해서 예상치 못한 값이 들어오거나 case로 정의하지 않은 나머지 값에 대해서 동일하게 처리하고자 한다면 default 항목은 매우 유용할 수 있습니다. default 항목은 if 구문에서의 else와 비슷한 용도를 가집니다.

커피 자판기를 switch 문을 통해서 표현하면 다음과 같습니다.

```
<?
  $choice = 4;

  switch ($choice) {
  case 1:
          echo "고급형 밀크 커피";
          break;

  case 2:
          echo "고급형 프림 커피";
          break;
```

```
    case 3:
            echo "고급형 블랙 커피";
            break;

    case 4:
            echo "일반형 밀크 커피";
            break;

    case 5:
            echo "일반형 프림 커피";
            break;

    case 6:
            echo "일반형 블랙 커피";
            break;

      default :
      echo "일반형 밀크 커피";
    }
?>
```

변수 $choice 값이 4이고 해당하는 case 값이 존재하므로 **일반형 밀크 커피**가 출력될 것입니다. 만약 1에서부터 6까지의 값이 아니라 9와 같은 값이 들어온다면 해당하는 case 값이 없으므로 default 항목인 **일반형 밀크 커피**가 출력될 것입니다.

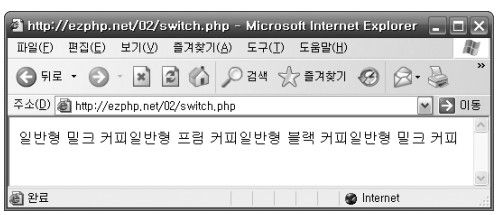
이 예제에서 주의할 점은 case와 case를 구분하기 위해서 중괄호를 사용하여 코드 블록을 만들지 않았다는 것입니다. 기존의 다른 제어 구조에서는 여러 개의 명령문을 수행하려면 반드시 중괄호를 이용하여 코드 블록을 만들어야 했습니다. 그런데 switch 문에서는 중괄호를 사용하지 않고 여러 개의 명령문을 수행하고 있습니다. case를 통해서 명확히 서로 구분할 수 있으므로 굳이 중괄호를 사용할 필요가 없어서입니다.

이에 덧붙여서 break의 용도에 대해서 알아봅시다. 위의 예제에서는 모든 case마다 마지막에 break 문이 꼭 들어 있지만 break 문은 switch 문에서 필수 항목이 아닙니다. 해당 case에 맞는 명령문을 모두 처리했으니 switch 문을 빠져나가라는 의미에서 사용된 것뿐입니다. 만약 break 문을 사용하지 않는다면 꽤 흥미로운 현상을 볼 수 있습니다.

```
<?
  $choice = 4;

  switch ($choice) {
  case 1:
          echo "고급형 밀크 커피";

  case 2:
```

```
            echo "고급형 프림 커피";

    case 3:
            echo "고급형 블랙 커피";

    case 4:
            echo "일반형 밀크 커피";

    case 5:
            echo "일반형 프림 커피";

    case 6:
            echo "일반형 블랙 커피";

      default :
      echo "일반형 밀크 커피";
    }
?>
```

이 예제의 결과는 흥미롭게도 다음과 같은 결과를 나타냅니다.



[그림 2-25] switch 문에서 break를 사용하지 않은 경우

의도는 **일반형 밀크 커피** 하나만 출력되는 것이지만 결과에서 보이듯이 세 가지 종류의 일반형 커피와 일반형 밀크 커피가 한 번 더 출력되었습니다. 그 이유는 switch 문에서는 break 문을 만날 때까지 다른 case에 대해서도 명령문을 수행하기 때문입니다. 그래서 $choice 값이 4이므로 case 4 이하의 모든 case와 default 항목에 대해서 처리가 된 것입니다. 이는 일종의 단점처럼 느껴지지만 다음과 같은 경우에는 매우 유용하게 사용될 수 있습니다.

```
<?
  $city = "수원";

  switch ($city) {
    case "수원":
    case "부천":
    case "고양":
    case "용인":
      echo "경기도";
```

```
        break;

      case "진주":
      case "창원":
      case "포항":
      case "경주":
        echo "경상도";
        break;
  }
?>
```

위의 예제에서처럼 여러 가지 값에 대해서 동일한 명령문의 수행이 필요한 경우 break 문을 적절히 사용하면 보다 효율적인 코드를 작성할 수 있습니다.

switch 문의 특성을 통해서 이미 눈치 챈 분도 있으리라 생각합니다. switch 문은 일종의 if-else 구문입니다. switch 문은 모든 경우에 if 문으로 만들 수 있습니다. 그러나 switch 문은 범위가 아니라 특정 값에 대해서 처리하기 때문에 범위를 사용하는 if 문의 경우에는 비효율적이거나 불가능할 수가 있습니다. 따라서 if 구문을 사용할 것인지 switch 문을 사용할 것인지는 용도에 따라서 편리한 것을 사용하는 것이 좋습니다.

include와 require

include와 require는 다른 파일을 가져와서 해당 위치에 파일의 내용을 적용하는 구조입니다. 이 두 구조는 동일하게 동작하며 차이점은 해당 파일이 존재하지 않는 경우 include는 경고를 출력하고 require는 에러를 발생시킵니다. 즉, include의 경우에는 경고만 출력하고 프로그램이 계속해서 수행되는데 반해서 require를 사용하면 에러가 발생하여 프로그램이 더는 실행되지 않습니다. include와 require의 기본 사용법은 다음과 같습니다.

```
include "/var/www/test.php";
include 'test.php';
include ('test.php');

$file = 'test.php';
include $file;
```

이처럼 파일의 위치를 지정하면 해당 파일을 읽어 와서 현재의 코드에 적용시킵니다. 다음의 예제를 통해서 동작원리를 알아봅시다.

head.html 파일

```
head.html 입니다.<BR>
```

tail.html 파일

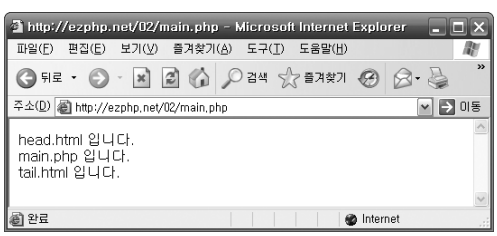
```
tail.html 입니다.<BR>
```

---

main.php 파일

```
<?
  include 'head.html';

  echo 'main.php 입니다.<BR>';

  include 'tail.html';
?>
```

위와 같이 head.html, tail.html 그리고 main.php가 있을 때 main.php 파일에서 나머지 두 파일을 include했습니다. 그 결과는 다음과 같습니다.



[그림 2-26] include 구조의 예

예제의 결과에서 보이듯이 이 세 개의 파일은 마치 main.php 파일 하나로 구성된 것과 같은 결과를 나타냅니다. 이번에는 PHP 파일을 include 해보겠습니다.

inc.php 파일

```
<?
  $name = '뇌를 자극하는 PHP 따라잡기';
?>
```

---

main2.php 파일

```
<?
  include 'inc.php';
  echo $name;
?>
```

main2.php 파일에서 PHP 문서인 inc.php 파일을 include했습니다. 그 결과는 다음과 같습니다.

[그림 2-27] PHP 파일을 include 한 결과

main2.php 파일에서는 변수 $name에 대한 어떠한 정의나 선언도 없었지만 올바른 값이 출력되는 것을 알 수 있습니다. 이는 include되는 inc.php 파일에 변수 $name이 존재하기 때문입니다. 이처럼 include를 통해서 PHP 문서를 include한 경우에도 마치 다음과 같이 해당 파일이 include 위치로 옮겨 온 것과 같은 결과를 얻을 수 있습니다.

```
<?
  $name = '뇌를 자극하는 PHP 따라잡기';
  echo $name;
?>
```

여기서 주의할 점은 include 문이 파일을 대체한다고 해서 include될 파일에서 〈? ?〉 태그를 지우면 안 됩니다. 만약에 PHP 시작과 종료 태그를 제거하면 PHP는 파일을 읽어 오면서 이 파일이 HTML 문서인지 아니면 PHP 문서인지를 판단할 수 없게 됩니다. 또한 PHP 문서에서 HTML 부분과 PHP 부분을 서로 구별할 수 없어서 올바르게 처리할 수가 없습니다.

include와 require는 매번 사용할 때마다 반복해서 파일을 가져옵니다. 만약 단 한 번만 파일을 가져오길 원한다면 include_once나 require_once를 사용할 수 있습니다. 이 두 구문은 각각 include와 require와 같으며 단지 한 번 처리되면 두 번 이상 처리하지 않는다는 것 이외에는 차이가 없습니다.

여기서 잠깐

include를 통해서 외부 서버의 파일을 가져올 수 있습니다. 예를 들어 include 'http://naver.com';이라고 한다면 네이버의 메인 화면을 볼 수 있습니다. 간혹 이 방법을 통해 다른 서버의 PHP 문서 코드를 훔칠 수 있지 않을까 생각할 수 있습니다. 그러나 외부의 파일의 경우에는 PHP 문서일지라도 HTML 문서로 인식하여 가져옵니다. 외부 서버 내에 있는 PHP 문서의 소스 코드는 접근할 수 없기 때문에 외부의 PHP 문서를 include하면 외부 서버의 웹 서버로부터 처리한 결과인 HTML 소스를 돌려받습니다. 그러나 원격 파일을 include하는 경우 웹 해킹의 가능성이 있기 때문에 보안을 위해서 php.ini 파일의 allow_url_fopen 옵션을 비활성화시키는 것이 좋습니다.

return

return은 현재 수행되고 있는 PHP 문서나 함수의 수행을 마치고 그 결과를 넘겨주는 구조입니다. 함수를 아직 다루지 않았기 때문에 함수 부분에서 다시 언급하도록 하겠습니다.

```
<?
    return;
    echo '나는 출력될 수 있을까?';
?>
```

return은 함수 외에서 사용될 때 해당 스크립트를 종료합니다. 따라서 위의 예제는 return의 등장으로 인해서 스크립트의 실행이 종료되어 더는 처리되지 않기 때문에 결과는 아무것도 출력되지 않습니다.

함수(function)

특정 역할을 하는 코드 블록을 빈번하게 사용할 때 매번 그 코드를 복사해서 사용한다면 코드의 길이도 길어질뿐만 아니라 여러모로 코드를 관리하기가 불편합니다. 예를 들어 숫자의 세 자리마다 콤마를 찍어 주는 10줄짜리 코드가 있다고 합시다. 만약 우리가 회계 프로그램을 작성해야 한다면, 이 10줄짜리 코드는 수없이 사용될 것입니다. 그런데 이 코드가 사용될 때마다 매번 다시 코드를 작성하거나 다른 곳에서 복사하여 사용한다면 프로그램을 작성하는데 효율이 무척이나 떨어집니다. 이때, 이 10줄짜리 코드에 '숫자에 쉼표 찍기'와 같은 이름을 붙여서 그 이름을 부르는 것만으로 처리할 수 있다면 어떨까요? 코드에 이름을 붙여주고 난 후에는 10줄 대신에 한 줄만 쓰면 되니까 매우 편리할 것이 분명합니다. 바로 이러한 이유에서 만들어진 것이 함수입니다.

함수에는 PHP에서 기본적으로 제공하는 내장 함수와 사용자가 편의에 따라 만들어서 사용하는 사용자 정의 함수가 있습니다. 내장 함수는 별도의 단원을 통해서 자세히 배우도록 하고 여기서는 사용자 정의 함수를 다루어 보겠습니다.

함수의 정의와 호출

사용자 함수를 정의하려면 다음과 같은 형식을 따라야 합니다.

```
function 함수이름 ($인자1, $인자2, $인자3 ... )
{
    명령문;
    return 반환값;
}
```

함수 이름은 변수 이름과 같은 식별자를 사용합니다. 이 함수 이름을 통해서 함수가 호출되기 때문에 함수의 이름은 그 역할을 충분히 표시할 수 있도록 정하는 것이 바람직합니다. 그리고 인자는 함수가 호출될 때 전달해 줄 값입니다. 숫자에 자릿수를 표시하려면 자릿수를 표시하고자 하는, 대상이 되는 수를 알려 주어야 합니다. 이처럼 함수의 처리 대상이 되거나 함수의 수행에 참조되는 값들을 함수 내부에서 사용할 수 있도록 전달하는 것이 바로 인자입니다. 변수의 범위에서 언급하였듯이 전역 변수가 아닌 경우에는 함수 내부에서 그 변수를 사용할 수 없으므로 함수 내부에서 참조해야 하는 변수는 모두 인자로 전달해야 합니다. 그 이유는 다른 코드에 영향을 받지 않고 함수를 독립적으로 수행할 수 있도록 하기 위함입니다. 함수 내에서 함수 밖에 존재하는 변수를 자주 참조할 경우에 그 함수는 다양한 소스 코드에 응용하기 힘들어 질 것입니다. 왜냐하면 함수 내부의 값뿐만이 아니라 함수 외부에 있는 변수들까지 고려해야만 올바르게 함수를 처리할 수 있기 때문입니다. 함수가 호출되면 명령문을 처리하고 그 결과를 return 문을 사용하여 되돌려 줍니다. 즉, 숫자에 자릿수를 표시한다면 쉼표로 구분된 숫자를 되돌려 주어야 할 것입니다. 이때 return 문을 사용할 수 있습니다. return 문이 나오면 함수의 나머지 명령문은 처리되지 않습니다. 만약 반환값이 없는 함수라면 return 문을 사용하지 않을 수 있습니다.

이처럼 함수를 정의하고 나면 다음과 같은 방법을 통해서 함수를 호출할 수 있습니다.

```
함수이름(인자값1, 인자값2, 인자값3 ...);
```

가장 단순한 형태의 함수를 통해서 함수의 사용법을 알아봅시다.

```
<?
  function sum($a, $b)
  {
    return $a+$b;
  }

  $result = sum(3, 4);
  echo $result . "<BR>";
  $result = sum(5, 7);
  echo $result . "<BR>";
?>
```

이 예제는 두 수를 더하는 기능을 하는 함수 sum()을 정의하고 sum() 함수를 호출하여 사용하는 법을 보여 주는 코드입니다. sum() 함수를 한 번 정의하고 난 후에 sum() 함수를 호출하는데 더하고자 하는 두 수를 인자로 넘겨주어 함수 내에서 덧셈을 처리할 수 있도록 합니다. 그리고 함수의 처리결과는 위와 같이 변수에 저장할 수 있습니다. 예제의 처리결과는 7과 12가 출력됩니다.

함수의 인자

PHP에는 함수의 인자를 넘겨주는 세 가지 방법이 있습니다.

① **값에 의한 인자 전달**

값에 의한 인자 전달 방법은 기본적인 방법으로 함수를 호출할 때 인자를 직접 값으로 전달하는 것을 말합니다. 좀 더 자세히 말씀드리면 인자로 숫자나 문자열 또는 배열을 전달하는 경우 그 값을 복사하여 복사본을 함수로 전달해 줍니다. 이는 워드를 잘 사용하는 친구에게 파일의 사본을 주면서 손을 봐 달라며 부탁하는 것과 같습니다. 친구는 전달받은 파일을 수정하여 되돌려 주지만 사본이 수정된 것이지 내가 가진 원본 자체가 수정되는 것은 아니라는 점을 유의해야 합니다.

```
<?
   function swap($a, $b)
   {
      $c = $a;
      $a = $b;
      $b = $c;
   }

   $num1 = 5;
   $num2 = 6;
   swap($num1, $num2);
   echo "$num1, $num2";
?>
```

이 예제는 두 변수값을 서로 교환하기 위해서 만든 swap 함수를 정의하고 있습니다. 프로그래머의 의도는 $num1이 6이 되고 $num2가 5가 되어 두 값이 서로 바뀌길 바랐으나 결과는 예상과 달리 변하지 않고 그대로의 값을 유지합니다. 즉, $num1은 5, $num2는 6의 값이 출력됩니다. 이것은 변수 $num1과 $num2가 전달된 것이 아니라 변수 $num1과 $num2의 값인 5와 6이 각각 인자 $a와 $b에 복사되었기 때문입니다. 그래서 $a와 $b의 값을 서로 바꾸더라도 $num1과 $num2의 값은 서로 바뀌지 않습니다.

② **참조에 의한 인자 전달**

참조에 의한 인자 전달 방법은 함수에 원본을 전달해주는 것과 같습니다. 원본을 인자로 전달해 주기 때문에 함수 내에서 그 값을 수정하면 원본의 내용까지 수정되는 것입니다. 참조에 의한 인자 전달 방법을 사용하려면 다음과 같이 함수의 정의에서 인자 앞에 엠퍼센트(&) 기호를 붙여주어야 합니다.

```
<?
   function swap(&$a, &$b)
   {
      $c = $a;
      $a = $b;
      $b = $c;
```

```
    }

    $num1 = 5;
    $num2 = 6;
    swap($num1, $num2);
    echo "$num1, $num2";
?>
```

이 예제는 값에 의한 전달 방법에 비해서 함수 정의에서 앰퍼센트 기호만 추가되었을 뿐입니다. 이 기호는 변수의 값이 아닌 변수 자체를 전달하라는 의미이기 때문에 함수 내에서 변경하는 것은 원본을 수정하는 것과 같습니다. 그래서 이 예제를 실행해보면 변수 $num1과 $num2의 값이 서로 바뀌어 있음을 알 수 있습니다.

③ 기본 인자값

기본 인자값은 함수의 인자에 기본값을 설정하는 것을 말합니다.

```
<?
    function ezphp_net ($url = "http://ezphp.net")
    {
        return "홈페이지 주소 : $url<BR>";
    }
    echo ezphp_net ();
    echo ezphp_net ("http://www.ezphp.net");
?>
```

이 예제와 같이 함수를 정의할 때 인자에 초기값을 부여하면 인자가 전달되지 않았을 때 이 값을 초기값으로 사용하게 됩니다. 만약 인자가 전달되면 기본 인자값은 무시되고 전달된 인자값이 사용됩니다. 따라서 이 예제의 결과는 다음과 같이 출력됩니다.

```
홈페이지 주소 : http://ezphp.net
홈페이지 주소 : http://www.ezphp.net
```

기본 인자값을 설정할 때 유의해야 할 점이 있습니다. 인자가 여러 개일 때 기본 인자값을 설정할 인자를 제일 마지막에 두어야 한다는 것입니다. 만약 처음 인자에 기본값을 지정한다면 에러를 보게 될 것입니다.

함수의 반환값

함수가 역할을 수행하고 난 후 그 결과를 반환해야 한다면 return을 통해서 결과값을 반환할 수 있습니다. 이때 인자의 전달과 마찬가지로 반환값도 두 가지 방법으로 되돌려 줄 수 있습니다.

① 값에 의한 반환

기본적인 방법으로 함수의 수행결과를 값으로 되돌려 줍니다.

```
<?
   function sum($a, $b)
   {
     return $a+$b;
   }

   $result = sum(3, 4);
   echo $result;
?>
```

기본적으로 함수는 하나의 값만을 반환할 수 있습니다. 그런데 간혹 함수를 통해서 여러 개의 값을 반환해야 하는 경우가 있는데 이럴 때는 배열을 이용할 수 있습니다.

```
<?
   function return_array()
   {
       return array(0, 1, 2);
   }
   $array = return_array();
   echo "$array[0], $array[1], $array[2]";
?>
```

이 예제의 결과는 0, 1, 2가 출력됩니다.

② 참조에 의한 반환

참조를 통해서 값을 반환받고자 할 때에는 함수의 정의와 호출 모두에서 앰퍼센트 기호를 사용해야 합니다.

```
<?

    function &func(){
        static $static = 0;
        $static++;
        return $static;
    }

    $var =& func();
    echo $var;
    func();
    func();
    echo $var;
    $var = 0;
    func();
    echo $var;
?>
```

이 예제와 같이 참조로 반환값을 전달하려면 함수의 정의에서 함수명 앞과 함수를 호출하여 반환값을 받을 때에 엠퍼센트 기호를 사용해야 합니다. 함수 func()는 변수의 범위에서 배웠듯이

정적 변수 $static을 사용하고 있기 때문에 함수가 호출되고 난 후에도 그 값이 사라지지 않습니다. 그래서 func() 함수가 호출될 때마다 정적 변수 $static의 값은 1씩 증가합니다. 그런데 이 정적 변수를 참조로 전달하면 변수 $var는 $static 변수와 같기 때문에 func()함수가 호출될 때마다 $var 값도 따라서 변경됩니다. 마찬가지로 변수 $var를 변경하면 정적 변수 $static도 따라서 변경됩니다. 그래서 이 예제의 결과는 131을 출력합니다.

함수의 사용

우리는 사용자가 정의한 함수를 어떻게 만들고 사용하는지에 대해 배웠습니다. 초보 프로그래머는 함수를 왜 써야 하는지 실감하지 못하기 때문에 주로 함수 없이 코딩하는 경우가 많습니다. 그러나 앞서도 말씀드렸듯이 제대로 잘 만들어 둔 함수는 두고두고 몇 번이고 쓸 수 있어서 코드의 양을 줄여 주고 심지어 다른 프로그램을 만들 때에도 유용하게 사용할 수 있습니다. 그래서 library.php와 같은 이름의 파일을 만들어 유용한 사용자 함수들을 정의해두고 필요한 경우 library.php 파일을 인클루드하여 사용합니다. 이 방법은 코드의 재사용성을 높여 주기 때문에 프로그램 개발의 속도를 높여 주고 또한 신뢰성 있는 프로그램을 작성하는데도 도움을 주게 됩니다. 따라서 처음에는 어렵더라도 함수를 하나씩 만들어서 사용하는 버릇을 기르는 것이 매우 중요합니다.

Chapter

03

폼 다루기

방명록이나 게시판 그리고 대부분의 웹 프로그램에서 폼을 사용합니다. 예전 초창기의 웹 페이지는 웹 서버 쪽에서 사용자에게 일방적으로 정보를 제공하는데 그쳤습니다. 그러나 CGI나 PHP와 같은 웹 프로그래밍 언어가 등장한 이후 폼은 사용자 쪽에서 서버에게 적극적으로 정보를 전달해주는 역할을 합니다. 폼은 파일과 파일 간에 정보를 전달하기 위한 매우 유용한 방법으로 HTTP 프로토콜에서 제공하는 GET이나 POST 방식을 사용하여 사용자가 입력한 값을 다른 파일로 전달합니다.

이 장에서는 폼을 이용하는 방법을 정리하여 사용자가 입력한 값을 정확하게 받아들이는 방법을 배워 보겠습니다.

Section 01

폼의 종류

폼을 사용하기 전에 먼저 폼에 어떤 것들이 있는지 알아봅시다.

〈FORM〉, 〈/FORM〉

〈FORM〉, 〈/FORM〉 태그는 폼의 시작과 끝을 나타냅니다. 뒤에서 다룰 각종 폼 컨트롤은 폼 태그 안에 있을 때만 정상적으로 동작합니다. 만약 폼 컨트롤이 폼 태그 외부에 존재한다면 해당 폼 컨트롤에 기록된 정보는 전달되지 않습니다. 폼 태그에는 여러 가지 속성이 있어서 이 속성들을 잘 설정해야 사용자가 입력한 정보를 올바르게 전달할 수 있습니다. 가장 일반적인 폼 태그의 사용방법은 다음과 같습니다.

```
<FORM NAME="폼이름" ACTION="데이터를 보낼 주소" METHOD="전송방식">
   여기에 HTML 및 기타 폼 요소들...
</FORM>
```

알다시피 HTML은 PHP와 달리 대문자와 소문자를 구분하지 않습니다. 따라서 소문자로 쓰거나 혼용해서 사용해도 무방합니다. 그러면 본격적으로 Form 태그의 속성과 이벤트 처리 함수를 들여다보도록 합시다.

속성	설명
name	폼의 이름
action	폼의 정보가 전달될 주소
method	폼 정보 전달방식 (GET 또는 POST)
enctype	폼 데이터의 인코딩 타입
target	폼 데이터의 처리 후 보여줄 프레임이나 창의 이름

[표 3-1] Form 태그의 속성

이벤트 핸들러	설명
onsubmit	submit을 실행하였을 때 이벤트 처리
onreset	reset을 실행하였을 때 이벤트 처리

[표 3-2] Form 태그의 이벤트 처리 함수

위의 표와 같이 폼에는 대표적으로 5가지의 속성과 2개의 이벤트 핸들러가 있습니다. 먼저 속성에 대해 살펴보면,

① NAME은 폼의 이름을 나타냅니다. 보통은 굳이 이름을 부여하지 않아도 상관없지만 하나의 HTML 문서 안에 여러 개의 폼이 있을 때 이를 구분 지으려고 폼의 이름을 사용합니다.

② ACTION은 사용자가 입력한 폼 정보를 전달할 주소를 지정하는 항목입니다. 데이터를 전달받는 주소는 반드시 PHP와 같은 서버 스크립트 파일이어야 합니다. 단순한 HTML에서는 폼 정보를 넘겨받을 수 있는 능력이 없기 때문입니다. 만약 ACTION 값을 지정하지 않으면 자신의 페이지로 정보를 전달합니다. 그러나 자기 자신이 아닌 다른 파일로 정보를 전달하고 싶다면 데이터를 전송받을 페이지를 반드시 알려 주어야겠지요?(자장면 시킬 때 주소 안 가르쳐 주면 배달 안 됩니다. ^^)

③ METHOD는 데이터를 전달하는 방식을 정의하는 부분입니다. 전송 방식에는 POST와 GET 두 가지 방식이 있으며 GET 방식은 URL을 통해서 전달하는 방식이고 POST 방식은 HTTP 바디(Body)에 데이터를 실어서 보내는 방식입니다. 이 부분은 다음 장에서 자세히 다루도록 하겠습니다.

④ ENCTYPE은 폼 데이터를 어떻게 인코딩(encoding)할 것인지를 지정하는 부분입니다. 그러나 폼 데이터 전송 형식이 GET이라면 ENCTYPE을 지정하더라도 원하는 결과를 얻을 수 없습니다. ENCTYPE은 POST 전송일 경우에만 적용되며 application/x-www-form-urlencoded와 multipart/form-data, 두 가지의 종류가 존재합니다. 우선 application/x-www-form-urlencoded의 경우 ENCTYPE을 지정하지 않았을 때의 기본값으로 폼 정보를 다음과 같이 URL 인코딩을 통해서 전송합니다.

[그림 3-1] 폼 입력 값의 인코딩 처리

만약 위의 그림과 같이 name 항목과 userid 항목을 입력하였을 때 웹 서버와의 통신을 엿보면 다음과 같이 전송되는 것을 확인할 수 있습니다.

```
<REQUEST-HEADER>
POST http://ezphp.net/form/form.php HTTP/1.0
Referer: http://ezphp.net/form/form.html
Accept-Language: ko
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 36

<REQUEST-BODY>
name=%BA%EA%B6%F3%BF%EE&userid=brown
```

헤더 정보에 폼 정보를 URL 인코딩을 통해서 전달하는 것을 표시하고 있고 Request Body에는 URL 인코딩된 값으로 폼 정보가 전달되는 것을 확인할 수 있습니다. 만약 Body의 내용을 urldecode 함수를 통해서 원상태로 되돌려보면 name=브라운&userid=brown이라는 값을 얻을 수 있을 것입니다.

이와 달리 multipart/form-data는 여러 가지 MIME 형식의 데이터를 전송하려는 방법으로 예를 들면 텍스트 정보와 GIF 이미지 정보를 동시에 전송하고자 할 때 사용하는 방식입니다. 이 방법은 바운더리(Boundary)를 설정하여 폼의 정보를 다음과 같은 방법으로 전송합니다.

```
<REQUEST-HEADER>
POST http://ezphp.net/form/form.php HTTP/1.0
Referer: http://ezphp.net/form/form.html
Accept-Language: ko
Content-Type: multipart/form-data; boundary=----------------------
-----7d729f1503da
Content-Length: 242

<REQUEST-BODY>
-----------------------------7d729f1503da
Content-Disposition: form-data; name="userid"

brown
-----------------------------7d729f1503da
Content-Disposition: form-data; name="userpw"

1234
-----------------------------7d729f1503da--
```

이처럼 "-----------------------------7d729f1503da"라는 임의의 문자열을 이용하여 각 항목을 구분 짓고 이를 통해 전송되는 값을 읽어들일 수 있습니다.

multipart/form-data를 사용하는 대표적인 경우가 바로 파일 업로드를 할 때입니다. 파일을 업로드할 때는 폼의 텍스트 정보와 GIF, JPG와 같은 이미지 또는 exe와 같은 실행 파일 등 여러 가지 MIME 타입을 갖는 데이터를 전송하기 때문입니다.

⑤ TARGET은 폼 정보를 모두 전송하고 그 결과를 보여 줄 프레임이나 창의 이름을 지정하는 부분입니다. 여기서 target은 하이퍼링크 태그인 〈a〉 태그에서 사용하는 target과 같은 것으로 생각하면 됩니다.

폼의 속성에 대해 알아보았으니 이제 이벤트 핸들러에 대해서 알아보겠습니다.

이벤트 핸들러(event handler)는 특정 이벤트가 발생했을 때 이를 감지하고 해당 이벤트를 처리할 수 있는 함수를 호출하거나 명령을 처리합니다. 폼 태그에서 사용할 수 있는 이벤트 핸들러는 OnSubmit과 OnReset이 있으며 명령어는 모두 자바스크립트로 처리됩니다.

① OnSubmit의 경우에는 submit(쿼리 전송) 버튼을 클릭하거나 자바스크립트를 이용하여 submit을 실행한 경우에 발생합니다. 사용자가 쿼리 전송 버튼을 클릭하는 경우 submit 이벤트가 발생하여 먼저 OnSubmit 핸들러에 지정된 명령어나 함수가 처리됩니다. 이 명령이 처리되고 ACTION에 지정된 주소로 폼 데이터를 전송합니다. 이 이벤트 핸들러를 사용하는 대표적인 경우는 사용자가 폼에 작성한 정보가 올바른지를 검사할 때입니다. 어떤 웹 사이트에 회원가입을 할 때 아이디의 글자 수 제한이라든지 사용자가 입력한 주민등록번호가 유효한지 등을 검사하는 경우가 많습니다. 이때 폼 정보를 페이지에 전송하기에 앞서 먼저 이 폼 값이 유효한지 검사를 하는 데 이 핸들러를 사용합니다.

② OnReset의 경우에는 reset(원래대로) 버튼을 클릭하거나 자바스크립트를 이용하여 reset을 실행한 경우에 발생합니다. 사용자가 리셋 버튼을 클릭하면 OnReset 핸들러에 지정된 함수나 명령어가 실행됩니다. OnReset 핸들러는 일반적으로 사용자의 입력에 의해서 설정된 값들을 초기화하기 위해서 사용합니다. 예를 들면 입력창을 동적으로 생성하여 입력한 경우 리셋 이벤트가 발생했을 때 동적으로 생성된 입력창들을 모두 제거해야 하는 경우가 발생합니다. 이때 OnRest 이벤트 핸들러를 통해서 이를 초기화해주는 자바스크립트를 작성할 수 있습니다.

폼 태그는 일반적으로 테이블(〈table〉)과 같이 사용하는 경우가 많습니다. 보통 다음과 같이 폼 태그와 테이블 태그를 사용합니다.

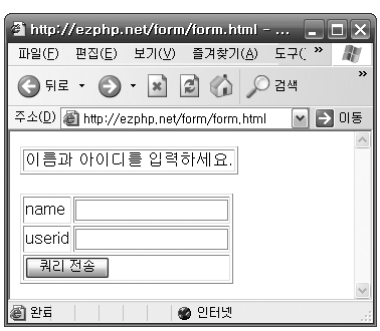
```
<FORM METHOD="POST" ACTION="form.php">
<TABLE border=1>
    <TR>
        <TD>name</TD>
        <TD><INPUT TYPE="TEXT" NAME="name"></TD>
    </TR>
    <TR>
        <TD>userid</TD>
        <TD><INPUT TYPE="TEXT" NAME="userid"></TD>
    </TR>
    <TR>
        <TD colspan=2><INPUT TYPE="SUBMIT"></TD>
    </TR>
</TABLE>
</FORM>
```

테이블이 하나일 때는 별 문제가 없지만 디자인 때문에 테이블이 여러 개면 폼 태그 때문에 테이블과 테이블 사이에 공백이 생깁니다.

[예제 3-1] 폼 태그로 인해 생기는 공백

```
1  <TABLE border=1>
2    <TR>
3      <TD>이름과 아이디를 입력하세요.</TD>
4    </TR>
5  </TABLE>
6  <FORM METHOD="POST" ACTION="form.php">
7  <TABLE border=1>
8    <TR>
9    <TD>name</TD>
10     <TD><INPUT TYPE="TEXT" NAME="name"></TD>
11   </TR>
12   <TR>
13     <TD>userid</TD>
14     <TD><INPUT TYPE="TEXT" NAME="userid"></TD>
15   </TR>
16   <TR>
17     <TD colspan=2><INPUT TYPE="SUBMIT"></TD>
18   </TR>
19 </TABLE>
20 </FORM>
```

[예제 3-1] 소스 코드를 웹 브라우저에서 확인해보면 다음과 같이 공백이 생기는 것을 알 수 있습니다.



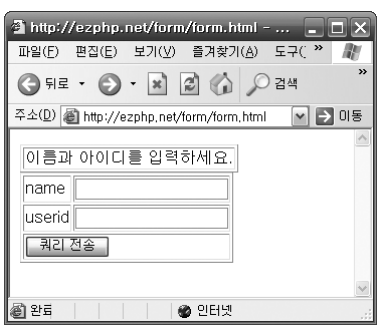
[그림 3-2] 폼 태그로 인한 공백

만약 테이블과 테이블 사이에 폼 태그가 없었다면 이와 같은 공백은 생기지 않습니다. 그래서 이러한 디자인적인 문제를 해결하기 위해서 다음과 같이 폼 태그를 테이블 태그의 안에 집어넣는 방법을 사용할 수 있습니다.

```
<TABLE border=1>
    <TR>
```

```
        <TD>이름과 아이디를 입력하세요.</TD>
    </TR>
</TABLE>
<TABLE border=1>
<FORM METHOD="POST" ACTION="form.php">
    <TR>
        <TD>name</TD>
        <TD><INPUT TYPE="TEXT" NAME="name"></TD>
    </TR>
    <TR>
        <TD>userid</TD>
        <TD><INPUT TYPE="TEXT" NAME="userid"></TD>
    </TR>
    <TR>
        <TD colspan=2><INPUT TYPE="SUBMIT"></TD>
    </TR>
</FORM>
</TABLE>
```

이렇게 〈TABLE〉 태그와 〈TR〉 태그 사이에 삽입하여 빈 공백이 발생하는 것을 막을 수 있습니다.



[그림 3-3] 폼 태그의 공백 제거

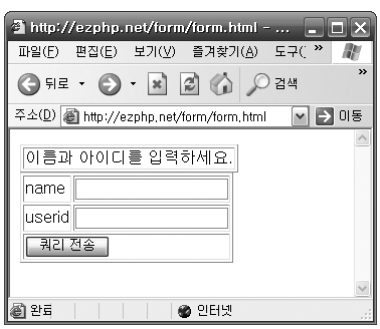
그러나 위처럼 반드시 특정 위치에만 넣어야 하는 제약이 있기 때문에 사용하는데 불편함이 있는데 스타일 시트를 이용하면 훨씬 편리하게 공백이 생기는 것을 막을 수 있습니다.

[예제 3-2] 스타일 시트를 이용한 공백 방지

```
1  <TABLE border=1>
2     <TR>
3        <TD>이름과 아이디를 입력하세요.</TD>
4     </TR>
5  </TABLE>
6  <FORM METHOD="POST" ACTION="form.php" STYLE="display:inline">
7  <TABLE border=1>
```

```
8     <TR>
9       <TD>name</TD>
10      <TD><INPUT TYPE="TEXT" NAME="name"></TD>
11    </TR>
12    <TR>
13      <TD>userid</TD>
14      <TD><INPUT TYPE="TEXT" NAME="userid"></TD>
15    </TR>
16    <TR>
17      <TD colspan=2><INPUT TYPE="SUBMIT"></TD>
18    </TR>
19 </TABLE>
20 </FORM>
```

이 방법은 폼 태그를 어디에 두든지 상관없이 스타일 시트를 이용하여 공백이 생기는 것을 원천적으로 차단할 수 있습니다.



[그림 3-4] 스타일 시트를 이용한 폼 태그의 공백 제거

〈INPUT〉

〈INPUT〉 태그는 각종 텍스트 값의 입력과 버튼 그리고 체크 상자와 라디오 상자를 만들 수 있습니다. 폼을 이용한 사용자 입력의 대부분을 차지하는 〈INPUT〉 태그는 타입을 설정하는 방법으로 여러 가지 모양의 입력 폼을 만들 수 있습니다.

TYPE	설명
TEXT	일반 텍스트 입력 상자
PASSWORD	비밀번호 입력 상자
FILE	파일 선택 상자
CHECKBOX	체크 상자

RADIO	라디오 버튼
BUTTON	일반 버튼
SUBMIT	쿼리 전송 버튼
RESET	초기화 버튼
IMAGE	이미지 버튼
HIDDEN	숨겨진 필드

[표 3-3] INPUT 태그의 종류

일반 텍스트 입력 상자

〈INPUT TYPE="TEXT"〉 태그는 짧은 텍스트를 입력할 수 있는 상자입니다. 길지 않은 문자열, 예를 들면 이름이나 이메일 주소와 같은 수십 자 이내의 값을 입력하고자 할 때 일반 텍스트 입력 상자를 사용합니다. 폼 입력 상자 중에서 가장 빈번히 사용됩니다.

[그림 3-5] 일반 텍스트 입력 상자

위의 그림에서 보는 바와 같이 문자열을 입력할 수 있게 상자가 만들어집니다. 일반 텍스트 입력 상자에 입력한 값을 전송하려면 이름을 설정해야 합니다.

```
<INPUT TYPE="TEXT" NAME="name">
```

위와 같이 이름을 설정하면 PHP 페이지에서 이 값을 $_GET['name'] 또는 $_POST['name']로 전달받을 수 있습니다. $_GET이냐 $_POST이냐는 〈FORM〉 태그의 METHOD 값이 GET이나 POST냐에 따라 결정됩니다.

name 이외에도 일반 텍스트 입력 상자에 사용할 수 있는 속성이 여럿 있습니다. 그중에서 가장 사용 빈도가 높은 것만 간추려 보면 value, size 그리고 maxlength를 들 수 있습니다. value는 입력 상자의 초기값을 지정해주는 기능을 하고, size는 입력 상자의 가로 길이를 조절하는 기능을 하며, maxlength는 입력 상자에 입력할 수 있는 최대 글자 수를 의미합니다. maxlength에서 영문자 하나와 한글 하나는 같이 한 자로 취급됩니다.

```
<INPUT TYPE="TEXT" NAME="name" SIZE="20" MAXLENGTH="5"
VALUE="0123456789">
```

위와 같이 태그를 작성하면 텍스트 입력창의 길이는 20이지만 최대 5자의 글자만 입력할 수 있습니

다. 만약 10자를 초과하면 더는 글자가 써지지 않는 것을 알 수 있습니다. 위 태그를 실행해보면 다음과 같습니다.

```
0123456789
```

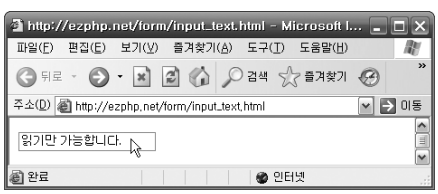
그런데 maxlength가 5임에도 10자가 출력되는 것을 확인할 수 있습니다. 이는 초기값의 경우 maxlength를 적용하지 않기 때문입니다. 이 경우 값을 8자로 수정하는 것은 가능하지만 8자로 수정한 값을 10자로 늘리는 것은 불가능합니다.

그렇다면 5자의 제한이 걸린 이 텍스트 입력 상자의 값은 제대로 전송이 될까요? 아니면 5자만 전송이 될까요? 정답은 "10자 모두 전송된다."입니다. maxlength 속성은 사용자의 입력을 원천적으로 봉쇄할 뿐이지 데이터 전송 길이를 제한하는 것이 아닙니다.

이외에 readonly라는 속성이 하나 더 있습니다. readonly 속성은 값을 입력하는 것이 아니라 태그에 다음과 같이 추가하면 적용됩니다.

```
<INPUT TYPE="TEXT" NAME="name" VALUE="읽기만 가능합니다." readonly>
```

위와 같이 readonly 속성을 추가하면 읽기만 가능한 입력 상자로 변경됩니다.



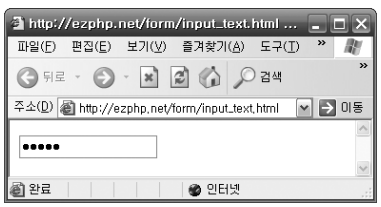
[그림 3-6] 읽기 전용의 텍스트 입력 상자

읽기 전용으로 만드는 경우는 사용자로 하여금 값을 입력하거나 변경하지 못하게 해야 하는 값을 전송하고자 할 때 주로 사용됩니다. 그러나 입력 값을 전달받는 PHP 페이지에서는 전달된 값이 읽기 전용으로 설정된 값인지 아닌지 판단할 수 없으므로 사용자가 쉽게 회피할 수 있습니다. 그래서 신뢰성이 매우 떨어지기 때문에 값을 악의적으로 수정한다고 해서 큰 문제가 생기지 않는 경우에만 사용하는 것이 좋습니다.

비밀번호 입력 상자

〈INPUT TYPE="PASSWORD"〉 태그는 일반 텍스트 입력 상자와 대부분 같지만 입력한 값이 별표(*)로 출력된다는 것이 다릅니다. 그리고 알파벳과 숫자 및 기호만 입력할 수 있어 한글로 "브라운"을 입력하게 되면 "qmfkdns"로 전달됩니다. 또한 입력된 값은 캐시에 저장되지 않기 때문에 뒤로

가기 버튼을 누르면 입력했던 값은 출력되지 않습니다. 따라서 이름에서도 알 수 있듯이 겉으로 드러나기를 꺼리는 암호나 주민등록번호 등을 입력할 때 많이 사용됩니다.



[그림 3-7] 비밀번호 입력 상자

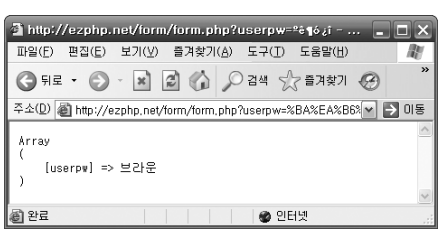
비밀번호 입력 상자는 데이터 전달 방식이나 사용할 수 있는 속성이 일반 텍스트 입력 상자와 모두 같습니다. 그래서 비밀번호 입력 상자에도 다음과 같이 초기값을 설정할 수 있습니다.

```
<INPUT TYPE="PASSWORD" VALUE="brown">
```

간혹 비밀번호를 겉으로 보이지 않는다고 위와 같이 초기값으로 보여주는 경우가 있습니다. 자동 로그인을 하게 하거나 주민등록번호를 보여 주고자 할 때 많이 사용합니다. 하지만 이것은 말 그대로 겉으로 별표로 보일 뿐 HTML 소스를 통해서 해당 값을 쉽게 볼 수 있으므로 될 수 있으면 초기값 사용을 자제하는 것이 좋습니다.

또한 비밀번호 입력 상자는 알파벳과 숫자 그리고 기호만 입력할 수 있는데 초기값을 한글로 설정하면 한글의 글자 수 대로 별표가 출력되고 그 값은 한글 값 그대로 전송됩니다. 따라서 비밀번호에서 초기값을 사용하는 경우 매우 주의 깊게 사용해야 합니다.

[그림 3-8] 비밀번호의 한글 초기값

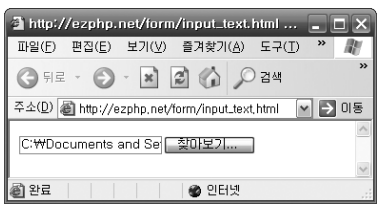


[그림 3-9] 전달된 한글 초기값

그리고 위의 예에서 보이듯이 비밀번호를 입력하게 할 때 GET 방식으로 전송하면 주소창에 비밀번호 정보가 그대로 드러납니다. 물론 URL 인코딩이 되어 기호로 보이지만 urldecode 함수를 통해 매우 쉽게 이 값을 확인할 수 있으므로 반드시 POST 방식으로 전송합니다.

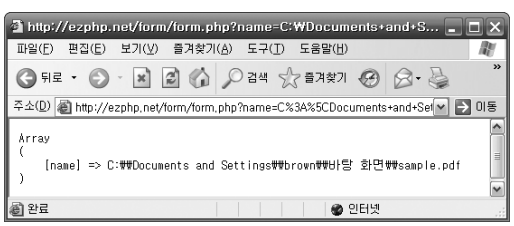
파일 선택 상자

〈INPUT TYPE="FILE"〉 태그는 파일을 업로드하거나 파일의 위치 정보를 기록하고자 할 때 사용할 수 있는 태그입니다. 파일 선택 상자를 클릭하면 웹 브라우저의 파일 탐색 브라우저가 띄워집니다. 파일 탐색 브라우저에서 선택한 파일은 텍스트 입력 상자에 파일의 주소가 자동으로 입력됩니다.



[그림 3-10] 파일 선택 상자

파일 선택 상자는 파일을 업로드할 때 주로 사용되며 POST 방식과 multipart/form-data 인코딩 형식을 지정해야 올바르게 동작합니다. 그렇지 않으면 단순히 일반 입력 상자와 같이 동작합니다.

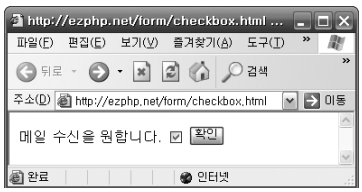


[그림 3-11] 인코딩 설정이 안된 경우 파일 입력 상자를 통해 전달된 값

체크 상자

〈INPUT TYPE="CHECKBOX"〉 태그는 각 항목에 대한 선택 여부를 확인하는 데 사용하는 태그입니다. 포털 사이트에 가입을 해보면 이메일 주소를 입력하고 "이메일을 수신하겠습니까?"라고 묻는 체크 상자를 볼 수 있습니다. 선택을 하면 지정된 값이 전송되는 방식으로 사용자로 하여금 해당 항목에 대한 〈예, 아니오〉로 대답할 수 있는 물음에 대한 응답을 받으려고 사용합니다.

```
<INPUT TYPE="CHECKBOX" NAME="is_recv_mail" VALUE="1">
```



[그림 3-12] 체크 상자

만약 체크 상자를 선택하면 $_GET['is_recv_mail']이나 $_POST['is_recv_mail']로 value에 지정된 값 1을 전달받을 수 있습니다. 그러나 체크 상자를 선택하지 않으면 값이 0이 전달되는 것이 아니라 변수 자체가 전달되지 않습니다.

체크 상자는 기본적으로 선택되지 않은 값으로 출력됩니다. 그러나 만약 기본적으로 선택되게 하고 싶다면 checked라는 속성을 사용하면 가능합니다.

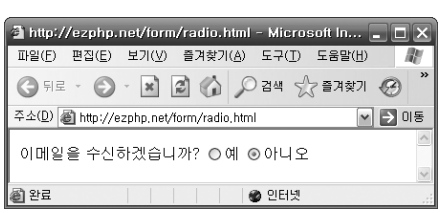
```
<INPUT TYPE="CHECKBOX" NAME="is_recv_mail" VALUE="1" checked>
```

라디오 상자

〈INPUT TYPE="RADIO"〉 태그는 체크 상자와는 달리 여러 개의 항목 중의 하나만을 선택해야 하는 경우 사용하는 태그입니다. 보통 2, 3개의 선택 항목 중에서 하나를 반드시 선택해야 하는 경우 사용합니다. 체크 상자와 마찬가지로 예와 아니오로 대답할 수 있는 질문에 사용하기도 하지만 중복으로 체크가 되지 않는다는 차이가 있습니다. 또한 일반적으로 4개 정도의 값에서 하나만을 선택해야 할 때에는 라디오 상자를 이용하지 않고 뒤에 다루게 되는 선택 상자(〈SELECT〉)를 사용합니다.

```
이메일을 수신하겠습니까?
<INPUT TYPE="RADIO" NAME="is_recv_mail" VALUE="1">예
<INPUT TYPE="RADIO" NAME="is_recv_mail" VALUE="0">아니오
```

위와 같이 사용할 수 있으며 중요한 것은 반드시 이름을 같게 지정해야 한다는 것입니다. 이름이 같지 않으면 각 항목을 다르게 인식하여 둘 다 선택이 가능해집니다.



[그림 3-13] 라디오 상자

실제로 위와 같이 실행해보면 예 혹은 아니오 둘 중에서 하나만 선택할 수 있는 것을 확인할 수 있

습니다. 둘 중에서 하나의 값을 선택하면 $_GET['is_recv_mail']이나 $_POST['is_recv_mail']로 선택된 값을 얻을 수 있습니다. 이 경우에는 예를 선택하면 "1"을, 아니오를 선택하면 "0"이란 값을 얻게 됩니다. 그런데 위와 같은 코드로 작성하면 기본적으로 어느 항목도 선택되지 않기 때문에 변수가 제대로 전달되지 않는 경우가 있습니다. 그래서 반드시 둘 중 하나의 값을 기본값으로 지정해야 합니다.

기본값을 지정하려면 앞서 체크 상자에서와 같이 checked 키워드를 사용하면 됩니다.

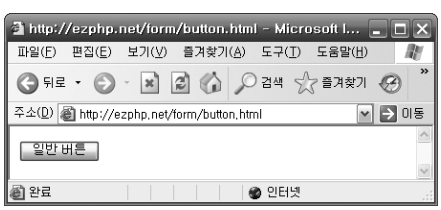
```
이메일을 수신하겠습니까?
<INPUT TYPE="RADIO" NAME="is_recv_mail" VALUE="1" checked>예
<INPUT TYPE="RADIO" NAME="is_recv_mail" VALUE="0">아니오
```

위와 같이 checked 키워드를 기본값에 설정해주면 웹 브라우저로 실행했을 때 기본적으로 해당 항목이 선택되어 있음을 확인할 수 있습니다.

일반 버튼

〈INPUT TYPE="BUTTON"〉 태그는 버튼 모양의 컨트롤을 만들어주는 태그입니다. 폼 컨트롤 중에서 버튼의 종류는 총 4가지입니다. 그중에서 단순히 버튼 모양만을 만들고 다른 기능이 없는 기본적인 태그가 바로 일반 버튼 태그입니다.

```
<INPUT TYPE="BUTTON" VALUE="일반 버튼">
```

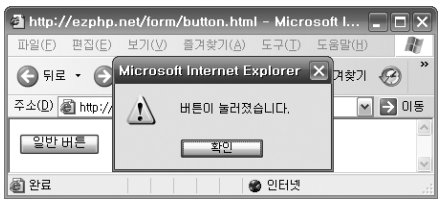


[그림 3-14] 일반 버튼

일반 버튼은 위의 그림처럼 VALUE 값으로 지정된 문구가 버튼에 출력됩니다. 하지만 아무런 기능이 없어서 클릭을 해도 변화가 전혀 없습니다. 그렇다면 왜 아무짝에 쓸모없는 태그가 있는 걸까요? 그 이유는 사용자로 하여금 원하는 기능을 부여하게 하기 위해서입니다. 즉, 프로그래머가 정의하는 기능을 하는 버튼이란 의미입니다.

일반 버튼의 기능을 정의하려면 자바스크립트를 이용해야 합니다. 버튼에서 사용할 수 있는 이벤트 핸들러를 사용하여 해당 이벤트가 발생했을 때 정의된 기능을 수행하게끔 하는 원리입니다. 일반적으로 버튼은 클릭했느냐 아니냐가 의미를 갖기 때문에 우선 클릭했을 때 클릭했음을 알려주는 경고창이 뜨도록 코드를 작성해 보겠습니다.

```
<INPUT TYPE="BUTTON" VALUE="일반 버튼" OnClick="alert('버튼이 눌러졌습니다.')">
```



[그림 3-15] 일반 버튼의 OnClick 핸들러

위의 그림처럼 버튼을 클릭하면 클릭 이벤트가 발생하여 OnClick 이벤트 핸들러에 정의된 스크립트가 실행됩니다. 스크립트를 어떻게 작성하느냐에 따라 무궁무진한 기능을 할 수 있는데 대체로 쿼리 전송(SUBMIT)을 하지 않고 어떤 기능을 처리해야 할 때 많이 사용합니다. 예를 들면 회원가입에서 사용되는 "아이디 중복 확인" 기능이라든지 "주소 입력" 기능에 사용할 수 있습니다. 이러한 기능들은 폼 정보를 채우기 위해서 도움이 되는 부수적인 동작들인 경우가 많습니다. 가장 일반적인 경우의 예를 한번 다루어 보겠습니다.

이름을 직접적으로 입력하지 않고 일반 버튼을 클릭하여 새로운 창을 띄운 다음 새로운 창에서 입력한 값이 원래 창의 이름 값에 등록되도록 합니다.

[예제 3-3] formpage.html

```
1  <script>
2    function insertName()
3    {
4      window.open("insertform.html","insertform",
5      "width=400,height=100");
6    }
7  </script>
8
9  <FORM NAME="insName" ACTION="form.php" METHOD="POST">
10   <INPUT TYPE="TEXT" NAME="uname">
11   <INPUT TYPE="BUTTON" VALUE="이름입력" OnClick="insertName();">
12   <INPUT TYPE="SUBMIT">
13 </FORM>
```

폼 페이지에서 이름 입력 버튼을 클릭하게 되면 insertName() 함수가 호출됩니다. insertName() 함수는 새로운 창을 띄우는 기능을 합니다.

```
window.open(주소, 창이름, 새창설정);
```

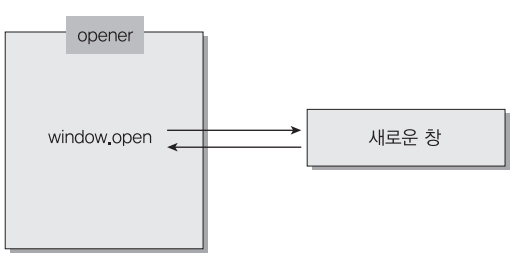
window.open을 통해서 새로운 창을 띄울 수 있으며 새로운 창에 띄워질 주소와 그 창의 이름 그리고 창에 대한 크기 등과 같은 설정 값을 지정하면 됩니다. 예를 들어 이름 입력 버튼을 여러 번 클릭하여 계속해서 새 창을 만들었을 경우와 같이 창의 이름이 같은 웹 브라우저가 있다면 새로운 창이 계속해서 생기는 것이 아니라 기존에 띄워진 창에 주소로 지정된 페이지가 불립니다. 또한 새로 띄워질 창의 모양에 대해서도 설정할 수 있는데 위의 [예제 3-3]에서는 창의 가로세로 길이를 지정했습니다.

[예제 3-4] insertform.html

```
1  <SCRIPT>
2  function insert()
3  {
4     var name = insertform.name.value;
5     opener.insName.uname.value = name;
6     self.close();
7  }
8  </SCRIPT>
9
10 <FORM NAME="insertform">
11 이름을 입력하세요.<BR>
12 <INPUT TYPE="TEXT" NAME="name"><INPUT TYPE="BUTTON" VALUE="입력"
13 OnClick="insert()">
14 </FORM>
```

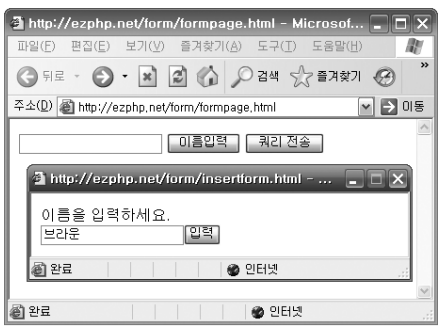
[예제 3-4]는 [예제 3-3]에서 버튼을 클릭하여 새 창에 보일 웹 페이지입니다. 일반적인 경우라면 주소를 검색하거나 중복된 이름이 있는지 검색하겠지만 편의를 위해서 곧바로 값을 입력받도록 하였습니다. [예제 3-4]에서도 마찬가지로 일반 버튼을 이용하여 insert() 함수를 호출합니다. insert() 함수는 입력한 이름값을 가져다가 원래 창에 다시 전송해주는 역할을 합니다. 그렇다면 어떻게 새 창에서 원래 창에다 정보를 보내 줄 수 있을까요?

opener는 새 창을 띄운 웹 브라우저를 지칭합니다. 즉, window.open을 사용한 창을 말하는 것입니다.



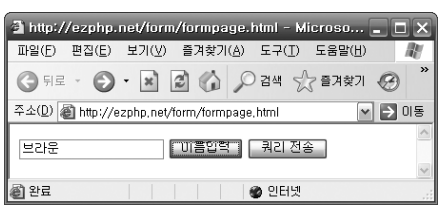
[그림 3-16] 기본 창과 팝업의 관계

"opener.폼이름.폼컨트롤이름" 과 같은 방법으로 자신을 오픈한 녀석의 값을 변경할 수 있습니다. opener의 값을 변경하고 난 후 자신(self)의 창을 닫게(close) 합니다.



[그림 3-17] 팝업을 통한 입력받기

〈이름 입력〉 버튼을 클릭하면 위와 같이 새로운 창이 하나 뜨게 됩니다. 여기에 이름을 입력하고 〈입력〉 버튼을 클릭하면 다음과 같이 입력한 값이 전달되는 것을 확인할 수 있습니다.



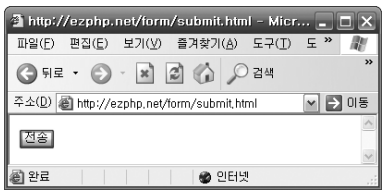
[그림 3-18] 팝업을 통해서 전달된 입력 값

쿼리 전송 버튼

〈INPUT TYPE="SUBMIT"〉 태그는 폼의 입력한 정보를 폼의 ACTION에 지정된 주소로 전송하는 버튼을 생성합니다. 폼의 정보를 전달하려면 반드시 SUBMIT(전송)이 필요합니다. 이 SUBMIT 기능을 하려면 자바스크립트를 이용한다든지 하는 몇 가지 방법이 있지만 가장 기본적인 방법이 바로 쿼리 전송 버튼입니다.

```
<INPUT TYPE="SUBMIT" VALUE="전송">
```

위와 같이 사용할 수 있으며 VALUE 값으로 지정된 문구가 버튼 모양에 출력됩니다.

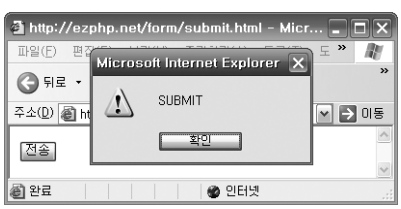


[그림 3-19] 쿼리 전송 버튼

쿼리 전송 버튼을 클릭하면 폼에서 SUBMIT 이벤트가 발생합니다. 따라서 OnSubmit 이벤트 핸들러를 사용하여 폼의 정보가 전송되기 이전에 사용자가 원하는 동작을 하게 만들 수 있습니다.

[예제 3-5] SUBMIT 이벤트 처리

```
1  <FORM METHOD="POST" OnSubmit="alert('SUBMIT')">
2  <INPUT TYPE="SUBMIT" VALUE="전송">
3  </FORM>
```



[그림 3-20] onSubmit 핸들러

위의 예제와 같이 폼의 정보에서 OnSubmit 이벤트 핸들러를 사용하여 입력 값을 검증하거나 폼 전송에 필요한 계산을 한다든지 하는 처리를 할 수 있습니다. 또한 쿼리 전송 버튼도 버튼의 일종이기 때문에 OnClick 이벤트 핸들러를 사용할 수 있습니다. 이를 이용하여 OnSubmit과 유사한 기능을 구현할 수 있지만 OnClick에 지정된 스크립트가 수행된 후에는 무조건 폼의 전송이 이루어진다는 것을 잊으면 안 됩니다. 즉, 사용자가 입력한 값을 검증해보았더니 규칙에 어긋나서 값을 전송하면 안 될 때에도 OnClick을 사용하는 경우 전송이 될 수 있다는 뜻입니다.

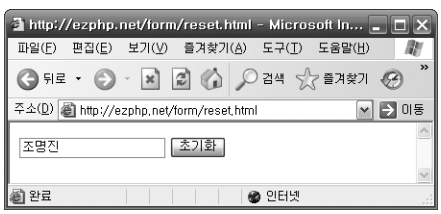
리셋 버튼

〈INPUT TYPE="RESET"〉 태그는 폼에 입력한 정보를 모두 원래의 초기값으로 되돌리는 버튼을 생성합니다. 기본적으로 빈 폼에 새로운 값을 입력했다면 모두 빈 폼으로 되돌아가게 되고 폼 컨트롤에 VALUE 속성 등을 이용하여 초기값으로 설정되어 있다면 해당 초기값으로 되돌아갑니다.

```
<INPUT TYPE="RESET" VALUE="초기화">
```

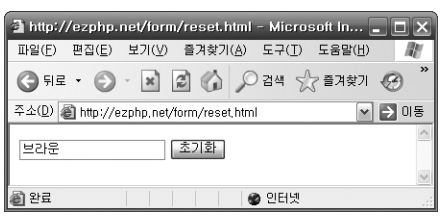
[예제 3-6] RESET 버튼

```
1  <FORM METHOD="POST">
2  <INPUT TYPE="TEXT" NAME="name" VALUE="브라운">
3  <INPUT TYPE="RESET" VALUE="초기화">
4  </FORM>
```



[그림 3-21] 리셋 버튼

[예제 3-6]에서 초기값으로 지정된 브라운을 위와 같이 수정하고 리셋 버튼을 클릭하면 다음과 같이 원래의 초기값으로 되돌아갑니다.

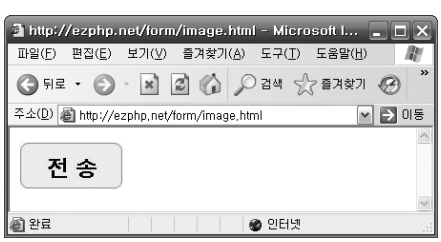


[그림 3-22] 리셋으로 인한 초기화

이미지 버튼

〈INPUT TYPE="IMAGE"〉 태그는 더욱 화려한 디자인을 추구하기 위해서 만들어진 사용자가 정의한 모양의 그림 이미지 버튼입니다. 이 이미지 버튼은 웹에서 사용 가능한 모든 이미지를 통해서 만들 수 있으며 쿼리 전송 버튼과 마찬가지로 버튼을 클릭하면 SUBMIT 이벤트가 발생합니다.

```
<INPUT TYPE="IMAGE" SRC="이미지파일">
```



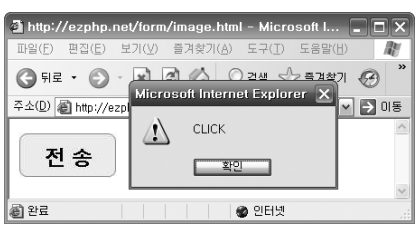
[그림 3-23] 이미지 버튼

다시 한번 유의할 점은 이미지 버튼은 쿼리 전송 버튼과 동일하게 SUBMIT 이벤트가 자동으로 발생한다는 것입니다. 그래서 이미지 버튼을 일반 버튼과 같이 사용하고자 할 때는 아래와 같이 OnClick 이벤트를 사용하고 SUBMIT 이벤트가 발생하지 않게 해주어야 합니다.

[예제 3-7] 이미지 버튼

```
1  <FORM METHOD="POST" OnSubmit="alert('SUBMIT')">
2  <INPUT TYPE="IMAGE" SRC="image.jpg" VALUE="전송"
3  onclick="alert('CLICK');return false;">
4  </FORM>
```

[예제 3-7]은 이미지 버튼을 클릭하면 "CLICK"이란 메시지를, SUBMIT 이벤트가 발생하면 "SUBMIT" 메시지를 출력하게 되어 있습니다. 여기서 유념해야 하는 것은 onclick 이벤트에서 alert() 뒷부분에 return false이라는 구문이 있다는 것인데 이 부분이 바로 SUBMIT 이벤트가 발생하지 않도록 막는 부분입니다. 이는 쿼리 전송 버튼에서도 동일하게 사용할 수 있습니다.



[그림 3-24] 이미지 버튼의 OnSubmit 핸들러 회피하기

[예제 3-7]을 실행해보면 CLICK 메시지 이후에 SUBMIT 메시지가 출력되지 않는 것을 확인할 수 있습니다.

숨겨진 상자

〈INPUT TYPE="HIDDEN"〉 태그는 이름에서도 알 수 있듯이 눈에 보이지 않는 숨겨진 상자를 만들어 줍니다. 이 숨겨진 상자는 일반적으로 사용자가 수정해서는 안 되는 값이나 눈에 보이지 않게 여러 가지 값을 전송할 때 사용합니다. 하지만 보안이 제대로 되지 않아 악의적으로 충분히 수정할 수 있기 때문에 숨겨진 상자를 신뢰해서 공개되어서는 안 되거나, 수정하면 큰 피해가 일어날 수 있는 중요한 정보를 전송하기 위해 사용해서는 안 됩니다.

```
<INPUT TYPE="HIDDEN" NAME="hName" VALUE="brown">
```

위와 같이 숨겨진 상자를 만든 경우에 일반 텍스트 입력 상자처럼 $_GET['hName'] 또는 $_

POST['hName']으로 값을 전달받을 수 있습니다.

〈SELECT〉 ... 〈/SELECT〉

이 태그는 일반적으로 3개 이상의 목록에서 하나 혹은 그 이상의 항목을 선택하고자 할 때 사용하는 폼 컨트롤입니다. 선택 상자에는 두 종류가 있는데 하나는 목록 중에서 하나만 선택하고자 하는 콤보 상자와, 다른 하나는 하나 혹은 그 이상을 복수로 선택할 수 있는 리스트 상자입니다.

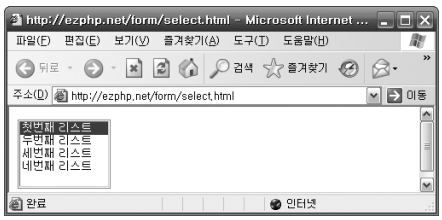
```
<SELECT name=select>
    <OPTION VALUE="1">첫번째 리스트</OPTION>
    <OPTION VALUE="2">두번째 리스트</OPTION>
    <OPTION VALUE="3">세번째 리스트</OPTION>
    <OPTION VALUE="4">네번째 리스트</OPTION>
</SELECT>
```

콤보 상자나 리스트 상자는 모두 동일하게 만들어지며 size 값을 지정하느냐 마느냐에 따라서 콤보 상자인지 리스트 상자인지 결정됩니다. 사이즈를 1이나 지정하지 않으면 기본적으로 콤보 상자로 보입니다. 콤보 상자로 출력하고자 할 때는 〈SELECT〉 태그에서 size를 지정하지 않으면 됩니다. 또한 콤보 상자는 목록 중에서 하나만 선택할 수 있습니다.

[그림 3-25] 콤보 상자

리스트 상자로 보여주고자 할 때는 다음과 같이 size를 지정해주면 됩니다.

```
<SELECT size=5 name=select>
    <OPTION VALUE="1">첫번째 리스트</OPTION>
    <OPTION VALUE="2">두번째 리스트</OPTION>
    <OPTION VALUE="3">세번째 리스트</OPTION>
    <OPTION VALUE="4">네번째 리스트</OPTION>
</SELECT>
```



[그림 3-26] 리스트 상자

만약 리스트 상자에서 다중 선택을 허용하는 경우에는 다음과 같이 multiple을 추가해주고 반드시 이름을 배열 형식으로 바꾸어야 합니다.

```
<SELECT size=5 name=select[] multiple>
    <OPTION VALUE="1">첫번째 리스트</OPTION>
    <OPTION VALUE="2">두번째 리스트</OPTION>
    <OPTION VALUE="3">세번째 리스트</OPTION>
    <OPTION VALUE="4">네번째 리스트</OPTION>
</SELECT>
```

[그림 3-27] 다중 선택 가능한 리스트 상자

콤보 상자와 리스트 상자 모두 $_GET['select'] 또는 $_POST['select']와 같이 선택된 값을 전달받을 수 있습니다. 그러나 리스트 상자에서 다중 선택을 가능하게 하면 선택된 여러 개의 값이 배열로 전달되기 때문에 $_GET['select'][0], $_GET['select'][1]과 같은 방법으로 전달받아야 합니다.

이 외에 콤보 상자와 리스트 상자에 상관없이 여러 항목 중에서 기본적인 선택 값을 지정하고자 한다면 〈OPTION〉 태그에 selected를 추가하면 됩니다.

```
<SELECT name=select>
    <OPTION VALUE="1">첫번째 리스트</OPTION>
    <OPTION VALUE="2">두번째 리스트</OPTION>
    <OPTION VALUE="3" selected>세번째 리스트</OPTION>
    <OPTION VALUE="4">네번째 리스트</OPTION>
</SELECT>
```

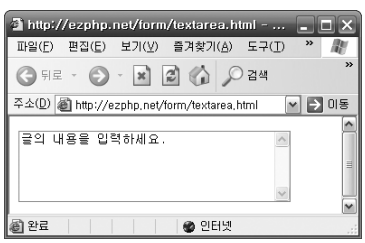
〈TEXTAREA〉 ... 〈/TEXTAREA〉

마지막으로 여러 문단에 걸친 긴 글을 입력하기 위한 〈TEXTAREA〉 태그에 대해서 알아보겠습니다. 일반적으로 게시판과 같은 곳에서 글의 내용을 입력하는 데 사용하며 입력창의 크기를 다음과 같은 방법으로 지정할 수 있습니다.

```
<TEXTAREA NAME="content" ROWS="5" COLS="40"><TEXTAREA>
```

여기서 ROWS는 세로의 크기를 나타내며 이 값이 5이면 다섯 줄을 보여 줄 수 있는 크기를 의미합니다. 또한 COLS는 가로의 길이를 나타내며 40이란 값은 영문을 기준으로 40자가 보이는 크기를 뜻합니다. 그리고 만약 다음과 같이 〈TEXTAREA〉 시작 태그와 끝 태그 사이에 문장을 넣으면 초기값으로 동작하게 됩니다.

```
<TEXTAREA NAME="content" ROWS="5" COLS="40">글의 내용을 입력하세요.
<TEXTAREA>
```



[그림 3-28] 긴 글 상자

긴 글 상자도 마찬가지로 $_GET['content']나 $_POST['content']로 값을 전달받을 수 있습니다.

Section 02

GET vs POST

GET과 POST 두 방식은 역할은 같지만 GET 방식이 URL로 정보를 전달하기 때문에 전달되는 값이 겉으로 드러난다는 것과, POST 방식은 REQUEST BODY를 통해서 전달되기 때문에 겉으로 값이 보이지 않는다는 것이 다릅니다.

GET 방식은 /index.php?name=brown과 같이 파일 이름 다음에 물음표(?) 기호를 사용하여 "?변수명=전달될 값"과 같은 방식으로 동작합니다. 만약 두 개 이상의 값을 전달하고자 할 때는 & 기호

를 통해서 ?name=brown&homepage=http://ezphp.net처럼 전달합니다. 이러한 특성 때문에 비밀번호와 같은 정보를 전달하면 비밀번호 정보가 그대로 노출되는 단점이 있습니다. 또한 URL을 통해서 전달되기 때문에 길이에 제한이 있습니다. 대략 2KB 미만의 정보를 전송할 때 GET 방식을 사용합니다.

반면 POST 방식은 REQUEST BODY를 통해서 전달되므로 겉으로 드러나지 않습니다. 또한 데이터의 길이에 대한 제약이 없어서 매우 큰 용량의 정보를 전송하고자 할 때도 사용할 수 있습니다. 그리고 폼 값과 함께 파일을 전송하는 등 여러 가지 형태의 데이터를 전송하고자 할 때도 POST를 사용해야만 전송할 수 있습니다. POST 방식이 GET 방식보다 신뢰성 있고 안전하지만 조작이 충분히 가능하기 때문에 무조건적인 신뢰를 해서는 안 됩니다.

여기서 잠깐

POST 방식으로 전달하는 데이터의 길이가 몇 MB인 경우 제대로 전송이 안되는 경우가 발생할 수 있습니다. 그 이유는 PHP 설정에서 POST 방식으로 전송할 수 있는 크기를 제한하고 있기 때문입니다. php.ini 파일에서 max_post_size 값을 늘려서 이 제한을 변경할 수 있습니다.

--

--

Chapter

04

함수 레퍼런스

4장에서는 PHP에서 제공하는 여러 가지 함수에 대해 알아보도록 합시다. PHP는 웹 프로그래밍에 필요한 많은 유용한 함수들을 미리 만들어 두었습니다. 이런 함수들은 PHP의 최대 장점이기도 합니다. 너무 많은 함수 때문에 오히려 혀를 내두를 분들도 계시겠지만 이러한 함수들은 PHP 신공에서 중요한 초식이 되므로 함수들을 많이 알면 알수록 내공이 상승하는 걸 체험할 수 있습니다. 그러나 수많은 초식 중에 별로 쓰지 않는 초식은 살~짝꿍 무시해주는 센스!!

PHP가 제공하는 함수들은 앞서 말씀드렸듯이 매우 많고 또한 많은 프로그래머에 의해 계속해서 추가되고 있습니다. 함수를 따로 공부할 필요는 거의 없습니다. PHP 제공 함수들은 여러 가지 분류로 잘 정리되어 있으므로 필요에 따라 찾아서 사용하면 됩니다. 예를 들어 "디렉토리 내의 파일 목록을 알고 싶은데 그런 기능을 하는 함수가 있을까?" 같은 생각이 떠오르면 함수 레퍼런스를 꺼내어 디렉토리와 파일 관련 부분을 뒤지면 간단히 해결할 수 있습니다.

만약 월급이나 예산을 출력할 때, 세 자리마다 쉼표를 찍어서 표시해야 한다면 숫자를 문자열로 인식하여 세 자리마다 쉼표를 추가해주는 함수를 구현해야 합니다. 대충 생각해봐도 루프를 통해서 자릿수를 계산하고 세 자리마다 쉼표를 추가하는 코드를 작성해야 할 것입니다. 그렇지만 PHP 함수를 잘 알고 있거나 "이런 프로그래머가 자주 쓸 법한 함수는 PHP 제공 함수 중에 있을지도 몰라"와 같은 생각을 한다면 함수 레퍼런스에서 금방 number_format()이라는 함수를 찾을 수 있을 것입니다.

PHP가 제공하는 모든 함수를 기억해둘 필요는 없습니다. PHP 프로그래밍을 하는데 있어서 자주 사용하는 함수들을 '이런 기능을 하는 함수가 있었던 것 같은데…' 하고 떠올릴 수 있을 정도로만 기억해 두어도 충분합니다. 정확한 함수 이름이 기억나지 않더라도 "그까짓 거 대충" 책에서 뒤져보면 되지 않겠습니까? 노련한 프로그래머는 모든 함수를 외우는 사람이 아니라 필요한 함수를 잘 찾는 사람입니다. 그래서 PHP 공식 홈페이지에는 다음과 같이 PHP에서 제공하는 함수들을 검색할 수 있는 기능을 제공하고 있습니다. 초보 프로그래머뿐만이 아니라 중급 이상의 프로그래머도 검색은 필수입니다. 이 책에서는 제한적인 함수를 다루고 있으므로 여기에서 다루지 못하는 많은 유용한 함수를 꼭 검색을 통해서 활용해보기 바랍니다.

[그림 4-1] PHP.NET의 함수 검색 기능

Section 01

날짜와 시간 함수

일정관리나 다이어리 그리고 단순한 오늘의 날짜, 현재 시간 등 PHP 프로그램을 작성할 때 정말 많이 쓰이는 것 중 하나가 바로 날짜와 시간입니다. 이 함수들은 PHP가 실행되고 있는 서버로부터 날짜와 시간의 정보를 가져올 때 사용할 수 있습니다. 여기서 '서버로부터'라는 말은 큰 의미가 있습니다.

필자가 경험한 이야기를 하나 들려 드리겠습니다. 윈도우를 새로 설치하고 윈도우 업데이트(http://windowsupdate.microsoft.com) 사이트에 들어가 보안 업데이트를 받으려고 했습니다. 그런데 웬일인지 보안 업데이트가 하나도 없다고 나오는 것이 아니겠습니까? 새로 접속을 해보기도 하고 심지어 재부팅도 해봤건만 변함이 없었습니다. 이런저런 시도 끝에 컴퓨터의 날짜가 무려 2년이나 느려져 있음을 알게 되었습니다. 날짜를 올바르게 수정했더니 보안 업데이트가 정상적으로 동작했습니다. 왜 이런 일이 벌어진 걸까요?

정답은 윈도우 업데이트 사이트가 서버 시간이 아닌 클라이언트의 시간을 체크했기 때문입니다. 오늘 날짜 이전에 보안 업데이트가 등록된 것이 있으면 목록을 출력하게 해두었는데 2년 전이라 보안 업데이트가 하나도 없었던 것이지요. 만약 서버 시간으로 처리했었더라면 저와 같은 불편을 겪은 사람이 없었으리라 생각합니다. (지금은 시간이 다를 경우 에러 메시지를 보여준답니다.)

웹 서버로 접속하는 많은 클라이언트는 각자 시간이 조금씩 다릅니다. 1분 느린 사람이 있을 수 있고 5분 빠른 사람이 있을 수 있습니다. 심지어는 연도가 다른 사람마저 있을지 모릅니다. 이 모든 사람에게 같은 날짜와 시간을 보여주려면 기준이 될 날짜와 시간이 필요합니다. 그래서 개인마다 차이가 나는 클라이언트 시계가 아닌 웹 서버의 시간이 기준이 됩니다. 그런데 기준이 되는 웹 서버의 시간이 잘못되었다면 더 큰 일이 아닐 수 없습니다. 이러한 이유로 대부분의 서버는 한국표준과학연구원과 같은 표준시간을 제공해주는 서버로부터 시간을 주기적으로 동기화하여 시간 오류를 최소화하고 있습니다.

함수 이름	기능
checkdate	주어진 날짜가 실제 존재하는 날짜인지 판단
date	날짜와 시간을 지정한 형식에 맞추어 반환
getdate	날짜와 시간을 배열로 반환
microtime	유닉스 형식의 시간으로 반환
mktime	주어진 시간을 유닉스 형식의 시간으로 반환
time	현재의 시간을 유닉스 형식의 시간으로 반환

[표 4-1] 날짜와 시간 함수

checkdate

[PHP 3, PHP 4, PHP 5]

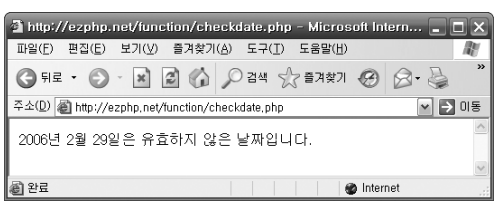
bool checkdate (int month, int day, int year)

주어진 날짜가 실제로 존재하는 날짜인지 검사하여 실제로 존재하면 TRUE를, 실제 존재하지 않는 날짜이면 FALSE를 반환한다.

checkdate() 함수는 연, 월, 일의 값을 입력받아 입력된 날짜가 실제로 존재하는 날짜인지를 검사합니다. 실제 존재하는 날짜인지의 판단은 그레고리력을 기준으로 합니다. 예를 들어 보통 2월의 마지막 날은 28일이지만 4년에 한 번씩 29일이 됩니다. 올해의 2월 29일이 유효한 날짜인지 확인하기 위해서 checkdate() 함수를 사용할 수 있습니다.

[예제 4-1] checkdate( ) 함수를 이용하여 날짜의 존재 여부를 검사한다.

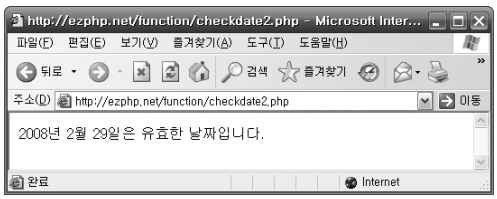
```
1  <?
2    $result = checkdate(2, 29, 2006);
3    if ($result) {
4      echo "2006년 2월 29일은 유효한 날짜입니다.";
5    } else {
6      echo "2006년 2월 29일은 유효하지 않은 날짜입니다.";
7    }
8  ?>
```



[그림 4-2] 예제 4-1의 실행결과

2006년 2월 29일은 실제로 28일까지이므로 29일은 유효하지 않은 날짜가 되어서 FALSE를 반환했습니다.

반면에 2008년 2월 29일은 실제로 존재하는 날짜입니다. [예제 4-1]을 수정하여 확인해보면 다음과 같습니다.



[그림 4-3] 예제 4-1을 이용하여 2008년 2월 29일을 검사한 결과

date

[PHP 3, PHP 4, PHP 5]

string date (string format [, int timestamp])

날짜와 시간을 주어진 형식에 맞추어 문자열로 반환한다.

인자	자료형	설명	비고
format	string	시간과 날짜의 반환 형식을 지정한다.	필수
timestamp	int	특정 시간을 지정한다.	옵션

[표 4-2] date 함수

date() 함수는 날짜와 시간을 특정 형식에 맞추어 문자열로 반환해 줍니다.

첫 번째 인자를 통해 형식을 정하고, 두 번째 인자를 통해 형식화할 시간을 지정할 수 있습니다. 만약 필수 인자인 형식만 지정하면 현재의 시간을 기준으로 형식에 맞춰 변형한 후 문자열로 반환합니다. 예를 들면 "2006년 1월 1일"이라는 날짜가 있을 때 이를 "2006-01-01"로 표현하고 싶다거나 "2006/1/1"로 표시하고 싶은 경우 date() 함수를 이용하여 변환할 수 있습니다. 이뿐만이 아니라 1월을 January로 나타낼 수도 있습니다. 여기서 날짜를 지정할 때 사용되는 두 번째 인자 타임스탬프(timestamp)는 유닉스 형식의 시간입니다. 유닉스 형식의 시간이란 1970년 1월 1일 0시 0분 0초를 시점으로 지정된 시간까지 몇 초가 흘렀는지를 나타냅니다. 예를 들어 2006년 1월 1일 0시 0분 0초의 타임스탬프는 1136041200입니다. 이는 기준 시점으로부터 1136041200초가 지난 후라는 것을 나타냅니다.

그런데 어떻게 "1월을 January로 표현하라"라고 알려줄 수 있을까요? 그것은 첫 번째 인자인 format에다 "F"라고 넣어주면 됩니다. [표 4-3]과 같이 다양한 형식의 날짜 표현 방식이 제공되고 있습니다.

분류	문자	설명	예
일	d	일자를 두 자리로 반환	01 ~ 31
	D	요일을 3글자로 반환	Mon ~ Sun
	j	일자를 0이 붙지 않는 형식으로 반환	1 ~ 31
	l	요일을 영문 단어로 반환	Sunday ~ Saturday
	S	영문 서수 접미사를 반환	st, nd, rd, th
	w	요일을 숫자로 반환	0(일) ~ 6(토)
	z	해당 연도의 몇 번째 날인지 반환	0 ~ 364
주	W	해당 연도의 몇 번째 주인지 반환	42 (연도의 42번째 주)
월	F	월을 영문 단어로 반환	January ~ December
	m	월을 두 자리 숫자로 반환	01 ~ 12
	M	월을 영문 3글자로 반환	Jan ~ Dec
	n	월을 0이 붙지 않는 숫자로 반환	1 ~ 12
	t	해당 월의 날짜 수를 반환	28 ~ 31
연도	L	윤년인지 여부를 반환	윤년이면 1, 아니면 0
	Y	연도를 네 자리 수로 반환	2006
	y	연도를 두 자리 수로 반환	06
시간	a	오전과 오후를 영문 소문자로 반환	am, pm
	A	오전과 오후를 영문 대문자로 반환	AM, PM
	g	시간을 12시간 형식으로 반환 0은 붙지 않는다.	1 ~ 12
	G	시간을 24시간 형식으로 반환 0은 붙지 않는다.	0 ~ 23
	h	시간을 12시간 두 자리 형식으로 반환	01 ~ 12
	H	시간을 24시간 두 자리 형식으로 반환	00 ~ 23
	i	분을 두 자리 형식으로 반환	00 ~ 59
	s	초를 두 자리로 반환	00 ~ 59
날짜 / 시간	c	ISO 8601 형식의 날짜로 반환	2006-01-01T12:00:00+09:00
	r	RFC 2822 형식의 날짜로 반환	Sun, 01 Jan 2006 12:00:00+09:00
	U	유닉스 형식 시간을 반환	1136041200 (2006년 1월 1일의 경우)

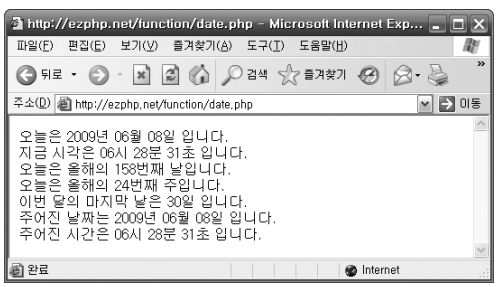
[표 4-3] date() 함수에서 사용할 수 있는 형식 문자

뭔가 기능이 참 많다고는 느껴지지만 어떻게 사용해야 하는지는 아직 감이 잡히질 않을 것입니다. 다음의 예제를 통해서 실제 사용법을 알아봅시다.

[예제 4-2] date() 함수를 이용하여 여러 가지 형식으로 출력해 본다.

```
1  <?
2      //날짜를 지정하지 않는 경우
```

```
3    echo date("오늘은 Y년 m월 d일 입니다.") . "<BR>";
4    echo date("지금 시각은 H시 i분 s초 입니다.") . "<BR>";
5    echo date("오늘은 올해의 z번째 날입니다.") . "<BR>";
6    echo date("오늘은 올해의 W번째 주입니다.") . "<BR>";
7    echo date("이번 달의 마지막 날은 t일 입니다.") . "<BR>";
8
9    /*
10   날짜를 지정하는 경우
11   mktime() 함수를 이용하여 timestamp를 만들 수 있습니다.
12   아래는 2009년 6월 8일 6시 28분 31초 입니다.
13   */
14   $date = mktime(6, 28, 31, 6, 8, 2009);
15   echo date("주어진 날짜는 Y년 m월 d일 입니다.", $date) . "<BR>";
16   echo date("주어진 시간은 H시 i분 s초 입니다.", $date) . "<BR>";
17 ?>
```

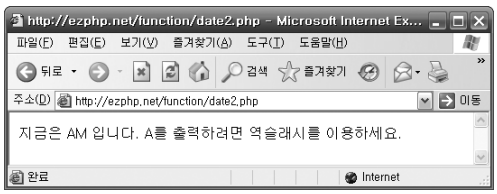


[그림 4-4] 예제 4-2의 실행결과

예제에서 볼 수 있듯이 형식 문자열에는 형식 문자뿐만이 아니라 한글과 같은 다른 문자열도 추가할 수 있습니다. 한글은 전혀 문제가 없는데 알파벳은 약간의 문제가 있습니다. 그저 "A"를 문자열에 추가하고 싶은데 A는 형식 문자이기 때문에 A가 반환되지 않고 오전 오후를 나타내는 AM이나 PM이 반환되어 버리기 때문입니다. 그래서 형식 문자로 지정된 문자를 출력하고 싶을 때는 역 슬래시(\)를 문자 앞에 붙이는 방법으로 출력할 수 있습니다.

[예제 4-3] 역 슬래시를 이용하여 형식 문자로 지정된 문자를 출력해본다.

```
1  <?
2    //A는 오전 오후를 나타내는 형식 문자
3    echo date("지금은 A 입니다. \A를 출력하려면 역슬래시를 이용하세요.");
4  ?>
```



[그림 4-5] 예제 4-3의 실행결과

getdate

[PHP 3, PHP 4, PHP 5]

array getdate ([int timestamp])

날짜와 시간을 주어진 형식에 맞추어 문자열로 반환한다.

인자	자료형	설명	비고
timestamp	int	특정 시간을 지정한다.	옵션

[표 4-4] getdate 함수

getdate() 함수는 날짜와 시간을 배열을 통해 반환해 줍니다.

getdate() 함수는 date() 함수와 유사하게 타임스탬프 값을 지정하는 경우에는 지정된 날짜에 대해서 날짜와 시간 정보를 배열에 담아 반환해 줍니다. 그렇지 않은 경우는 현재 시간을 기준으로 반환합니다. 반환된 배열은 다음과 같은 원소들로 구성됩니다.

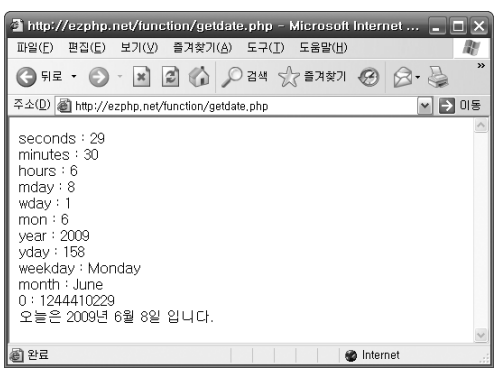
키	설명	예
"seconds"	초의 숫자 표현	0 ~ 59
"minutes"	분의 숫자 표현	0 ~ 59
"hours"	시간의 숫자 표현	0 ~ 23
"mday"	일의 숫자 표현	1 ~ 31
"wday"	요일의 숫자 표현	0 (일) ~ 6 (토)
"mon"	월의 숫자 표현	1 ~ 12
"year"	연도의 4자리 숫자 표현	2006
"yday"	해당 연도의 몇 번째 날인지 표현	0 ~ 364
"weekday"	요일의 영문 표현	Sunday ~ Saturday
"month"	월에 대한 영문 표현	January ~ December
0	유닉스 형식의 시간	1159102220

[표 4-5] 반환되는 연관 배열의 키 원소들

예제를 통해서 getdate() 함수의 사용법을 알아봅시다.

[예제 4-4] getdate() 함수를 이용하여 오늘의 날짜와 시간 정보를 배열로 얻는다.

```
1  <?
2    $today = getdate();
3
4    foreach ($today as $key => $value) {
5      echo $key . " : " . $value . "<BR>";
6    }
7
8    echo "오늘은 " . $today[year] . "년 " . $today[mon] . "월 "
9    . $today[mday] . "일 입니다.";
10 ?>
```



[그림 4-6] 예제 4-4의 실행결과

예제에서 볼 수 있듯이 배열의 키 인덱스를 이용하여 원하는 날짜 정보를 얻을 수 있습니다. 또한 앞서 말씀드린 것처럼 타임스탬프 값을 인자로 입력해주면 해당하는 타임스탬프에 대한 각각의 정보를 배열로 얻을 수 있습니다. 또한 배열의 원소를 출력하기 위해 foreach 구문을 사용하지 않고 print_r() 함수를 이용하여 간편하게 배열의 원소를 출력할 수 있습니다.

microtime

[PHP 3, PHP 4, PHP 5]

mixed microtime ([bool get_as_float])

현재의 시간을 유닉스 형식의 시간으로 100만분의 1초 단위까지 반환한다.

인자	자료형	설명	비고
get_as_float	bool	타임스탬프를 float 형으로 반환 (PHP5)	옵션

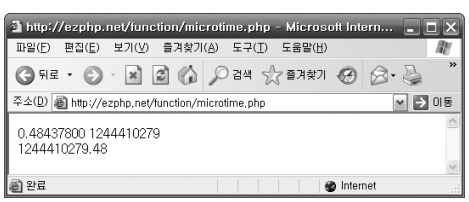
[표 4-6] microtime 함수

microtime() 함수는 현재의 시간을 유닉스 형식으로 100만분의 1초 단위까지 정밀한 값을 반환해 줍니다. 일반적으로 유닉스 형식의 시간은 초 단위까지 표시됩니다. 앞서 date() 함수에서 배웠듯이 유닉스 형식의 시간은 1970년 1월 1일 0시 0분 0초를 시작으로 몇 초가 흘렀는지를 나타내기 때문에 초 단위까지만 알려주었습니다. 하지만 microtime() 함수는 100만분의 1초까지 자세하게 알려주어서 주로 어떤 알고리즘을 처리하는 데 걸리는 시간을 측정한다든지, 웹 페이지를 모두 출력하는 데 걸리는 시간이 얼마인지 등을 측정하는 데 사용합니다.

기본적으로 microtime() 함수는 마이크로 단위의 정밀 값과 기존의 유닉스 형식의 시간 둘로 표시되는데 get_as_float 인자를 TRUE로 하면 둘이 합쳐져서 float 형의 시간을 반환합니다. get_as_float 인자는 PHP5에서 새로 생긴 것으로 이전 버전에서는 동작하지 않습니다.

[예제 4-5] microtime() 함수를 이용하여 타임스탬프를 마이크로 단위까지 출력한다.

```
1  <?
2    echo microtime();
3    echo "<BR>";
4    echo microtime(TRUE);
5  ?>
```



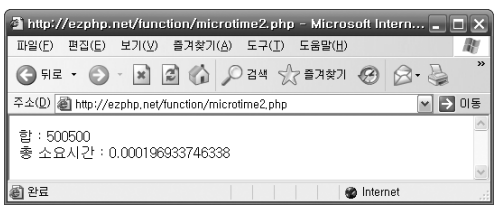
[그림 4-7] 예제 4-5의 실행결과

위의 결과는 왼쪽에 마이크로 단위의 시간 값이, 오른쪽에 기존의 유닉스 형식의 시간 값이 출력되었습니다. 아래의 결과는 둘의 결과가 합쳐진 것으로 시간이 약간 다른 것은 microtime() 함수를 두 번 호출하였기 때문에 그 사이에 0.001초의 시간이 흘러가서 생긴 차이입니다. 그러면 실제로는 어떻게 사용되는지를 알아봅시다.

[예제 4-6] microtime() 함수를 이용하여 1부터 1000까지의 합을 구하는 데 걸린 시간을 측정한다.

```
1  <?
2    $start_time = microtime(TRUE);
```

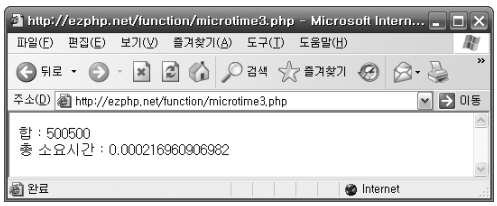
```
3     $sum = 0;
4
5     for ($i=1; $i <= 1000; $i++) {
6       $sum += $i;
7     }
8
9     $end_time = microtime(TRUE);
10
11    echo "합 : $sum <BR>";
12    echo "총 소요시간 : " . ($end_time - $start_time);
13  ?>
```



[그림 4-8] 예제 4-6의 실행결과

[예제 4-7] microtime() 함수의 get_as_float 인자를 사용하지 않고 속도를 측정한다.

```
1   <?
2     // microtime(TRUE)의 역할을 하는 함수
3     function microtime_float()
4     {
5       list($usec, $sec) = explode(" ", microtime());
6       return ((float)$usec + (float)$sec);
7     }
8
9     $start_time = microtime_float();
10
11    $sum = 0;
12    for ($i=1; $i <= 1000; $i++) {
13      $sum += $i;
14    }
15
16    $end_time = microtime_float();
17
18    echo "합 : $sum <BR>";
19    echo "총 소요시간 : " . ($end_time - $start_time);
20  ?>
```



[그림 4-9] 예제 4-7의 실행결과

mktime

[PHP 3, PHP 4, PHP 5]

int mktime ([int hour [, int minute [, int second [, int month [, int day [, int year [, int is_dst]]]]]]])

지정한 날짜와 시간을 유닉스 형식의 시간으로 반환한다.

인자	자료형	설명	비고
hour	int	시간을 지정한다	옵션
minute	int	분을 지정한다.	옵션
second	int	초를 지정한다.	옵션
month	int	월을 지정한다.	옵션
day	int	일자를 지정한다.	옵션
year	int	연도를 지정한다.	옵션
is_dst	int	일광 절약 시간제 사용 여부 (Summer Time 제도)	옵션

[표 4-7] mktime 함수

mktime() 함수는 현재의 시간이나 지정한 날짜와 시간을 유닉스 형식의 시간으로 반환해 줍니다.

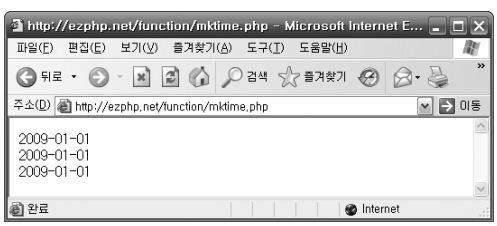
이름에서 알 수 있듯이 make time, 즉 시간을 만들어주는 함수입니다. 앞서 배운 date() 함수의 예제에 잠깐 출연한 적이 있는 함수로 원하는 날짜와 시간을 타임스탬프 값으로 얻고자 할 때 쓰는 아주 유용한 함수입니다.

인자들은 모두 옵션으로 인자를 입력하지 않으면 현재의 시간에 대한 타임스탬프를 반환하고, 인자를 입력하면 해당 날짜와 시간에 대한 타임스탬프를 반환합니다. 이때 주의해야 할 것이 인자의 순서입니다. 한국에서는 "연월일 시분초"와 같이 주로 쓰기 때문에 순서를 틀릴 때가 많습니다. "시분초"는 거의 틀리지 않으나 주로 "월일년"에서 틀리니 "일월년"이나 "연월일"로 쓰지 않게 주의하기 바랍니다.

마지막 인자는 일광 절약 제도(Daylight Saving Time)를 사용하는지에 대한 값으로 우리나라에서도 시행한 적이 있는 속칭 서머타임(Summer Time)제도를 말합니다. 우리나라와 유럽은 쓰지 않고 있으나 미국은 이 제도를 사용하고 있습니다.

[예제 4-8] mktime() 함수를 이용하여 특정 시간에 대한 타임스탬프 값을 구한다.

```
1  <?
2    echo date("Y-m-d", mktime(0, 0, 0, 12, 32, 2008)) . "<BR>";
3    echo date("Y-m-d", mktime(0, 0, 0, 13, 1, 2008)) . "<BR>";
4    echo date("Y-m-d", mktime(0, 0, 0, 1, 1, 2009)) . "<BR>";
5  ?>
```



[그림 4-10] 예제 4-8의 실행결과

[예제 4-8]은 mktime() 함수를 이용하여 타임스탬프 값을 다양한 방법으로 구하는 것을 보여줍니다. 주의할 점은 인자로 주어진 값이 비록 존재하지 않는 값일지라도 이를 계산하여 실제 존재하는 값으로 반환해 준다는 것입니다. 2번 줄을 확인해보면 2008년 12월 32일이라고 입력했지만 mktime() 함수는 2008년 12월 31일보다 하루 더 지난 날이라고 계산하여 2009년 1월 1일이라고 반환해 주었습니다. 3번 줄 또한 2008년 13월 1일이라고 입력했지만 2008년 12월 1일보다 한달 후의 값은 2009년 1월 1일이라고 반환했습니다.

time

[PHP 3, PHP 4, PHP 5]

int time (void)

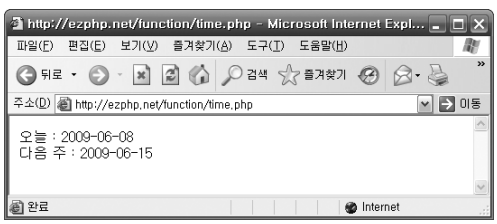
현재의 날짜와 시간을 유닉스 형식의 시간으로 반환한다.

time() 함수는 현재의 날짜와 시간을 유닉스 형식의 시간으로 반환해 줍니다. 앞서 배운 mktime()

함수에서 인자를 입력하지 않은 경우와 동일한 역할을 하며 현재 시간에 대한 타임스탬프 값을 구할 때 유용하게 사용할 수 있습니다.

[예제 4-9] time() 함수를 이용하여 다음 주의 날짜를 구한다.

```
1  <?
2    // 7 일; 24 시간; 60 분; 60 초
3    $next_week = time() + (7 * 24 * 60 * 60);
4
5    echo '오늘 : ' . date('Y-m-d') . "<BR>";
6    echo '다음 주 : ' . date('Y-m-d', $next_week);
7  ?>
```



[그림 4-11] 예제 4-9의 실행결과

3번 줄의 곱셈 연산은 7일에 대한 초 단위 시간을 나타냅니다. 즉, 60초(1분)가 60번 있으면 1시간, 1시간이 24번 있으면 1일, 1일이 7번 있으니 총 7일이라는 시간이 됩니다. time() 함수의 결과가 현재 시간의 타임스탬프 값이므로 여기에 7일이라는 시간을 더하면 오늘부터 7일 후인 날짜가 되는 것입니다.

Section 02 파일 시스템 함수

파일 시스템 함수는 리눅스 및 윈도우 시스템에서 파일과 디렉토리를 다루고자 제공하는 함수입니다. 파일과 디렉토리를 생성하고 이름을 바꾸는 등 파일과 관련된 기능 대부분을 함수로 제공하고 있으며 리눅스 시스템의 권한 설정과 소유자의 변경 등을 지원합니다. 특히 PHP가 웹 프로그래밍 언어이기 때문에 파일 업로드와 같은 웹을 위한 특별한 함수들도 지원하고 있습니다. 이러한 파일

시스템 함수들을 잘 이용하면 웹 기반 파일 서버나 웹 하드를 만들 수 있으며 이렇게 만들어진 웹 하드 프로그램은 ActiveX와 같은 외부 프로그램의 도움 없이도 완벽하게 동작합니다.

파일 시스템 함수는 서버의 파일을 직접 다룰 수 있게 해주기 때문에 사용에 매우 신중함을 기해야 합니다. 예를 들어 실수로 중요한 파일을 삭제하거나 내용을 덮어쓰게 될 수 있으며 특히 파일을 업로드할 때 제한 없이 업로드가 가능하도록 프로그램을 제작하면 크래커의 먹잇감이 될 것이 분명합니다. 파일 시스템 함수는 보안 이슈가 참 많은 함수이므로 이 함수를 사용할 때 안전한지를 반드시 확인하는 습관을 기르는 것이 좋습니다.

여기서 잠깐

리눅스 시스템에서 파일 시스템에 대한 접근 권한은 nobody 혹은 아파치 웹 서버의 실행 권한과 같습니다. 따라서 파일 시스템 함수를 이용하여 root와 같은 관리자나 다른 사용자의 권한 없는 파일을 삭제하거나 수정하는 등의 기능을 수행할 수 없습니다. 주로 읽기 권한이 있는 파일의 내용을 읽어오거나 모든 사용자에게 모든 권한이 부여된 파일들을 수정하고 삭제하는 데 사용합니다.

함수 이름	기능
basename	경로에서 파일 이름만을 반환
chmod	파일의 모드를 변경
copy	파일을 복사
dirname	경로에서 디렉토리 이름만 반환
fclose	열려 있는 파일 포인터를 종료
feof	파일 포인터가 파일의 끝에 있는지 검사
fgetc	파일로부터 하나의 문자를 가져옴
fgets	파일로부터 한 줄의 데이터를 가져옴
file_exists	파일이 존재하는지 여부를 확인
file	파일 전체를 배열로 읽어들임
filesize	파일의 크기를 반환
filetype	파일의 형식을 반환
fopen	파일이나 URL을 연다.
fpassthru	파일 포인터로부터 파일의 끝까지의 모든 데이터를 출력
fputs, fwrite	파일 포인터에 지정한 크기의 문자열을 쓴다.
fread	파일로부터 지정된 크기의 데이터를 읽음
is_dir	디렉토리인지 여부를 반환
is_file	파일인지 여부를 반환
is_uploaded_file	업로드된 파일인지 여부를 반환

mk_dir	디렉토리를 생성
move_uploaded_file	업로드된 파일을 지정한 위치로 이동
readfile	파일의 모든 데이터를 출력
rename	파일을 새 이름으로 변경
rmdir	디렉토리를 삭제
unlink	파일을 삭제

[표 4-8] 파일 시스템 함수

basename

[PHP 3, PHP 4, PHP 5]

string basename (string path [, string suffix])

경로명에서 파일의 이름만 반환한다.

인자	자료형	설명	비고
path	string	경로를 지정한다.	필수
suffix	string	파일 이름에서 제거될 접미사	옵션

[표 4-9] basename 함수

basename() 함수는 주어진 경로에서 파일의 이름만을 반환해 줍니다. 두 번째 인자인 접미사(suffix) 부분을 입력하면 해당 접미사 부분은 파일 이름에서 제거됩니다.

[예제 4-10] basename() 함수를 이용하여 파일의 이름을 구한다.

```
1  <?
2     $path = "/home/httpd/html/index.php";
3
4     echo basename ($path) . "<BR>";
5     echo basename ($path,".php");
6  ?>
```

[그림 4-12] 예제 4-10의 실행결과

chmod

[PHP 3, PHP 4, PHP 5]

bool chmod (string filename, int mode)

파일의 모드를 변환한다.

인자	자료형	설명	비고
filename	string	파일을 지정한다.	필수
mode	int	변환할 모드를 지정	필수

[표 4-10] chmod 함수

chmod() 함수는 주어진 파일의 모드를 변환해 줍니다. 유닉스 파일에는 읽기, 쓰기, 실행 세 가지의 권한이 있습니다. 파일에 대한 권한이 부여되어야만 읽기, 쓰기, 실행 행위를 할 수 있습니다. 파일의 소유자나 수퍼유저는 chmod() 함수를 이용하여 파일의 권한을 다음과 같은 방법으로 부여할 수 있습니다.

읽기 권한 = 4
쓰기 권한 = 2
실행 권한 = 1

이 세 가지 권한을 조합하여 파일에 권한을 부여할 수 있습니다. 파일에 읽기와 쓰기 권한을 주고자 한다면 읽기 + 쓰기 = 6과 같은 방법으로 사용합니다.

또한 파일의 권한은 소유자, 그룹, 모든 사용자별로 각각 권한을 부여할 수 있습니다. 순서는 소유자, 그룹, 모든 사용자의 순이며 666의 경우 소유자, 그룹, 모든 사용자가 파일을 읽고 쓸 수 있게 됩니다.

[예제 4-11] chmod() 함수를 이용한 파일의 모드 변경

```
1  <?
2    chmod ("/somedir/somefile", 0755);
3  ?>
```

0755의 0은 8진수임을 나타내는 것으로 0을 빼고 10진수인 755를 사용하면 제대로 동작하지 않을 수 있습니다. 따라서 8진수를 나타내는 0을 반드시 붙이도록 합시다.

755는 앞에서 설명 드린 것처럼 읽기 + 쓰기 + 실행 = 7, 읽기 + 실행 = 5입니다. 따라서 소유자는 모든 권한을 갖고 그룹과 다른 모든 사용자는 읽기와 실행 권한을 가지게 됩니다.

copy

[PHP 3, PHP 4, PHP 5]

int copy (string source, string dest)

파일을 복사한다.

인자	자료형	설명	비고
source	string	원본 파일의 이름	필수
dest	string	복사본 파일의 이름	필수

[표 4-11] copy 함수

copy() 함수는 주어진 파일을 복사하여 복사본의 파일을 만들어 줍니다. 파일의 복사본을 만드는 도중 복사에 실패하면 FALSE를 반환합니다.

[예제 4-12] copy() 함수를 이용하여 파일의 복사본을 생성한다.

```
1  <?
2    $file = "test.php";
3
4    if (copy($file, $file.'.bak')) {
5      echo "$file을 복사하였습니다...<br>\n";
6    } else {
7      echo "$file을 복사하는데 실패했습니다...<br>\n";
8    }
9  ?>
```

dirname

[PHP 3, PHP 4, PHP 5]

string dirname (string path)

경로에서 디렉토리 이름을 반환한다.

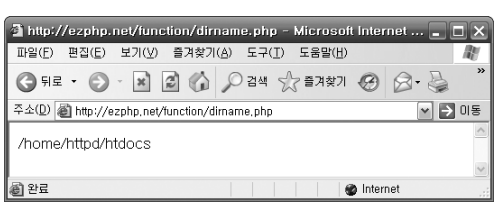
인자	자료형	설명	비고
path	string	경로 이름	필수

[표 4-12] dirname 함수

dirname() 함수는 어떤 파일에 대한 경로가 주어졌을 때 파일 이름을 제외한 디렉토리 이름을 반환합니다.

[예제 4-13] dirname() 함수를 이용하여 디렉토리 이름을 가져온다.

```
1  <?
2    $path = "/home/httpd/htdocs/index.php";
3    echo dirname ($path);
4  ?>
```



[그림 4-13] 예제 4-13의 실행결과

fclose

[PHP 3, PHP 4, PHP 5]

int fclose (resource handle)

열려 있는 파일 포인터를 닫는다.

인자	자료형	설명	비고
handle	resource	파일 포인터	필수

[표 4-13] fclose 함수

fclose() 함수는 열려 있는 파일 포인터를 닫습니다. fopen() 이나 fsockopen() 등에 의해 성공적으로 열린 파일 포인터를 fclose() 함수를 통하여 닫을 수 있습니다. 성공 여부에 따라 성공하면 TRUE를, 실패하면 FALSE를 반환합니다. fgetc() 함수의 예제를 통해 사용 방법을 확인할 수 있습니다.

feof

[PHP 3, PHP 4, PHP 5]

int feof (resource handle)

파일 포인터가 파일의 끝에 있는지 확인한다.

인자	자료형	설명	비고
handle	resource	파일 포인터	필수

[표 4-14] feof 함수

feof() 함수는 열려 있는 파일 포인터가 파일의 끝에 있는지 확인합니다. feof() 함수는 File End Of File의 약자로 파일을 읽을 때 파일의 끝까지 다 읽었는지를 확인하기 위해 사용하는 함수입니다. fgetc() 함수의 예제를 통해 사용 방법을 확인할 수 있습니다.

fgetc

[PHP 3, PHP 4, PHP 5]

string fgetc (resource handle)

파일 포인터로부터 하나의 문자를 가져온다.

인자	자료형	설명	비고
handle	resource	파일 포인터	필수

[표 4-15] fgetc 함수

fgetc() 함수는 열린 파일의 포인터에 있는 하나의 문자를 가져옵니다.

[예제 4-14] fgetc() 함수를 이용하여 파일을 출력한다.

```
1  <?
2    $fd = fopen ("somefile.txt", "r");
3
4    while (!feof ($fd)) {
5      $buffer = fgetc($fd);
6      echo $buffer;
7    }
8
9    fclose ($fd);
10 ?>
```

4번 줄에서 파일 포인터가 파일의 끝에 도달했는지를 확인하면서 5번 줄의 fgetc() 함수를 이용해 파일의 내용을 한 문자씩 가져오고 있습니다. 파일 전체의 내용을 출력할 때는 비효율적인 방법이 아닐 수 없습니다. 한 문자씩을 가져와야 하는 특별한 경우를 제외하고 파일의 데이터를 읽어들일 때에는 fgets() 함수를 사용하는 것이 더 좋습니다.

fgets

[PHP 3, PHP 4, PHP 5]

string fgets (resource handle [, int length])

파일 포인터로부터 지정된 길이의 문자열을 가져온다.

인자	자료형	설명	비고
handle	resource	파일 포인터	필수
length	int	한 번에 읽어 들일 크기	옵션

[표 4-16] fgets 함수

fgets() 함수는 열린 파일의 포인터로부터 지정된 길이만큼의 문자열을 가져옵니다.

fgets() 함수는 fgetc() 함수와 거의 유사하나 fgetc() 함수가 한 문자씩 읽어오는 반면 fgets() 함수는 지정된 크기의 문자열을 가져옵니다. 주로 fgets() 함수는 파일의 한 줄씩 가져오는 데 사용되는데 4096바이트의 길이를 지정하였더라도 개행문자(\n)를 만나면 개행문자까지의 문자열을 반환합니다. 두 번째 인자를 생략하는 경우 한 줄의 문자열을 반환합니다.

[예제 4-15] fgets() 함수를 이용하여 파일을 출력한다.

```
1  <?
2    $handle = @fopen("somefile.txt", "r");
3
4    if ($handle) {
5      while (!feof($handle)) {
6        $buffer = fgets($handle, 4096);
7        echo $buffer;
8      }
9      fclose($handle);
10   }
11 ?>
```

file_exists

[PHP 3, PHP 4, PHP 5]

bool file_exists (string filename)

파일 이름에 지정된 값이 실제로 존재하는지 검사한다.

인자	자료형	설명	비고
filename	string	파일 이름	필수

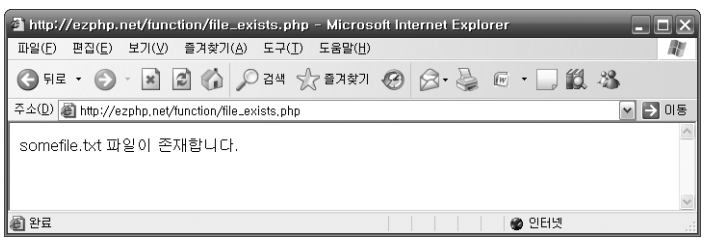
[표 4-17] file_exists 함수

file_exists() 함수는 지정된 값이 실제로 파일이나 디렉토리로 존재하는지를 검사합니다.

[예제 4-16] file_exists() 함수를 이용하여 파일이 실제로 존재하는지를 검사한다.

```
1  <?
2    $filename = 'somefile.txt';
```

```
3
4     if (file_exists($filename)) {
5       echo "$filename 파일이 존재합니다.";
6     } else {
7       echo "$filename 파일이 존재하지 않습니다.";
8     }
9   ?>
```



[그림 4-14] 예제 4-16의 실행결과

file

[PHP 3, PHP 4, PHP 5]

array file (string filename [, int use_include_path [, resource context]])

파일 전체를 배열로 반환한다.

인자	자료형	설명	비고
filename	string	파일 이름	필수
use_include_path	int	include path를 사용할지 여부	옵션
context	resource	stream context	옵션

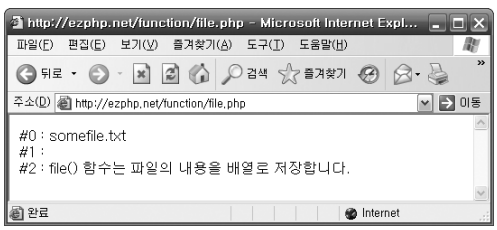
[표 4-18] file 함수

file() 함수는 지정된 파일의 모든 데이터를 한 줄씩 배열의 원소에 저장합니다. 마지막 개행문자까지도 배열에 저장되기 때문에 개행문자를 제거하기 위해 rtrim() 함수를 사용할 수 있습니다. 또한 파일의 모든 데이터를 단지 변수에 저장하고 싶을 때는 file() 함수를 쓰는 대신 file_get_contents() 함수를 이용하면 파일의 모든 데이터를 변수 하나에 저장할 수 있습니다.

두 번째 인자인 use_include_path 항목을 1로 입력하면 php.ini의 기본적인 설정인 include path 디렉토리 내에 지정된 파일이 존재하는지 검사하여 존재하면 파일을 읽어들여 배열로 반환합니다.

[예제 4-17] file() 함수를 이용하여 파일을 출력한다.

```
<?
  $array = file('somefile.txt');

  foreach ($array as $line_num => $line) {
    echo "#{$line_num} : $line <br>\n";
  }
?>
```

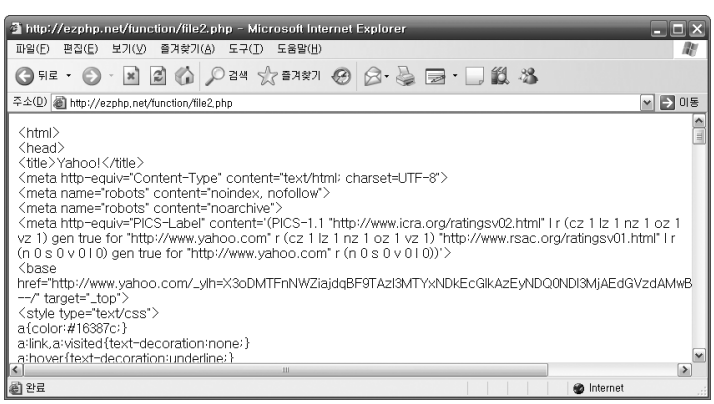


[그림 4-15] 예제 4-17의 실행결과

file() 함수의 첫 번째 인자는 비단 파일 뿐만이 아니라 URL을 지정할 수도 있습니다.

[예제 4-18] file() 함수를 이용하여 해당 URL의 HTML 소스를 출력한다.

```
<?
  $array = file('http://www.yahoo.com/');

  foreach ($array as $line_num => $line) {
    echo htmlspecialchars($line) . " <br>\n";
  }
?>
```



[그림 4-16] 예제 4-18의 실행결과

5번째 줄의 htmlspecialchars() 함수는 가져온 데이터가 HTML 태그를 포함하는 경우 해당 태그를 웹 브라우저가 인식해서 태그가 적용되는 것을 방지하기 위해 즉, 태그를 문자 그대로 출력하기 위해서 사용됩니다. 만약 5번째 줄을 echo $line으로 수정한다면 다음과 같이 출력됩니다.

[그림 4-17] file() 함수를 이용하여 가져온 yahoo.com의 데이터를 순수하게 출력한다.

filesize

[PHP 3, PHP 4, PHP 5]

int filesize (string filename)

파일의 크기를 반환한다.

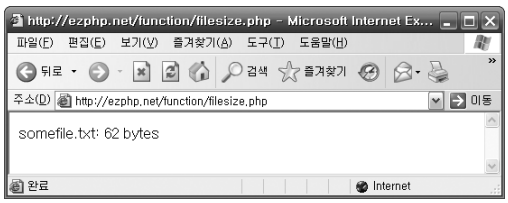
인자	자료형	설명	비고
filename	string	파일 이름	필수

[표 4-19] filesize 함수

filesize() 함수는 지정된 파일의 크기를 반환합니다.

[예제 4-19] filesize() 함수를 이용하여 파일의 크기를 출력한다.

```
1  <?
2    $filename = 'somefile.txt';
3    echo $filename . ': ' . filesize($filename) . ' bytes';
4  ?>
```



[그림 4-18] 예제 4-19의 실행결과

filetype

[PHP 3, PHP 4, PHP 5]

string filetype (string filename)

파일의 형식을 반환한다.

인자	자료형	설명	비고
filename	string	파일 이름	필수

[표 4-20] filetype 함수

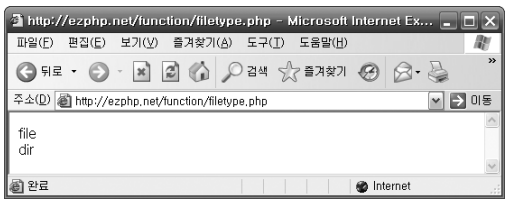
filetype() 함수는 지정된 파일의 형식을 반환합니다. 가능한 파일 형식은 다음과 같습니다.

파일의 형식	설명
block	block special device
char	character special device
dir	directory
fifo	FIFO (named pipe)
file	regular file
link	symbolic link
unknown	unknown file type

[표 4-21] 파일의 형식

[예제 4-20] filetype() 함수를 이용한 파일의 형식 출력

```
1  <?
2     echo filetype("/etc/passwd") . "<BR>";
3     echo filetype("/etc/");
4  ?>
```



[그림 4-19] 예제 4-20의 실행결과

fopen

[PHP 3, PHP 4, PHP 5]

resource fopen (string filename, string mode [, bool use_include_path [, resource zcontext]])

파일이나 URL을 연다.

인자	자료형	설명	비고
filename	string	파일 이름	필수
mode	string	파일 열기 모드	필수
use_include_path	bool	include path 사용 여부	옵션
zcontext	resource	stream context	

[표 4-22] fopen 함수

fopen() 함수는 지정된 파일이나 URL을 여러 가지 모드로 엽니다. 두 번째 인자인 mode를 통해서 파일에 대한 접근 유형을 설정합니다. 설정 가능한 유형은 다음과 같습니다.

모드	설명
r	읽기 전용으로 연다. 파일 포인터를 파일의 맨 앞에 놓는다.
r+	읽기, 쓰기가 가능하게 연다. 파일 포인터를 파일의 맨 앞에 놓는다.
w	쓰기 전용으로 연다. 파일의 포인터를 파일의 맨 앞에 놓는다. 파일의 크기를 0으로 만들고 만약에 파일이 없으면 새로이 만든다.
w+	읽기, 쓰기가 가능하게 연다. 파일 포인터를 파일의 맨 앞에 놓는다. 파일의 크기를 0으로 만들고 만약에 파일이 없으면 새로이 만든다.
a	쓰기 전용으로 연다. 파일 포인터를 파일의 끝에 놓는다. 만약에 파일이 없으면 새로이 만든다.
a+	읽기, 쓰기가 가능하게 연다. 파일 포인터를 파일의 끝에 놓는다. 만약에 파일이 없으면 새로이 만든다.

x	파일을 생성하고 쓰기 전용으로 연다. 파일 포인터는 파일의 맨 앞에 놓는다. 해당 파일이 이미 존재하면 FALSE를 반환한다.
x+	파일을 생성하고 읽기, 쓰기가 가능하게 연다. 파일 포인터는 파일의 맨 앞에 놓는다. 만약에 해당 파일이 이미 존재하면 FALSE를 반환한다.

[표 4-23] fopen() 함수에서 지정 가능한 mode

r과 r+는 파일을 읽어들인 다음 파일 포인터를 파일의 맨 앞으로 놓습니다. 파일을 읽어들이기만 하므로 파일이 존재하지 않으면 에러 메시지를 출력합니다. 또한 파일 포인터가 파일의 맨 앞에 놓이기 때문에 r+의 경우 파일에 데이터를 쓰면 기존의 데이터는 지워집니다.

w와 w+는 파일이 없는 경우에 새로 만들고 r, r+와 마찬가지로 파일 포인터를 파일의 맨 처음에 놓습니다.

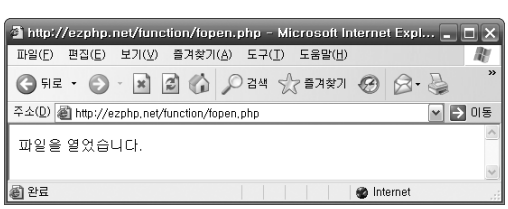
a와 a+는 파일이 없는 경우에 새로 만들지만 파일 포인터를 파일의 끝에 놓아 데이터를 추후에 쓰게 되더라도 기존의 데이터는 지워지지 않고 덧붙여집니다.

x와 x+는 파일을 새로 생성하고자 할 때 사용하며 기존에 파일이 존재하는 경우 에러 메시지를 출력합니다.

세 번째 인자인 use_include_path는 file() 함수와 마찬가지로 해당 파일을 include path에서도 찾을지를 선택하는 부분입니다.

[예제 4-21] fopen() 함수를 이용하여 파일을 연다.

```
1  <?
2    $handle = fopen("/home/httpd/file.txt", "r");
3
4    if ($handle) {
5      echo "파일을 열었습니다.";
6    } else {
7      echo "파일을 여는데 실패하였습니다.";
8    }
9
10   fclose($handle);
11 ?>
```



[그림 4-20] 예제 4-21의 실행결과

fpassthru

[PHP 3, PHP 4, PHP 5]

int fpassthru (resource handle)

파일 포인터에 남아있는 모든 데이터를 출력한다.

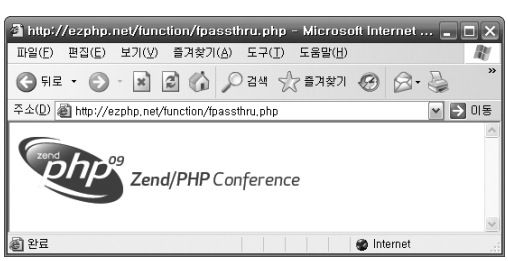
인자	자료형	설명	비고
handle	resource	파일 포인터	필수

[표 4-24] fpassthru 함수

fpassthru() 함수는 fopen()이나 fsockopen()으로 열린 파일 포인터의 위치로부터 파일의 끝까지의 모든 데이터를 출력합니다. fpassthru() 함수가 실패하면 FALSE를, 성공하면 출력한 데이터의 크기를 반환합니다. 주로 바이너리 파일을 보여주거나 해당 파일을 다운로드시키고자 할 때 사용하는 함수입니다.

[예제 4-22] fpassthru() 함수를 이용하여 이미지를 출력한다.

```
1  <?
2    $name = 'zend.png';
3    $fp = fopen($name, 'rb');
4
5    header("Content-Type: image/png");
6    header("Content-Length: " . filesize($name));
7
8    fpassthru($fp);
9    exit;
10 ?>
```



[그림 4-21] 예제 4-22의 실행결과

fputs, fwrite

[PHP 3, PHP 4, PHP 5]

int fwrite (resource handle, string string [, int length])

파일에 지정된 크기의 문자열을 기록한다.

인자	자료형	설명	비고
handle	resource	파일 포인터	필수
string	string	쓰일 문자열	필수
length	int	쓰일 문자열의 길이	옵션

[표 4-25] fputs, fwrite 함수

fputs와 fwrite는 같은 함수입니다. 이 함수는 첫 번째 인자로 지정된 파일 포인터에 두 번째 인자의 문자열 데이터를 세 번째 인자인 길이만큼 쓰는 함수입니다.

[예제 4-23] fwrite( ) 함수를 이용하여 파일에 문자열을 기록한다.

```
1  <?
2    $handle = fopen("/home/httpd/file.txt", "w");
3
4    $string = "뇌를 자극하는 PHP 프로그래밍";
5    fwrite($handle, $string);
6
7    fclose($handle);
8  ?>
```

fread

[PHP 3, PHP 4, PHP 5]

string fread (resource handle, int length)

파일로부터 지정된 크기의 데이터를 읽는다.

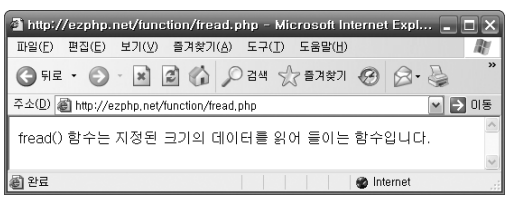
인자	자료형	설명	비고
handle	resource	파일 포인터	필수
length	int	읽어올 문자열의 길이	필수

[표 4-26] fread 함수

fread() 함수는 파일로부터 지정된 크기의 데이터를 읽어들이는 함수입니다. 두 번째 인자인 length 길이를 통해 파일 포인터에서 해당 길이만큼의 데이터를 읽어들입니다.

[예제 4-24] fread() 함수를 이용하여 파일의 데이터를 읽어 들인다.

```
1  <?
2     $filename = "something.txt";
3
4     $handle = fopen($filename, "r");
5     $contents = fread($handle, filesize($filename));
6
7     fclose($handle);
8  ?>
```



[그림 4-22] 예제 4-24의 실행결과

is_dir

[PHP 3, PHP 4, PHP 5]

bool is_dir (string filename)

입력된 인자가 디렉토리인지 확인한다.

인자	자료형	설명	비고
filename	string	파일 이름	필수

[표 4-27] is_dir 함수

is_dir() 함수는 filename이 실제로 존재하고 또한 디렉토리인지 검사한 결과를 반환하는 함수입니다. filename이 실제로 존재하지 않거나 파일이면 FALSE를 반환합니다.

[예제 4-25] is_dir() 함수를 이용하여 디렉토리인지 검사한다.

```
1  <?
2    $filename = "something.txt";
3
4    if (is_dir($filename)) {
5      echo "$filename은 디렉토리 입니다.";
6    } else {
7      echo "$filename은 파일이거나 존재하지 않습니다.";
8    }
9  ?>
```

[그림 4-23] 예제 4-25의 실행결과

is_dir() 함수는 실제로 존재하면서 디렉토리인 경우가 아니면 모두 FALSE를 반환합니다. 따라서 실제 filename의 값이 존재하지 않는 경우와 존재하지만 디렉토리가 아니라 파일이면 모두 FALSE를 반환합니다. 구체적으로 FALSE 값인 이유가 값이 존재하지 않아서인지 파일이어서 그런 것인지 구분하려면 file_exists() 함수를 같이 사용합니다.

is_file

[PHP 3, PHP 4, PHP 5]

bool is_file (string filename)

입력된 인자가 파일인지 확인한다.

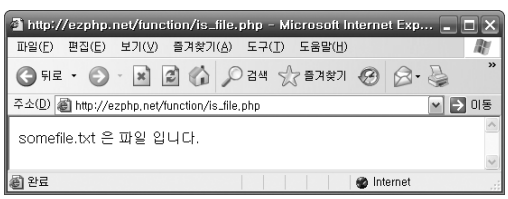
인자	자료형	설명	비고
filename	string	파일 이름	필수

[표 4-28] is_file 함수

is_file() 함수는 filename이 실제로 존재하는 파일인지를 반환하는 함수입니다. filename이 실제로 존재하지 않거나 파일이 아니면 FALSE를 반환합니다.

[예제 4-26] is_file() 함수를 이용하여 파일인지 검사한다.

```
1  <?
2     $filename = "somefile.txt";
3
4     if (is_file($filename)) {
5        echo "$filename 은 파일 입니다.";
6     } else {
7        echo "$filename 은 파일이 아니거나 존재하지 않습니다.";
8     }
9  ?>
```



[그림 4-24] 예제 4-26의 실행결과

is_uploaded_file

[PHP 3 >= 3.0.17, PHP 4 >= 4.0.3, PHP 5]

bool is_uploaded_file (string filename)

해당 파일이 업로드된 파일인지 확인한다.

인자	자료형	설명	비고
filename	string	파일 이름	필수

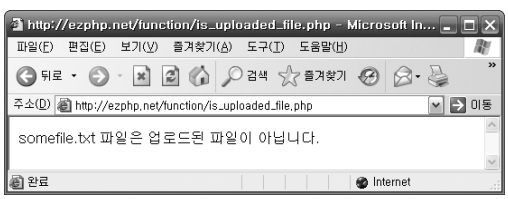
[표 4-29] is_uploaded_file 함수

is_uploaded_file() 함수는 HTTP POST를 통해서 업로드된 파일인지를 검사합니다. HTTP POST를 이용하여 파일을 업로드하는 경우 $_FILES[userfile][tmp_name]과 같이 업로드된 파일명을 얻을 수 있습니다. userfile은 입력 폼의 이름으로 input 태그의 이름입니다.

HTTP POST를 이용하여 업로드하는 경우 업로드된 파일은 임시 디렉토리에 저장됩니다. 임시 저장된 파일의 이름이 tmp_name 배열 원소에 저장됩니다.

[예제 4-27] is_uploaded_file() 함수를 이용하여 업로드된 파일인지 검사한다.

```
1  <?
2    $_FILES['userfile']['tmp_name']= "somefile.txt";
3
4    if (is_uploaded_file($_FILES['userfile']['tmp_name'])) {
5      echo $_FILES['userfile']['tmp_name']." 파일이 업로드 되었습니다.";
6    } else {
7      echo $_FILES['userfile']['tmp_name']." 파일은 업로드된 파일이 아닙니다.";
8    }
9  ?>
```



[그림 4-25] 예제 4-27의 실행결과

mk_dir

[PHP 3, PHP 4, PHP 5]

bool mk_dir (string pathname [, int mode [, bool recursive [, resource context]]])

디렉토리를 생성한다.

인자	자료형	설명	비고
pathname	string	생성할 디렉토리 이름	필수
mode	int	생성할 디렉토리의 사용자 권한	옵션
recursive	bool	재귀적으로 디렉토리를 생성할지 여부	옵션
context	resource	stream context	옵션

[표 4-30] mk_dir 함수

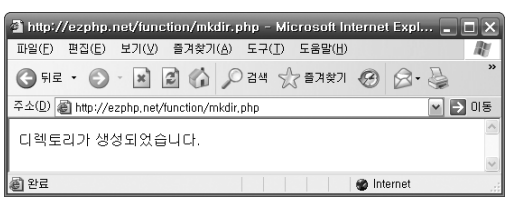
mkdir() 함수는 지정한 디렉토리를 생성하는 함수입니다. 유닉스 시스템은 두 번째 인자를 통해서

사용자 권한을 부여할 수 있습니다. 세 번째 인자는 pathname이 여러 개의 디렉토리를 포함하고 있을 때 상위 디렉토리가 존재하지 않는 경우 상위 디렉토리부터 최하단 디렉토리까지 생성할지에 대한 여부를 결정짓는 인자입니다.

예를 들면 "/home/brown/book/test/"를 생성한다고 했을 때 실제로 /home/brown까지의 디렉토리밖에 존재하지 않는 경우 recursive를 TRUE로 셋팅하면 /home/brown/book 디렉토리를 생성한 후 /home/brown/book/test/ 디렉토리까지 생성할 수 있습니다.

[예제 4-28] mk_dir() 함수를 이용하여 디렉토리를 생성한다.

```
1  <?
2    if (mkdir("/path/to/my/dir", 0700)) {
3      echo "디렉토리가 생성되었습니다.";
4    } else {
5      echo "디렉토리를 만드는데 실패하였습니다.";
6    }
7  ?>
```



[그림 4-26] 예제 4-28의 실행결과

move_uploaded_file

[PHP 3, PHP 4, PHP 5]

bool move_uploaded_file (string filename, string destination)

업로드된 파일을 다른 디렉토리로 이동한다.

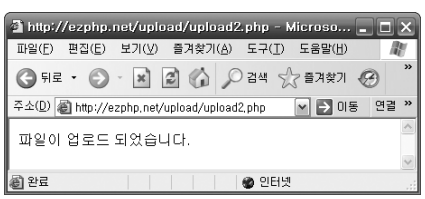
인자	자료형	설명	비고
filename	string	업로드된 파일의 임시 저장 위치	필수
destination	string	옮겨질 위치	필수

[표 4-31] move_uploaded_file 함수

move_uploaded_file() 함수는 HTTP POST를 이용하여 업로드된 파일을 임시 디렉토리에서 다른 디렉토리로 이동시키는 함수입니다. HTTP POST를 이용하여 업로드를 한 경우 임시저장 공간에 일시적으로 저장되는데 스크립트가 종료되면 임시 저장 공간의 임시 파일들은 모두 삭제됩니다. 이에 업로드 후 파일을 다른 특정 위치로 옮겨주어야 할 때가 많습니다. 앞서 배운 is_uploaded_file() 함수와 함께 사용할 수 있으며 is_uploaded_file() 함수로 하여금 업로드된 파일이 맞는지 체크한 후에 move_uploaded_file() 함수를 이용하여 특정 위치로 이동할 수 있습니다.

[예제 4-29] move_uploaded_file() 함수를 이용하여 업로드된 파일을 이동한다.

```
1  <?
2    //폼을 통해서 파일이 전달된 경우
3    if (is_uploaded_file($_FILES['userfile']['tmp_name'],
4    './uploads_  dir')) {
5      echo "파일이 업로드 되었습니다.";
6    } else {
7      echo "임시 저장된 파일을 이동하는데 실패하였습니다.";
8    }
9  ?>
```



[그림 4-27] 예제 4-29의 실행결과

move_uploaded_file() 함수를 실제로 테스트해보기 위해서는 위의 소스만으로는 불가능하고 실제로 파일을 PHP로 전달해주는 폼 페이지가 필요합니다. 이 부분은 별도로 13장 업로드 단원에서 자세히 다룹니다.

readfile

[PHP 4, PHP 5]

int readfile (string filename [, int use_include_path])

파일의 모든 데이터를 읽어서 표준 출력으로 내보낸다.

인자	자료형	설명	비고
filename	string	읽어들일 파일 이름	필수
use_include_path	int	include_path에서 파일을 찾고 싶으면 1로 설정	옵션

[표 4-32] readfile 함수

readfile() 함수는 파일 전체의 내용을 한 번에 읽어들이는 함수입니다. 읽어들인 데이터는 표준 출력장치로 출력되고 그 결과로 데이터의 크기를 반환합니다. 만약 파일을 읽어들이는데 실패한다면 FALSE를 반환합니다.

[예제 4-30] readfile() 함수를 이용하여 파일의 내용을 출력한다.

```
1  <?
2    if (readfile("readfile.txt") == 0)
3      echo "파일을 읽어 들이지 못하였습니다.";
4  ?>
```

이 예제를 실행하면 readfile.txt 파일의 내용이 모두 웹 브라우저에 출력됩니다. 특정 파일의 내용을 출력하고자 하는 용도로 사용할 수도 있으나 실제로는 파일 다운로드를 구현하기 위해서 많이 사용하는 함수입니다. 일반적으로 파일의 링크를 연결해두면 링크를 클릭하는 것만으로도 파일 다운로드 창이 뜨게 됩니다. 그러나 TXT 문서나 이미지와 같이 웹 브라우저에서 지원하는 형식의 파일이면 파일 다운로드 창이 뜨지 않고 바로 웹 브라우저에서 보이는 현상이 발생합니다. 이처럼 웹 브라우저에서 보이는 것을 방지하고 파일 다운로드 창이 뜨도록 하려면 파일의 형식을 속여야 합니다.

원리는 간단합니다. 웹 브라우저가 인식하는 파일 형식을 인식하지 못하는 파일 형식인 것으로 오인하게 만드는 것입니다. 예를 들어 이미지 파일인데 exe 실행 파일인 것처럼 속이는 것입니다. 그러기 위해서 헤더 정보를 수정합니다.

[예제 4-31] readfile() 함수를 이용하여 다운로드 구현

```
1  <?
2    $filename = "/var/www/temp/readfile.txt";
3
4    //헤더 정보를 변경한다.
5    header("Pragma: public");
6    header("Expires: 0");
7    header("Cache-Control: must-revalidate, post-check=0, pre-check=0");
8    header('Content-Description: File Transfer');
9
10   //파일의 형식을 변경
```

```
11    header('Content-Type: application/octet-stream');
12
13    //파일의 크기
14    header('Content-Length: ' . filesize($filename));
15
16    //파일 이름
17    header('Content-Disposition: attachment; filename='
18    . basename($filename));
19    //파일을 출력
20    @readfile($filename);
21  ?>
```

rename

[PHP 4, PHP 5]

bool rename (string oldname , string newname)

파일의 이름을 변경한다.

인자	자료형	설명	비고
oldname	string	기존 파일 이름	필수
newname	string	변경할 새 파일 이름	필수

[표 4-33] rename 함수

rename() 함수는 파일의 이름을 변경하는 함수입니다. 파일 이름을 변경하는 데 성공하면 TRUE를, 실패하면 FALSE를 반환합니다.

[예제 4-32] rename() 함수를 이용하여 파일의 이름을 변경한다.

```
1  <?
2     rename("/tmp/tmp_file.txt", "/var/www/file.txt");
3  ?>
```

rmdir

[PHP 4, PHP 5]
bool rmdir (string dirname)
디렉토리를 제거한다.

인자	자료형	설명	비고
dirname	string	디렉토리 이름	필수

[표 4-34] rmdir 함수

rmdir() 함수는 디렉토리를 제거합니다. 그러나 디렉토리를 제거하려면 반드시 디렉토리 안이 비어 있어야 하고 디렉토리를 지우기 위한 권한을 갖고 있어야 합니다. 디렉토리가 지워지면 TRUE를, 지워지지 않으면 FALSE를 반환합니다.

unlink

[PHP 4, PHP 5]
bool unlink (string filename)
파일을 삭제한다.

인자	자료형	설명	비고
filename	string	삭제할 파일 이름	필수

[표 4-35] unlink 함수

unlink() 함수는 지정된 파일을 삭제합니다. 유닉스용 C의 unlink() 함수를 본떠서 만들었으나 윈도우와 유닉스 시스템 모두에서 동작합니다. 파일을 삭제하려면 파일을 삭제할 수 있는 권한을 가져야만 하며 삭제되면 TRUE를, 삭제되지 않으면 FALSE를 반환합니다.

Section 03

문자열 처리 함수

문자열 처리는 대부분의 프로그래밍에서 가장 중요한 부분 중 하나입니다. 다른 프로그래밍 언어에서는 문자열 처리를 하려고 자신이 직접 구현한 함수를 사용하는 경우가 많은데 그것은 프로그래밍 언어 차원에서 지원하는 함수가 너무 부족하기 때문입니다. 그래서 C++의 STL(Standard Template Library)이나 MFC(Microsoft Foundation Class)와 같은 라이브러리를 보면 표준 라이브러리에서 지원하지 않는 편리한 문자열 클래스와 함수를 지원하고 있습니다.

다른 프로그래밍 언어보다 웹 프로그래밍 언어는 문자열 처리를 해야 하는 경우가 많습니다. 다행히도 PHP에서는 다양하면서 유용한 문자열 처리 함수를 지원하기 때문에 직접 문자열을 처리하는 불편함을 겪는 경우가 적습니다. 많은 PHP 개발진들의 노력으로 우리는 보다 쉽게 웹 프로그래밍을 할 수 있게 된 것입니다. 문자열 처리는 매우 중요한 부분인 만큼 전체적으로 어떤 함수들이 있는지 정도는 반드시 알고 넘어가야 합니다. 이 책에 언급하지 못한 많은 함수는 PHP 홈페이지에서 찾아서 기억하기 바랍니다.

함수 이름	기능
crypt	단방향으로 문자열을 암호화
echo	문자열을 출력
explode	문자열을 특정 문자열을 기준으로 분리
htmlentities	해당하는 모든 문자를 HTML 엔티티로 변환
htmlspecialchars	특수문자를 HTML 엔티티로 변환
implode / join	배열의 원소를 문자열로 연결
ltrim	문자열 왼쪽의 공백을 제거
md5	문자열의 MD5 해시값을 반환
nl2br	문자열의 모든 줄 바꿈 앞에 〈br /〉 태그를 삽입
print	문자열을 출력
printf	형식화된 문자열을 출력
rtrim / chop	문자열 오른쪽의 공백을 제거
sprintf	형식화된 문자열로 반환
sscanf	문자열을 형식에 따라 처리
str_replace	문자열을 찾아서 치환
strip_tags	문자열에서 HTML과 PHP 태그를 제거
strlen	문자열의 길이를 반환
strpos	문자열이 처음 나타나는 위치를 반환
strstr / strchr	문자열이 처음으로 나타나는 위치를 반환

substr	문자열의 일부를 반환
trim	문자열의 처음과 끝에 있는 공백을 제거

[표 4-36] 문자열 처리 함수

문자열 함수에 대한 자세한 내용을 알고 싶다면 PHP 매뉴얼을 이용해 보십시오.

http://www.php.net/manual/kr/ref.strings.php

crypt

[PHP 4, PHP 5]

string crypt (string str [, string salt])

단방향 문자열 암호화를 한다.

인자	자료형	설명	비고
str	string	암호화하고자 하는 문자열	필수
salt	string	암호화 방식을 결정하는 문자열	옵션

[표 4-37] crypt 함수

crypt() 함수는 단방향 알고리즘을 사용하여 문자열을 암호화합니다. 단방향 알고리즘이란 암호화된 문자열을 다시 원문으로 되돌리는 복호화 알고리즘이 존재하지 않는 알고리즘을 말합니다. 즉, 이 함수를 통해 얻어진 문자열을 통해서 원래의 문자열을 얻을 수 있는 방법이 없습니다. crypt() 함수는 다음과 같은 4가지의 암호화 방식을 지원합니다.

① CRYPT_STD_DES: 2문자 salt를 가지는 표준 DES 기반 암호화
② CRYPT_EXT_DES: 9문자 salt를 가지는 확장 DES 기반 암호화
③ CRYPT_MD5: 1로 시작하는 12문자 salt를 가지는 MD5 암호화
④ CRYPT_BLOWFISH: 2로 시작하는 16문자 salt를 가지는 Blowfish 암호화

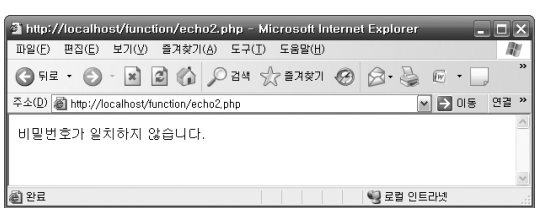
두 번째 인자인 salt로 암호화 방식을 선택할 수 있으며 salt를 지정하지 않는 경우 자동으로 선택됩니다. 그러나 각 암호화 알고리즘은 시스템이 지원하지 않으면 사용할 수 없습니다. 확장 DES 알고리즘을 지원하는지 알고 싶다면 CRYPT_EXT_DES 값이 1인지 확인해보면 됩니다.

여기서 주의할 점은 표준 DES 기반 암호화를 하는 경우 문자열의 앞에서 8글자만을 사용하여 암호화합니다. 예를 들어 happybrown이라는 문자열을 암호화한 것과 happybroxx이라는 문자열을 암

호화한 것이 동일하게 된다는 의미입니다. 따라서 표준 DES 암호화 방식은 되도록 사용하지 않는 것이 좋습니다.

[예제 4-33] crypt() 함수를 이용하여 비밀번호를 검증한다.

```
<?
  $original_string = "happybrown";
  $salt = "$1$brownsalt";
  $user_input = "happyblack";

  if (CRYPT_MD5 == 1)
  {
    $crypt_string = crypt($original_string, $salt);

    if ($crypt_string == crypt($user_input, $salt))
    {
      echo "비밀번호가 일치합니다.";
    }
    else
    {
      echo "비밀번호가 일치하지 않습니다.";
    }
  }
  else
  {
    echo "MD5 암호화 알고리즘을 지원하지 않습니다.";
  }
?>

```



[그림 4-28] 예제 4-33의 실행결과

[예제 4-33]과 같이 crypt() 함수는 비밀번호 검증과 같은 부분에 사용할 수 있습니다. 회원가입을 통해서 입력받은 비밀번호를 순수한 문자열로 저장하면 유출 시 큰 문제가 될 수 있습니다. 만약 비밀번호를 암호화하여 저장하면 설사 비밀번호가 유출되었다 하더라도 암호화되기 전의 원문을 알 수 없어서 악용할 수 없습니다. 왜냐하면 로그인을 위해서 비밀번호에 암호문을 입력하게 된다면 입력된 암호문이 다시 한번 암호화되어 비교되기 때문입니다.

echo

[PHP 4, PHP 5]

void echo (string str1 [, string str2 …])

문자열을 출력한다.

인자	자료형	설명	비고
str1	string	출력하고자 하는 문자열	필수
str2 …	string	출력하고자 하는 문자열	옵션

[표 4-38] echo 구조

echo는 실제로 함수가 아니라 제어 구조입니다. 따라서 함수와 같이 괄호를 사용해도 되고 괄호를 사용하지 않아도 무관합니다. 그러나 둘 이상의 인자를 사용하는 경우 괄호를 사용하면 안 됩니다.

[예제 4-34] echo 구조의 사용법

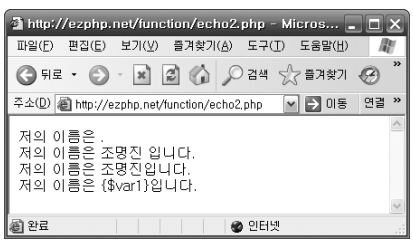
```
1  <?
2     echo "안녕하세요. 조명진입니다.";
3
4     echo "여러 줄로 나누어 사용해도
5     사용이 가능합니다.";
6
7     echo "여러 줄로 나누어 사용해도\n사용이 가능합니다.";
8
9     echo "따옴표를 출력하고 싶다면 \"이렇게\" 합니다.";
10 ?>
```

[예제 4-34]는 가장 일반적인 echo 구조의 사용 방법입니다. 큰따옴표를 이용하여 출력하며 4-5행처럼 여러 줄로 나누어서 문자열을 출력해도 문법상 오류가 나지 않습니다. 왜냐하면 줄 바꿈이 된 부분에 개행문자(\n)가 있다고 생각하기 때문입니다. 그래서 7행과 실질적으로 같은 결과를 얻게 됩니다. 마지막으로 따옴표나 변수 기호와 같이 출력 구문에서 의미가 있는 기호를 모양 그대로 출력하고자 할 때는 9행처럼 역슬래시(\)를 앞에 붙여주어야 합니다.

[예제 4-35] echo 구조에서 변수 사용법

```
1  <?
2     $var1 = "조명진";
```

```
3
4     echo "저의 이름은 $var1입니다.<BR>";
5     echo "저의 이름은 $var1 입니다.<BR>";
6     echo "저의 이름은 {$var1}입니다.<BR>";
7
8     echo '저의 이름은 {$var1}입니다.";
9  ?>
```



[그림 4-29] 예제 4-35의 실행결과

[예제 4-35]는 echo 구문 내에서 변수를 다루는 방법을 알려주고 있습니다. 그런데 4행은 우리가 원하는 대로 출력되지 않고 **저의 이름은 .**으로 끝나고 있음을 알 수 있습니다. 그 이유는 '$var1입니다' 전체가 변수 이름으로 인식되었기 때문입니다. 즉, 변수와 뒤의 문자열이 붙어서 어디까지가 변수 이름인지 알 수 없게 되어 버린 것입니다.

그래서 5행처럼 변수 뒤에 공백을 두는 방법을 많이 사용합니다. 변수 뒤에 공백을 두면 첫 번째 구문과 같이 변수의 이름을 오해하는 경우가 생기지 않을 것이기 때문입니다. 그러나 이름 뒤에 공백이 붙게 되므로 임시적인 방편일 뿐입니다.

6행이 바로 변수 문제를 해결하는 방법입니다. 변수 앞뒤에 중괄호를 추가해서 해당 부분이 변수임을 확실히 나타내는 방법입니다. 조금 번거롭기는 하지만 확실한 방법이므로 이 방법을 사용하는 것이 좋습니다.

마지막으로 8행에서 위의 세 가지 방법과 달리 작은따옴표를 사용하여 변수를 출력해 보았습니다. 그러나 변수가 출력되지 않고 변수 이름이 그대로 출력된 것을 확인할 수 있습니다. 왜냐하면 작은따옴표에서는 파싱을 하지 않고 작은따옴표 안의 문자열을 그대로 출력하기 때문입니다. 그래서 echo 구문 속에 변수를 사용하고 싶다면 큰따옴표를 사용해야 합니다. 하지만 문자열 파싱이 필요 없는 문자열이라면 큰따옴표를 사용해서 굳이 문자열 파싱을 할 이유가 없습니다. 즉, 단순 문자열 출력은 불필요한 파싱 절차를 없애는 작은따옴표를 이용하는 것이 성능 측면에서 좋습니다.

explode

[PHP 4, PHP 5]

array explode (string separator, string str [, int limit])

문자열을 주어진 문자열을 기준으로 분리한다.

인자	자료형	설명	비고
separator	string	분리 기준이 되는 문자열	필수
str	string	분리하고자 하는 문자열	필수
limit	int	최대 배열 원소의 수	옵션

[표 4-39] explode 함수

explode() 함수는 문자열을 지정한 문자열을 기준으로 분리하여 배열의 원소로 저장합니다. 첫 번째 인자가 분리의 기준이 되는 문자열이고 두 번째 인자가 분리하고자 하는 원문입니다. 마지막으로 limit 값을 지정하면 최대 원소의 수가 정해지고 최대 원소의 수보다 더 많이 분리되면 마지막 원소에는 나머지 문자열이 모두 들어가게 됩니다.

어떤 문자열을 특정 기호로 나누어 여러 가지 값을 저장했을 때 이 함수를 이용하여 쉽게 배열로 값을 분리할 수 있습니다. explode() 함수와 함께 list() 함수를 같이 사용하는 경우가 많습니다.

[예제 4-36] explode() 함수를 이용하여 문자열을 분리한다.

```
1  <?
2    $nickname = "조명진|브라운|다우니|brown|행복한브라운|happybrown";
3    $nickname_array = explode("|", $nickname);
4
5    echo "<PRE>";
6    print_r($nickname_array);
7    echo "</PRE>";
8  ?>
```

[그림 4-30] 예제 4-36의 실행결과

htmlentities

[PHP 4, PHP 5]

string htmlentities (string str [, int quote_style [, string charset]])

해당하는 모든 문자를 HTML 엔티티로 변환한다.

인자	자료형	설명	비고
str	string	변환하고자 하는 문자열	필수
quote_style	int	따옴표 처리 방법	옵션
charset	string	변환에 사용할 문자셋	옵션

[표 4-40] htmlentities 함수

htmlentities() 함수는 알파벳과 같은 단순 문자를 제외한 모든 기호를 HTML 엔티티로 변환합니다. HTML 태그는 '<' 문자로 시작하여 '>' 문자로 끝이 납니다. 그런데 만약 '<' 문자를 그대로 출력하고자 한다면 어떻게 표현해야 할까요? 이런 문제를 해결하기 위해서 정의된 것이 HTML 엔티티입니다. HTML에서 의미가 있는 모든 특수기호들을 웹 페이지에 표시하기 위해서 정의된 문자열입니다. 대표적인 HTML 엔티티들은 다음과 같습니다.

특수문자	HTML 엔티티 문자열
"	&quot;
'	&#039;
&	&amp;
<	&lt;
>	&gt;
스페이스	

[표 4-41] HTML 엔티티

이외에도 수많은 기호와 심지어 한글까지 HTML 엔티티로 정의되어 있습니다. 그러나 실제로 유념해 두어야 할 것들은 위의 6가지 정도밖에 되지 않습니다. 두 번째 인자인 따옴표 처리 방식은 세 가지가 있습니다.

① ENT_COMPAT: 큰따옴표만 변환하고 작은따옴표는 변환하지 않는다.

② ENT_QUOTES: 큰따옴표와 작은따옴표를 모두 변환한다.

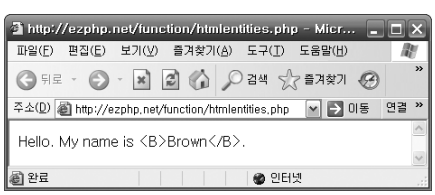
③ ENT_NOQUOTES: 큰따옴표와 작은따옴표를 모두 변환하지 않는다.

위와 같은 인자를 주어 따옴표를 어떻게 처리할 것인지를 지정할 수 있습니다. 만약 따옴표 처리 방식을 지정하지 않으면 ENT_COMPAT이 기본값으로 설정됩니다.

마지막으로 charset은 변환에 사용할 문자셋을 지정하는 것입니다. 그러나 한글은 지원되지 않아서 문자열에 한글이 있으면 한글을 제대로 볼 수 없습니다.

[예제 4-37] htmlentities() 함수를 이용하여 특수문자를 모두 HTML 엔티티로 변환한다.

```
1  <?
2    $string = "Hello. My name is <B>Brown</B>.";
3
4    echo htmlentities($string);
5  ?>
```



[그림 4-31] 예제 4-37의 실행결과

아쉽지만 한글이 지원되지 않기 때문에 안 되는 영어를 써보았습니다. 출력된 결과를 보면 HTML로 작성된 문장이 HTML 태그가 반영되지 않고 그대로 출력되는 것을 알 수 있습니다. 모든 특수문자가 HTML 엔티티로 변환되었기 때문입니다. 위의 실제 변환 결과는 다음과 같습니다.

```
Hello. My name is &lt;B&gt;Brown&lt;/B&gt;.
```

이처럼 '<', '>' 기호들이 각각 '<', '>'로 변환된 것을 알 수 있습니다.

htmlspecialchars

[PHP 4, PHP 5]

string htmlspecialchars (string str [, int quote_style [, string charset]])

특수문자를 HTML 엔티티로 변환한다.

인자	자료형	설명	비고
str	string	변환하고자 하는 문자열	필수
quote_stylei	nt	따옴표 처리 방법	옵션
charset	string	변환에 사용할 문자셋	옵션

[표 4-42] htmlspecialchars 함수

htmlspecialchars() 함수는 htmlentities() 함수와 기능이 동일하나 모든 문자에 대해서 변경하지 않고 5가지의 문자에 대해서만 HTML 엔티티로 변경합니다.

특수문자	HTML 엔티티 문자열
"	&quot;
'	&#039;
&	&amp;
〈	&lt;
〉	&gt;

[표 4-43] HTML 엔티티

단, 두 번째 인자인 따옴표 처리 방식에 따라 큰따옴표와 작은따옴표의 변환 여부가 결정됩니다.

[예제 4-38] htmlspecialchars() 함수를 이용하여 특수문자를 HTML 엔티티로 변환한다.

```
1  <?
2    $string = "안녕하세요. <B>조명진</B>입니다.";
3
4    echo htmlspecialchars($string);
5  ?>
```

[그림 4-32] 예제 4-38의 실행결과

htmlentities() 함수가 한글까지 모두 HTML 엔티티로 변경하는 데 비해서 htmlspecialchars() 함수는 특수문자 5개에 한해서 HTML 엔티티로 변경합니다. 이 함수는 주로 게시판이나 방명록과 같이 사용자가 HTML 태그를 입력했을 때 그것을 HTML로 보여주지 않고 입력한 값을 그대로 보여주기 위한 용도로 많이 사용됩니다.

implode, join

[PHP 4, PHP 5]

string implode (string glue, array pieces)

배열의 원소를 하나의 문자열로 반환한다.

인자	자료형	설명	비고
glue	string	문자열 구분자	필수
pieces	array	변환하고자 하는 배열	필수

[표 4-44] implode, join 함수

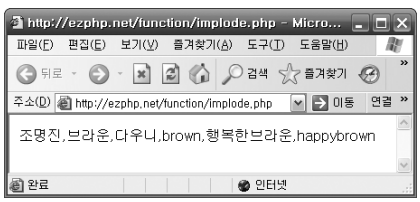
implode() 함수는 explode() 함수와 정반대의 기능을 하는 함수로 배열을 하나의 문자열로 반환합니다. 그리고 join() 함수는 implode() 함수의 별칭입니다.

첫 번째 인자는 배열의 원소를 문자열로 합칠 때 원소의 구분자로 쓸 문자열을 의미합니다. 두 번째 인자는 변환하고자 하는 배열을 나타냅니다. 그런데 관습적으로 두 인자의 순서를 바꾸는 것을 허용합니다. 또한 glue 인자를 입력하지 않는 것을 허용합니다. 그러나 될 수 있으면 explode() 함수와 같은 순서인 위와 같은 인자의 순서를 지키고 glue 인자를 사용하기를 권장합니다.

[예제 4-39] implode() 함수를 이용하여 배열을 문자열로 변환한다.

```
1  <?
2     $nickname_array = array('조명진','브라운','다우니','brown',
```

```
3     '행복한브라운','happybrown');
4
5     $nickname = implode(",", $nickname_array);
6     echo $nickname;
7   ?>
```



[그림 4-33] 예제 4-39의 실행결과

ltrim

[PHP 4, PHP 5]

string ltrim (string str [, string charlist])

문자열 시작 부분의 공백을 제거한다.

인자	자료형	설명	비고
str	string	공백을 없애고자 하는 문자열	필수
charset	string	제거하고 싶은 문자 목록	옵션

[표 4-45] ltrim 함수

ltrim() 함수는 문자열의 시작 부분에 존재하는 공백을 제거합니다. 제거하는 공백은 일반적인 스페이스 공백( )과 탭(\t), 줄 바꿈(\n), 캐리지 리턴(\r), 널 문자(\0) 그리고 수직 탭(\x0B)입니다. 두 번째 인자에 제거하고자 하는 문자를 나열하면 해당 문자가 제거됩니다.

[예제 4-40] ltrim() 함수를 이용하여 왼쪽의 공백을 제거한다.

```
1   <?
2     $string = " \t안녕하세요.\t조명진입니다.";
3     echo ltrim($string, " \t조명진");
4   ?>
```

ltrim() 함수를 이용하여 공백과 탭 문자 그리고 조명진이라는 문자열을 지우려고 시도해 보았습니다. 그런데 다음과 같이 예상외의 결과가 발생합니다.

```
안녕하세요.  조명진입니다.
```

소스 보기를 통해서 그 결과를 확인해보면 위와 같은 결과가 나옵니다. 앞에 공백과 탭 문자는 잘 제거되었지만 중간의 탭 공간과 조명진이라는 이름은 제거되지 않은 것을 알 수 있습니다. 그 이유는 ltrim() 함수가 문자열의 왼쪽 공간만 고려하기 때문입니다. 문자열의 처음부터 스페이스와 탭 문자 그리고 '조명진'이라는 문자열을 찾아서 없애기 시작하다가 이에 해당하지 않는 '안녕하세요'라는 문자열을 보고 문자를 제거하는 것을 그만두었기 때문입니다.

그리고 여기서 주의해야 할 점은 두 번째 인자를 이용하여 문자를 제거하고자 할 때 공백을 지정해 주지 않으면 공백은 지워지지 않는다는 것입니다. 즉, 두 번째 인자가 지정되면 지정된 문자만을 제거합니다.

md5

[PHP 4, PHP 5]

string md5 (string str [, bool raw_output])

문자열의 해시값을 반환한다.

인자	자료형	설명	비고
str	string	해시값을 구하고자 하는 문자열	필수
raw_output	bool	16자 순수 바이너리 형식 여부 (PHP 5.0.0 이상)	옵션

[표 4-46] md5 함수

md5() 함수는 문자열을 통해서 16진수의 32글자 문자 형태인 MD5 해시를 계산하여 반환합니다. 또한 해시는 일방향 함수이므로 해시값을 통해서 원문을 구할 수 없습니다.

두 번째 인자인 raw_output을 TRUE로 지정하면 32자가 아닌 16자의 바이너리 형식 해시값을 반환합니다.

[예제 4-41] md5() 함수를 이용하여 해시값을 반환한다.

```
1  <?
2    $string = "brown";
3    echo md5($string);
4  ?>
```

결과값은 다음과 같습니다.

```
6ff47afa5dc7daa42cc705a03fca8a9b
```

md5() 함수는 crypt() 함수와 함께 비밀번호를 암호화하여 검증하는 데 자주 사용합니다. 개인적으로 salt 값을 지정해줄 필요가 없어서 crypt() 함수보다 md5를 더 선호합니다.

nl2br

[PHP 4, PHP 5]

string nl2br (string str)

문자열의 모든 개행문자 앞에 〈br /〉 태그를 추가한다.

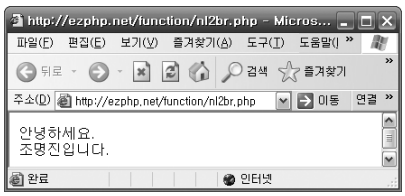
인자	자료형	설명	비고
str	string	처리하고자 하는 문자열	필수

[표 4-47] nl2br 함수

nl2br() 함수는 문자열의 줄 바꿈 기호 앞에 HTML 줄 바꿈 태그를 삽입합니다. 단순한 줄 바꿈이 웹 페이지에서는 의미가 없기 때문에 HTML 줄 바꿈 태그를 삽입하여 웹 페이지에서도 줄 바꿈이 되게 만들어주는 함수입니다.

[예제 4-42] nl2br() 함수를 이용하여 HTML 줄 바꿈 태그를 삽입한다.

```
1  <?
2    $string = "안녕하세요.\n조명진입니다.";
3    echo nl2br($string);
4  ?>
```



[그림 4-34] 예제 4-42의 실행결과

결과에서 알 수 있듯이 웹 페이지에서도 줄 바꿈이 이루어진 것을 확인할 수 있습니다. 그래서 게시판이나 방명록에서 사용자가 입력한 글의 줄 바꿈을 웹 페이지에서도 적용하기 위해 nl2br() 함수를 많이 사용합니다.

print

[PHP 4, PHP 5]

int print (string str)

문자열을 출력한다.

인자	자료형	설명	비고
str	string	출력하고자 하는 문자열	필수

[표 4-48] print 구조

print는 echo 구문과 마찬가지로 함수가 아니라 언어 구조입니다. 둘은 거의 동일하지만 echo가 여러 개의 문자열을 콤마를 통해서 인자로 받아들일 수 있는데 반해 print는 하나의 문자열만 출력할 수 있습니다. 또한 echo가 리턴값이 없는 데 반해, print는 리턴값을 되돌려 줍니다. print는 언어 구조이긴 하나 함수와 같이 동작하기도 합니다.

[예제 4-43] print 언어 구조를 이용하여 문자열을 출력한다.

```
1  <?
2    $var = TRUE;
3    $var ? print('TRUE'): print('FALSE');
4  ?>
```

결과는 다음과 같습니다.

```
TRUE
```

[예제 4-43]은 삼항 연산자를 통해서 $var 변수의 참 거짓 여부를 판단하고 참이면 TRUE를, 거짓이면 FALSE를 출력하도록 했습니다. echo의 경우에는 언어 구조로만 동작하기 때문에 print 대신에 echo로 변경하면 에러가 발생합니다. 그러나 print의 경우에는 비록 echo와 같은 언어 구조이기는 하나 함수처럼 동작할 수 있기 때문에 위와 같이 함수처럼 사용하여도 에러가 발생하지 않습니다. 이러한 차이를 제외하고는 echo와 print의 차이는 없습니다.

printf

[PHP 4, PHP 5]

void printf (string format [, mixed args])

문자열을 형식화하여 출력한다.

인자	자료형	설명	비고
Format	string	출력하고자 하는 형식	필수
Args	mixed	출력하고자 하는 변수	옵션

[표 4-49] printf 함수

printf() 함수는 문자열을 지정된 형식으로 출력합니다. 두 번째 인자를 통해서 전달된 변수 값을 첫 번째 인자에 지정된 형식을 통해서 출력합니다.

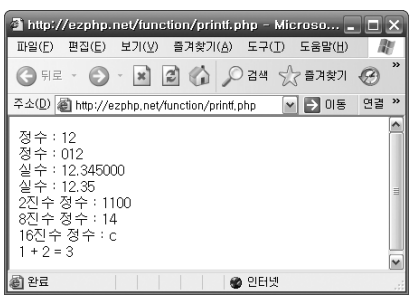
형식 지정어	설명
%d	10진수인 정수
%f	실수
%b	2진수인 정수
%o	8진수인 정수
%x	16진수인 정수
%s	문자열

[표 4-50] 형식 지정어

이외에도 몇 가지 형식 지정어가 있지만 대부분의 경우 위의 표에 언급된 지정어를 통해 원하는 형식으로 출력할 수 있습니다. % 기호와 알파벳 사이에는 숫자를 지정할 수 있어서 출력 형식의 자릿수를 맞출 수도 있습니다.

[예제 4-44] printf() 함수를 이용하여 문자열을 형식화하여 출력한다.

```
1  <?
2     $float = 12.345;
3
4     printf("정수 : %d <BR>", $float);
5     printf("정수 : %03d <BR>", $float);
6
7     printf("실수 : %f <BR>", $float);
8     printf("실수 : %.2f <BR>", $float);
9
10    printf("2진수 정수 : %b <BR>", $float);
11    printf("8진수 정수 : %o <BR>", $float);
12    printf("16진수 정수 : %x <BR>", $float);
13
14    printf("%d + %d = %d", 1, 2, 1+2);
15 ?>
```



[그림 4-35] 예제 4-44의 실행결과

실수인 12.345를 각 숫자의 형식에 맞춰서 출력했습니다. 4행은 $float 변수가 실수지만 10진수인 정수로 출력했기 때문에 소수점 이하의 숫자는 출력되지 않았습니다. 5행은 %와 d 사이에 03을 추가하여 3자리의 숫자로 출력하되 자리수가 모자라면 0을 채워넣도록 했습니다. 7행은 실수 형식으로 출력했는데 기본적으로 소수점 6자리까지 출력되므로 뒤에 3자리가 0으로 채워져 있음을 알 수 있습니다. 그래서 소수점 자릿수를 지정하여 2자리만 출력하게 한 것이 8행입니다. 10행에서부터 12행까지는 $float 변수값을 각각 2진수, 8진수, 16진수로 출력한 것입니다. 마지막으로 14행은 하나의 변수가 아닌 여러 개의 변수를 사용하는 방법을 나타낸 것으로 형식 지정어의 수와 변수의 수만 같다면 계속해서 값을 추가할 수 있습니다.

rtrim, chop

[PHP 4, PHP 5]
string rtrim (string str [, string charlist])
문자열 끝 부분의 공백을 제거한다.

인자	자료형	설명	비고
str	string	공백을 제거하고자 하는 변수	필수
charlist	string	제거하고자 하는 문자 목록	옵션

[표 4-51] rtrim, chop 함수

rtrim() 함수는 ltrim() 함수와 유사하며 문자열의 끝 부분에 존재하는 공백을 제거합니다. chop() 함수는 rtrim() 함수의 별칭입니다. rtrim() 함수는 ltrim() 함수의 사용법과 동일하게 사용할 수 있습니다.

sprint

[PHP 4, PHP 5]
string sprintf (string format [, mixed args])
문자열을 형식화하여 문자열로 반환한다.

인자	자료형	설명	비고
format	string	출력하고자 하는 형식	필수
args	mixed	출력하고자 하는 변수	옵션

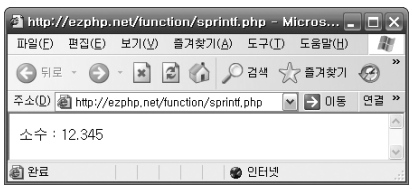
[표 4-52] sprint 함수

sprint() 함수는 printf() 함수와 동일하나 표준 출력을 하지 않고 문자열로 형식화된 문자열을 반환하여 줍니다.

[예제 4-45] sprintf() 함수를 이용하여 형식화한 문자열을 반환한다.

```
1  <?
2     $float = 12.345;
3
```

```
4     $result = sprintf("소수 : %.3f", $float);
5     echo $result;
6   ?>
```



[그림 4-36] 예제 4-45의 실행결과

sscanf

[PHP 4 >= 4.0.1, PHP 5]

mixed sscanf (string str, string format [, string var1])

문자열을 형식에 따라 처리한다.

인자	자료형	설명	비고
str	string	처리하고자 하는 문자열	필수
format	string	해석하고자 하는 형식	필수
var1	string	처리된 값을 해당 변수에 저장	옵션

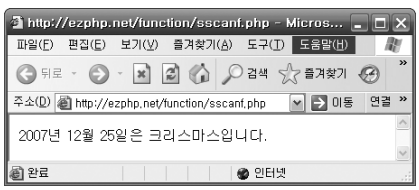
[표 4-53] sscanf 함수

sscanf() 함수는 특정 형식으로 작성된 문자열에서 지정된 형식에 따라 값을 해석해서 가져오려고 할 때 사용하는 함수입니다. sprint() 함수의 반대 성격을 가진 함수라고 할 수 있습니다. 옵션 인자를 사용하지 않으면 형식에 따라 해석된 값은 배열에 저장되고, 옵션 인자를 사용하면 지정한 변수에 값이 하나씩 할당됩니다. 그런데 두 번째 인자를 사용할 때는 반드시 참조 형식으로 전달해야 합니다.

[예제 4-46] sscanf() 함수를 이용하여 형식화된 값을 배열로 저장한다.

```
1   <?
2   $christmas = "2007-12-25";
3
4   $date = sscanf($christmas, "%d-%d-%d");
```

```
5  echo "{$date[0]}년 {$date[1]}월 {$date[2]}일은 크리스마스입니다.";
6  ?>
```



[그림 4-37] 예제 4-46의 실행결과

문자열 '2007-12-25'는 '년-월-일'과 같은 형식으로 이루어져 있습니다. 년월일은 각각 10진수 정수이므로 printf() 함수에서 사용한 형식 지정어를 사용하여 "%d-%d-%d" 형식 문자열을 만들 수 있습니다. 이를 통해서 년월일 값은 각각 $date 배열에 원소로 저장됩니다.

[예제 4-47] sscanf() 함수를 이용하여 형식화된 값을 해당 변수에 저장한다.

```
1  <?
2    $christmas = "2007-12-25";
3
4    $date = sscanf($christmas, "%d-%d-%d", &$year, &$month, &$day);
5    echo "{$year}년 {$month}월 {$day}일은 크리스마스입니다.";
6  ?>
```

[예제 4-47]과 같이 옵션 인자를 참조 형식으로 전달하면 해석된 형식에 따라 순서대로 값이 저장됩니다. 그래서 [예제 4-46]과 같은 결과를 얻을 수 있습니다.

str_replace

[PHP 4, PHP 5]

mixed str_replace (mixed search, mixed replace, mixed subject [, int &count])

문자열을 형식에 따라 처리한다.

인자	자료형	설명	비고
search	mixed	찾고자 하는 문자열 또는 배열	필수

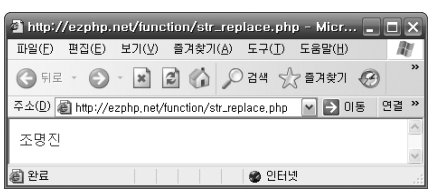
replace	mixed	찾은 문자열을 치환하는 문자열 또는 배열	필수
subject	mixed	치환하고자 하는 문자열 또는 배열	필수
count	int	치환된 횟수 (PHP 5.0.0 이상)	

[표 4-54] str_replace 함수

str_replace() 함수는 어떤 문자열(subject)에서 특정 문자열(search)을 찾아서 다른 문자열(replace)로 치환하는 함수입니다. 문자열뿐만이 아니라 배열도 치환할 수 있습니다.

[예제 4-48] str_replace() 함수를 이용하여 문자열을 치환한다.

```
1  <?
2    $html = "<font color='%font_color%'>조명진</font>";
3    $html = str_replace("%font_color%", "blue", $html);
4    echo $html;
5  ?>
```



[그림 4-38] 예제 4-48의 실행결과

str_replace() 함수는 스킨을 만들고자 자주 사용하는 함수입니다. 위의 예제에서도 알 수 있듯이 글꼴의 색상이나 배경색 그리고 이미지 파일의 이름 등을 '%font_color%', '%bgcolor%', '%img1%'과 같이 쉽게 사용하지 않는 특별한 문자열로 나타낸 후 해당 값을 실제 값으로 치환하는 방법을 이용하여 스킨을 제작할 수 있습니다.

str_replace() 함수는 문자열뿐만이 아니라 배열까지 치환이 가능하므로 게시판이나 방명록에서 사용 금지되는 단어를 배열에다 등록해서 치환하는 기능을 구현하는 데도 사용할 수 있습니다.

[예제 4-49] str_replace() 함수를 이용하여 배열을 치환한다.

```
1  <?
2    $text = "브라운님아. 사시미 조낸 많이 먹어서 배가 만땅이예요.";
3
4    $bad_words = array("님아", "만땅", "조낸", "사시미");
5    $good_words = array("님", "가득", "매우", "회");
6
7    $text = str_replace($bad_words, $good_words, $text);
```

```
8     echo $text;
9  ?>
```

결과는 다음과 같습니다.

```
브라운님. 회 매우 많이 먹어서 배가 가득이예요.
```

strip_tags

[PHP 4, PHP 5]

string strip_tags (string str [, string allowable_tags])

문자열에서 HTML과 PHP 태그를 제거한다.

인자	자료형	설명	비고
str	string	태그를 제거하고자 하는 문자열	필수
allowable_tags	string	허용할 태그	옵션

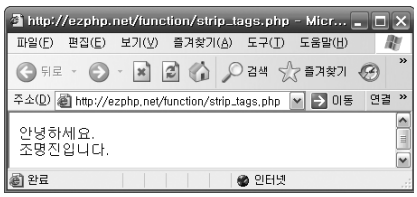
[표 4-55] strip_tags 함수

strip_tags() 함수는 문자열에서 모든 HTML과 PHP 태그를 제거합니다. 심지어는 HTML 주석까지 제거할 수 있습니다. 그러나 HTML의 유효성 검사를 하지 않기 때문에 부분적이거나 깨진 HTML 태그로 인해서 원하지 않는 결과를 얻을 수 있습니다. 또한 자바스크립트는 시작과 끝 태그를 제외한 코드 부분은 지워지지 않습니다.

두 번째 인자에서 허용할 태그를 지정해주면 그 태그를 제외한 나머지 모든 태그를 제거합니다.

[예제 4-50] strip_tags() 함수를 이용하여 HTML 및 PHP 태그를 제거한다.

```
1  <?
2     $text = '안녕하세요.<BR><B>조명진</B>입니다.';
3     echo strip_tags($text, "<br>");
4  ?>
```



[그림 4-39]예제 4-50의 실행결과

strlen

[PHP 4, PHP 5]

int strlen (string str)

문자열의 길이를 반환한다.

인자	자료형	설명	비고
str	string	길이를 구하고자 하는 문자열	필수

[표 4-56] strlen 함수

strlen() 함수는 문자열의 길이를 반환합니다. 그러나 1바이트를 하나의 문자로 계산하기 때문에 한글은 정확한 문자열의 길이를 측정할 수가 없습니다.

[예제 4-51] strlen() 함수를 이용하여 문자열의 길이를 측정한다.

```
1  <?
2    $str = 'abcdef';
3    echo "'{$str}'의 길이 : " . strlen($str) . "<BR>";
4
5    $str = '가나다 라마';
6    echo "'{$str}'의 길이 : " . strlen($str) . "<BR>";
7  ?>
```

[그림 4-40] 예제 4-51의 실행결과

위의 결과에서 알 수 있듯이 알파벳은 6글자로 정확하게 맞춘 것에 비해 한글은 한 글자가 2바이트이기 때문에 총 5글자, 10바이트와 스페이스 1바이트를 합하여 11이라는 길이가 나옴을 알 수 있습니다. 게시판에서 목록을 보여줄 때 제목의 길이 제한을 하는 경우가 있는데 단순하게 strlen() 함수를 이용하여 길이를 측정해서 30글자 이상이면 30글자로 잘라버리는 경우가 많습니다. 위의 경우처럼 11이라는 길이를 통해서 10글자로 줄여버리면 '마' 글자의 절반이 잘려나가는 꼴이 되어 마지막 글자가 깨지는 결과가 나타날 수 있습니다.

strops

[PHP 4, PHP 5]

int strpos (string haystack, string needle [, int offset])

문자열이 처음 나타나는 위치를 반환한다.

인자	자료형	설명	비고
haystack	string	대상 문자열	필수
needle	string	찾을 문자열	필수
offset	int	haystack에서 찾기 시작할 위치	옵션

[표 4-57] strops 함수

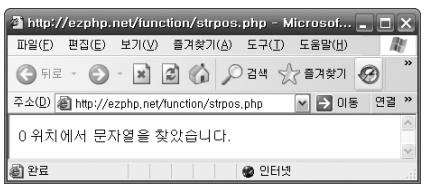
strpos() 함수는 어떤 문자열에서 특정 문자열의 위치를 찾으려고 사용하는 함수입니다. 첫 번째 인자인 haystack에서 needle을 찾고, 만약 찾으면 그 위치를 반환합니다. 여기서 주의할 점은 첫 번째 글자와 일치하는 경우 결과가 0이 된다는 것입니다. 그런데 needle을 찾지 못하는 경우의 결과가 'FALSE'라는데 문제가 있습니다. 자칫 0을 FALSE로 오인할 수 있는 소지가 있기 때문입니다. 그래서 문자열을 찾았는지에 대한 여부를 확인하기 위해서 리턴값을 비교하려면 '===' 연산자를 사용해야 합니다.

옵션 인자인 offset을 지정하면 지정된 위치부터 needle을 찾기 시작합니다.

[예제 4-52] trpos() 함수를 이용하여 특정 문자열을 포함하는지 검사한다.

```
1  <?
2     $haystack = "abcdef";
3     $needle = "abc";
4
5     $pos = strpos($haystack, $needle);
6
```

```
7     if ($pos === false)
8     {
9       echo "문자열을 찾지 못했습니다.";
10    }
11    else
12    {
13      echo "{$pos} 위치에서 문자열을 찾았습니다.";
14    }
15  ?>
```



[그림 4-41] 예제 4-52의 실행결과

strpos() 함수 이외에 대소문자를 구분하지 않고 문자열을 찾아주는 stripos() 함수와 문자열을 뒷부분부터 시작해서 찾아주는 strrpos() 함수 그리고 대소문자 구별 없이 문자열의 뒷부분에서부터 찾아주는 strripos() 함수가 있습니다. 다른 3가지 함수들도 모두 찾는 위치와 대소문자 구별 여부만 다를 뿐 사용법은 같습니다.

strstr, strchr

[PHP 4, PHP 5]

string strstr (string haystack, string needle)

문자열이 처음 나타나는 위치를 찾아 그 위치부터 끝까지의 문자열을 반환한다.

인자	자료형	설명	비고
haystack	string	대상 문자열	필수
needle	string	찾을 문자열	필수

[표 4-58] strstr, strchr 함수

strstr() 함수는 어떤 문자열에서 특정 문자열을 찾으면 그 위치부터 남은 문자열의 끝까지를 반환해주는 함수입니다. 만약 haystack에서 needle을 찾지 못하면 FALSE를 반환합니다. strchr() 함

수는 strstr() 함수의 별칭입니다.

[예제 4-53] strstr() 함수를 이용하여 이메일에서 도메인을 분리한다.

```
<?
  $email = 'happybrown@naver.com';
  $domain = strstr($email, '@');
  echo $domain; //@naver.com이 출력됨
?>
```

strstr() 함수를 이용해서 특정 문자열을 포함하는지를 판단할 수도 있습니다. 그러나 단순히 문자열을 포함하는지를 판단하기 위해서라면 더 빠르고 메모리를 적게 소비하는 strpos() 함수를 사용하기 바랍니다.

strstr() 함수 이외에 대소문자 구별을 하지 않고 문자열을 찾아주는 stristr() 함수와 뒤에서부터 찾아주는 strrchr() 함수가 존재합니다. 모두 사용법은 같습니다.

substr

[PHP 4, PHP 5]

string substr (string str, int start [, int length])

문자열의 일부를 반환한다.

인자	자료형	설명	비고
str	string	대상 문자열	필수
start	int	잘라낼 문자열의 시작 위치	필수
length	int	잘라낼 문자열의 길이	옵션

[표 4-59] 함수

substr() 함수는 어떤 문자열에서 특정 부분을 잘라내어 그 값을 반환받고자 할 때 사용하는 함수입니다. 문자열의 start 위치에서 length 길이만큼의 문자열을 잘라냅니다. 만약 옵션 인자인 length를 지정하지 않으면 start 위치부터 문자열의 끝까지를 반환합니다.

[예제 4-54] substr() 함수를 이용하여 문자열을 잘라낸다.

```
1  <?
2    echo substr ("abcdef", 1); // bcdef
3    echo substr ("abcdef", 1, 3); // bcd
4
5    echo substr ("abcdef", -1); // f
6    echo substr ("abcdef", -2); // ef
7    echo substr ("abcdef", -3, 1); // d
8
9    echo substr("abcdef", 1, -1); // bcde
10 ?>
```

2행은 length를 지정하지 않아서 1에서부터 문자열의 끝까지를 반환합니다. 그러나 3행은 length로 3을 지정하여 세 글자만을 반환합니다. 5행에서 7행까지는 start 값이 음수인 경우로 start 값을 음수로 지정하면 문자열의 뒤에서부터 거꾸로 위치를 찾아갑니다. 9행처럼 length가 음수가 되면 잘린 문자열의 마지막에서부터 한 글자씩 사라지는 값을 반환하게 됩니다.

trim

[PHP 4, PHP 5]

string trim (string str [, string charlist])

문자열의 앞뒤에 있는 공백을 제거한다.

인자	자료형	설명	비고
str	string	대상 문자열	필수
charlist	string	제거하고자 하는 문자의 목록	옵션

[표 4-60] trim 함수

trim() 함수는 ltrim() 함수와 rtrim() 함수를 동시에 수행한 것과 같이 문자열의 처음과 끝에 존재하는 공백을 제거합니다. 그래서 ltrim() 함수와 rtrim() 함수를 사용하기보다는 trim() 함수를 일반적으로 사용합니다. 게시판이나 방명록에서 사용자가 입력한 값의 공백을 제거하는 데 매우 유용하게 사용할 수 있습니다.

Section 04

기타 유용한 함수

PHP는 매우 많은 내장 함수를 지원하는데 이것은 PHP의 최대 장점이기도 합니다. 각종 데이터베이스 관련 함수, 수학 함수, PDF, 그림을 그리기 위한 GD 함수 등등 앞에서 다루지 않은 수많은 함수가 있습니다. 그중에서도 알아두면 피가 되고 살이 되는 함수들만 요약해서 다루어 보도록 하겠습니다.

분류	함수 이름	기능
메일 함수	mail	메일을 보낸다.
수학 함수	abs	절대값
	ceil	올림
	floor	내림
	pow	거듭제곱
	rand	정수형의 난수를 생성한다.
	mt_rand	향상된 난수를 생성한다.
	round	반올림
URL 함수	base64_encode	데이터를 MIME base 64로 인코딩한다.
	base64_decode	base64로 인코딩된 데이터를 디코딩한다.
	urlencode	문자열을 URL 인코딩한다.
	urldecode	URL 인코딩된 문자열을 디코딩한다.
변수 관련 함수	empty	변수가 비어있는지 확인
	isset	변수가 존재하는지 확인
	unset	변수를 제거
	is_array	변수가 배열인지 확인

[표 4-61] 기타 유용한 함수

mail

[PHP 4, PHP 5]0

bool mail (string to, string subject, string message [, string add_headers,[, string add_params]])

메일을 보낸다.

인자	자료형	설명	비고
to	string	받는 사람 메일 주소	필수
subject	string	메일 제목	필수
message	string	메일 내용	필수
add_headers	string	추가할 메일 헤더	옵션
add_params	string	추가할 파라미터	옵션

[표 4-62] mail 함수

mail() 함수는 시스템에 설치된 메일 서버(sendmail, qmail 등)를 이용하여 지정한 수신자에게 메일을 보냅니다. 메일에 성공하면 TRUE를, 실패하면 FALSE를 반환합니다.

[예제 4-55] mail() 함수를 이용하여 메일을 보낸다.

```
1  <?
2     $subject = "책 정말 좋군요.";
3     $message = "라면 받침대로 쓰기에...";
4
5     $to = "moratorium@gmail.com";
6     $from = "your@mail.com";
7
8     $header = "From: {$from}\r\nReply-To: {$from}\r\n";
9
10    mail($to, $title, $message, $header);
11 ?>
```

mail() 함수의 기본 인자에는 보내는 사람 항목이 없습니다. 그래서 보내는 사람의 메일 주소를 지정하려면 위와 같이 헤더를 추가해주어야 합니다. 또한 헤더 정보를 수정하면 HTML 형식의 메일이나 첨부 파일을 추가할 수 있습니다.

mail() 함수를 이용하면 매우 간편하게 메일 보내기를 구현할 수 있습니다. 그러나 메일을 사용할 수 있는 기본적인 환경이 시스템에 먼저 구성되어 있어야 합니다. PHP의 mail() 함수는 기본적으로 sendmail과 연동됩니다. 따라서 메일 함수를 테스트하려면 시스템에 sendmail을 설치해야 합니다.

abs

[PHP 4, PHP 5]

number abs (mixed number)

number의 절대값을 구한다.

인자	자료형	설명	비고
number	mixed	절대값을 구하고자 하는 수	필수

[표 4-63] abs 함수

abs() 함수는 절대값을 구하는 함수입니다. 만약 정수형의 값을 입력하면 결과로 정수형의 값이 반환되고 실수형을 입력하면 실수형의 결과를 반환합니다.

[예제 4-56] abs() 함수를 이용하여 절대값을 구한다.

```
1  <?
2     echo abs(3); // 3
3     echo abs(-3); // 3
4     echo abs(-3.3) // 3.3
5  ?>
```

ceil

[PHP 4, PHP 5]

float ceil (float value)

소수점 이하를 올림한다.

인자	자료형	설명	비고
value	float	올림하고자 하는 수	필수

[표 4-64] ceil 함수

ceil() 함수는 소수점을 올림하여 정수를 반환합니다. 정수를 반환함에도 불구하고 리턴값의 형식이 float인 이유는 integer형보다 float형의 범위가 더 크기 때문입니다. 즉, integer 범위를 넘어서는

값을 올림하였을 경우를 고려하기 위함입니다.

[예제 4-57] ceil() 함수를 이용하여 올림한다.

```
1  <?
2    echo ceil(3.00); // 3
3    echo ceil(3.3); // 4
4    echo ceil(-3.3) // -3
5  ?>
```

3.00의 경우에는 소수점 이하 자리가 존재하지 않기 때문에 올림값이 그대로입니다. 3.3의 경우에는 소수점이 존재하므로 올림하여 4가 되고 -3.3의 경우 소수점이 올림되어 -3이 됩니다. -3보다 -3.3이 더 작은 수이기 때문에 올림하면 -3이 됩니다.

floor

[PHP 4, PHP 5]

float floor (float value)

소수점 이하를 내림한다.

인자	자료형	설명	비고
value	float	내림하고자 하는 수	필수

[표 4-65] floor 함수

floor() 함수는 ceil() 함수와 반대로 소수점을 내림한 정수를 반환합니다. ceil() 함수와 마찬가지의 이유로 반환값은 float형을 사용합니다.

[예제 4-58] floor() 함수를 이용하여 내림한다.

```
1  <?
2    echo floor(3.00); // 3
3    echo ceil(3.3); // 3
4    echo ceil(-3.3) // -4
5  ?>
```

3.00을 내림하면 소수점이 없으니 버릴 값도 없어서 그대로 3을 유지합니다. 3.3은 소수점이 존재하여 소수점 이하를 버리게 되면 3이 됩니다. 마지막으로 −3.3은 −4보다 큰 수이나 내림을 해서 결과가 −4가 됩니다.

pow

[PHP 4, PHP 5]

number pow (number base, number exp)

수를 거듭제곱한다.

인자	자료형	설명	비고
base	number	거듭제곱하고자 하는 수	필수
exp	number	몇 승	필수

[표 4-66] pow 함수

pow() 함수는 거듭제곱을 계산하는 함수입니다. 단순히 값을 여러 번 곱하면 거듭제곱을 구할 수 있으나 곱하는 횟수가 크면 pow() 함수를 이용하여 거듭제곱을 구하는 것이 낫습니다. base 값을 exp 승 합니다.

[예제 4-59] pow() 함수를 이용하여 거듭제곱을 한다.

```
1  <?
2    echo pow(1, 1000); // 1
3    echo pow(-1, 2); // 1
4    echo pow(2, 3); // 8
5  ?>
```

rand

[PHP 4, PHP 5]

int rand ([int min] , int max)

정수형 난수를 생성한다.

인자	자료형	설명	비고
min	int	난수의 최소값	옵션
max	int	난수의 최대값	필수

[표 4-67] rand 함수

rand() 함수는 정수형 난수를 생성하는 함수입니다. 인자 없이 사용하면 0부터 RAND_MAX 사이의 임의의 난수를 생성합니다. 몇몇 플랫폼(윈도우 등)에서는 RAND_MAX 값이 32768입니다. 만약 32768보다 큰 수를 원하는 경우에는 min, max를 수정하거나 향상된 난수 생성 함수인 mt_rand() 함수를 사용하면 됩니다.

PHP 4.2.0 이전에서는 난수를 생성하기 위해서 난수값 생성기를 초기화해야 했습니다. 왜냐하면 난수값 생성기를 초기화하지 않을 때 새로 고침을 해보면 매번 동일한 난수값을 생성하기 때문입니다. 그러나 PHP 4.2.0부터는 자동으로 초기화되기 때문에 별도로 난수 생성기를 초기화하지 않아도 됩니다.

[예제 4-60] rand() 함수를 이용하여 난수를 생성한다.

```
1  <?
2    echo rand(); // 205
3    echo rand(1, 10); // 10
4    echo rand(40000, 50000); // 44732
5  ?>
```

[예제 4-60]의 결과는 항상 다릅니다. 따라서 위와 같은 결과가 나오지 않았다고 해서 전혀 좌절할 필요가 없습니다. 만약 위와 같은 결과를 얻었다면 저와 굉장한 인연을 가지신 분이라 생각됩니다. 경우의 수가 RAND_MAX * 10 * 10000 정도 되니까 대략 같은 결과를 낼 확률이 32억7680만 분의 1이 될 것입니다.

실제로 예상되는 값은 다음과 같습니다. 첫 번째 rand()를 통해서 생성한 수는 0부터 RAND_MAX 사이의 임의의 값이 될 것입니다. 두 번째는 1과 10 사이의 어느 한 값이 될 것입니다. 마지막으로 세 번째는 4만에서 5만 사이의 임의의 값이 출력될 것입니다. 이 함수를 이용하여 로또 번호 생성기를 한번 만들어 보는 것도 좋을 것 같네요.

mt_rand

[PHP 4, PHP 5]

int mt_rand ([int min] , int max)

향상된 정수형 난수를 생성한다.

인자	자료형	설명	비고
min	int	난수의 최소값	옵션
max	int	난수의 최대값	필수

[표 4-68] mt_rand 함수

mt_rand() 함수는 rand() 함수보다 향상된 난수를 생성합니다. rand() 함수가 libc 난수 생성기를 사용하여 느렸던 것에 비해 mt_rand() 함수는 새로운 알고리즘을 사용하여 4배나 더 빠르게 난수를 생성할 수 있습니다.

[예제 4-61] mt_rand() 함수를 이용하여 난수를 생성한다.

```
1  <?
2    echo mt_rand(); // 1930233306
3    echo mt_rand (1, 10); // 9
4    echo mt_rand (40000, 50000); // 43758
5  ?>
```

mt_rand() 함수도 rand() 함수와 마찬가지로 PHP 4.2.0부터 난수값 생성기를 초기화할 필요가 없어졌습니다.

round

[PHP 4, PHP 5]

float round (float val [, int precision])

실수를 반올림한다.

인자	자료형	설명	비고
val	float	반올림할 실수	필수
precision	int	반올림할 소수점 아래의 자리 수	옵션

[표 4-69] round 함수

round() 함수는 실수를 반올림합니다. 두 번째 인자인 precision을 사용하면 소수점 아래의 몇 자리에서 반올림을 할 것인지를 지정할 수 있습니다.

[예제 4-62] round () 함수를 이용하여 반올림한다.

```
1  <?
2     echo round(3.3); // 3
3     echo round(3.5); // 4
4     echo round(1.23456, 3); //1.235
5     echo round(123456, -3); //123000
6  ?>
```

precision에 음수를 지정하는 때도 예제의 경우처럼 사용할 수 있습니다.

base64_encode

[PHP 4, PHP 5]

string base64_encode (string data)

MIME 규약에 따라 데이터를 base64로 인코딩한다.

인자	자료형	설명	비고
data	string	인코딩할 문자열	필수

[표 4-70] base64_encode 함수

base64_encode() 함수는 MIME 규약에 따라 데이터를 base64로 인코딩한 결과를 반환합니다. MIME(Multipurpose Internet Mail Extensions)은 전자메일의 표준 형식입니다. 전자메일은 7비트 ASCII 문자를 사용하여 전송되기 때문에 한글과 같은 8비트 이상의 코드를 사용하는 문자나 바이너리 파일은 올바르게 전송할 수가 없습니다. 이러한 문제를 해결하기 위해서 MIME은 ASCII가 아닌 문자 인코딩을 이용하여 영어가 아닌 다른 언어로 된 전자메일을 처리할 수 있게 합니다. 8비트를

7비트로 변환하는 이러한 인코딩 방식 중 하나가 바로 base64입니다. base64를 사용하면 데이터의 손실 없이 전송할 수 있지만 원래의 문자열보다 약 33% 정도 더 길어지는 단점이 있습니다.

[예제 4-63] base64_encode () 함수로 문자열을 인코딩한다.

```
1  <?
2    $text = '행복한 브라운';
3    echo base64_encode($text); //x+C6ucfRILrqtvO/7g==
4  ?>
```

base64_decode

[PHP 4, PHP 5]

string base64_encode (string data)

MIME 규약에 따라 인코딩된 데이터를 디코딩한다.

인자	자료형	설명	비고
data	string	디코딩할 문자열	필수

[표 4-71] base64_decode 함수

base64_decode() 함수는 base64로 인코딩된 데이터를 디코딩한 결과를 반환합니다.

[예제 4-64] base64_decode () 함수로 인코딩된 문자열을 디코딩한다.

```
1  <?
2    $text = 'x+C6ucfRILrqtvO/7g==';
3    echo base64_decode($text); //행복한 브라운
4  ?>
```

urlencode

[PHP 4, PHP 5]
string urlencode (string str)
문자열을 URL 인코딩한다.

인자	자료형	설명	비고
str	string	인코딩할 문자열	필수

[표 4-72] urlencode 함수

urlencode() 함수는 URL을 이용하여 데이터를 전송하고자 할 때 URL로 인코딩한 문자열을 반환합니다.

[예제 4-65] urlencode() 함수로 문자열을 인코딩한다.

```
1  <?
2    $text = '가나12';
3    echo urlencode($text); //%B0%A1%B3%AA12
4  ?>
```

urldecode

[PHP 4, PHP 5]
string urldecode(stringstr)
문자열을 URL 디코딩한다.

인자	자료형	설명	비고
str	string	디코딩할 문자열	필수

[표 4-73] urldecode 함수

urldecode() 함수는 URL로 인코딩된 문자열을 원문 형태로 디코딩한 문자열을 반환합니다.

[예제 4-66] urldecode() 함수로 URL 인코딩한 문자열을 디코딩한다.

```
1  <?
2    $text = '%B0%A1%B3%AA12';
3    echo urldecode($text); //가나12
4  ?>
```

empty

[PHP 4, PHP 5]

bool empty(mixedvar)

변수가 비어 있는지 확인한다.

인자	자료형	설명	비고
var	mixed	확인할 변수	필수

[표 4-74] empty 함수

empty() 함수는 인자로 들어온 변수를 확인하여 변수가 비어 있는지를 확인합니다. 빈 문자열, 0, 문자열 "0", NULL, FALSE, 빈 배열 그리고 클래스에서 값이 없이 선언된 변수가 비어 있는 것으로 간주됩니다.

[예제 4-67] empty() 함수로 변수가 비어 있는지를 확인한다.

```
1  <?
2    $text = '0';
3    if (empty($text)) echo '변수는 비어있는 상태입니다.'; //출력
4    else echo '변수는 $text 값을 가지고 있습니다.';
5  ?>
```

isset

[PHP 4, PHP 5]

bool isset(mixedvar[,mixedvar[,…]])

변수가 설정되어 있는지 확인한다.

인자	자료형	설명	비고
var	mixed	확인할 변수	필수

[표 4-75] isset 함수

isset() 함수는 변수가 어떤 값으로 설정되어 있는지를 확인합니다. empty() 함수는 0의 값이나 문자열 "0"의 값을 가지고 있음에도 불구하고 변수가 비어 있다고 판단했지만 isset() 함수는 0이든 문자열 "0"이든 값이 설정되어 있기 때문에 TRUE를 반환합니다. 인자는 여러 개를 받을 수 있으며 여러 개의 인자를 확인하는 경우 모든 인자의 값이 설정되어 있어야 TRUE를 반환합니다.

[예제 4-68] isset() 함수로 변수가 설정되어 있는지를 확인한다.

```
1  <?
2    $text = '';
3    if (isset($text)) echo '변수는 설정되어 있습니다.'; //출력
4    else echo '변수가 설정되어 있지 않습니다.';
5   ?>
```

unset

[PHP 4, PHP 5]

void unset(mixedvar[,mixedvar[,…]])

주어진 변수를 제거한다.

인자	자료형	설명	비고
var	mixed	제거할 변수	필수

[표 4-76] unset 함수

unset() 함수는 인자로 주어진 변수를 제거하기 위해 사용합니다. 일반 변수나 배열 그리고 객체의 속성도 제거할 수 있습니다. 특히 배열의 경우에는 배열 전체를 제거하는 것뿐만 아니라 배열의 일부 원소를 제거할 수도 있습니다.

[예제 4-69] unset() 함수로 변수를 제거한다.

```
1  <?
2    $arr = array('brown','브라운');
3    unset($arr[0]);
4    echo $arr[0] . $arr[1];//브라운이 출력됨
5   ?>
```

is_array

[PHP 4, PHP 5]

bool is_array(mixedvar)

주어진 변수가 배열인지를 확인한다.

인자	자료형	설명	비고
var	mixed	확인할 변수	필수

[표 4-77] is_array 함수

is_array() 함수는 인자로 주어진 변수가 배열인지를 확인합니다. 변수가 배열이면 TRUE를 반환하며 이와 유사한 기능을 하는 함수로 변수가 소수인지를 확인하는 is_float(), 정수인지를 확인하는 is_int(), 문자열인지를 확인하는 is_string() 그리고 객체인지를 확인하는 is_object() 함수가 있습니다.

[예제 4-70] is_array() 함수로 변수가 배열인지를 확인한다.

```
1  <?
2    $arr = array('brown','브라운');
3    if(is_array($arr)) echo "주어진 변수는 배열입니다.";//출력
4    else echo "주어진 변수는 배열이 아닙니다.";
5   ?>
```

Chapter

05

배열과 객체

01. 배열
02. 객체

이 장에서는 PHP의 데이터형인 배열과 객체에 대해서 알아봅니다. 배열과 객체는 몇 가지 개념이나 사용법을 알아야만 완벽하게 사용할 수 있습니다. 특히 배열은 반드시 알아야 할 뿐만 아니라 능숙히 다룰 수 있어야 합니다. 객체는 객체지향 프로그래밍이라는 최근의 프로그래밍 트렌드에 맞춰서 PHP가 제공하는 개념으로 PHP 버전이 올라갈 때마다 점점 객체지향 언어의 모양을 갖추어 가고 있습니다. 최근 다양한 프로그램들이 객체로 제공되고 있어서 점차 그 사용이 늘어나고 있습니다. 그러나 소규모 프로그래밍에서 객체는 크게 의미가 없으므로 여전히 대부분의 PHP 프로그램에서는 객체를 사용하지 않고 있습니다. 그러나 객체를 완벽히 이해하지는 않더라도 객체지향 프로그래밍에 대한 개념은 반드시 알아두기 바랍니다.

Section 01

배열

배열은 PHP 프로그래밍에서 매우 유용하게 사용할 수 있는 자료형으로 다양한 배열 함수가 제공되어 다른 프로그래밍에서 배열을 다루는 것에 비해 매우 편리합니다. 배열을 한 마디로 표현하자면 "변수 보따리"라고 할 수 있습니다. 앞에서 배운 변수들은 대부분 하나의 변수형에 하나의 값을 저장했지만 보따리에 여러 가지의 물건을 넣을 수 있듯이 배열에도 하나 이상의 데이터를 저장할 수 있습니다. 심지어는 배열 속에 배열을 다시 저장할 수도 있습니다.

우선 배열이 어떻게 생겼는지 알아봅시다.

```
<?
    $a[0]="abc";
    $a[1] ="def";
    $b["foo"]=13;
?>
```

위의 소스 코드에서 보듯이 배열은 기존의 변수 이름 뒤에 "[인덱스]"를 덧붙인 모양을 갖습니다. 하나의 변수 이름에 여러 가지의 값을 갖다 보니 "배열의 몇 번째 값"과 같은 뜻의 인덱스를 이용하게 된 것입니다.

변수는 대부분 하나의 변수에 하나의 값을 갖는 게 기본인데 배열은 위처럼 a라는 변수에 abc와 def를 모두 가질 수 있습니다. 두 값의 구분은 [0], [1] 등으로 몇 번째에 저장되어 있는지를 표기하여 구분합니다. 이뿐만이 아니라 숫자 인덱스를 사용하지 않고 4번 줄과 같이 사용자가 지정한 구분자를 두어 배열의 값을 구분할 수도 있습니다.

배열의 생성

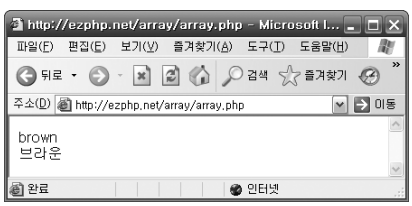
배열을 생성하는 방법은 두 가지가 있습니다. 하나는 array() 언어 구조(여기서는 함수라고 하지 않음)를 사용하여 생성하는 방법이고 다른 하나는 위에서 잠깐 언급하였듯이 대괄호 문법을 사용해서 생성하는 방법입니다.

array() 언어 구조를 이용한 배열 생성

[예제 5-1] array() 언어 구조를 이용하여 배열 생성

```
1  <?
2       $arr = array("name"=>"brown", 7=>"브라운");
3
4         echo $arr[name];
5       echo "<BR>";
6       echo $arr[7];
7  ?>
```

array() 언어 구조는 key=value의 형태로 콤마로 구분된 여러 개의 쌍을 등록할 수 있게 해줍니다. 위의 소스는 name이라는 이름의 키와 7이라는 키를 갖는 배열 $arr을 생성하는 코드입니다. 이러한 배열을 연관 배열이라고 합니다. 그리고 연관 배열은 키를 이용하여 배열에 저장된 값에 접근할 수 있습니다.



[그림 5-1] 예제 5-1~ 3의 실행결과

만약 키를 등록하지 않고 값만 부여한다면 [0], [1]과 같은 순서의 인덱스를 통하여 배열에 접근할 수 있습니다. 이러한 배열을 인덱스 배열이라고 합니다.

[예제 5-2] 인덱스 배열

```
1  <?
2       $arr = array("brown", "브라운");
3
4         echo $arr[0];
5        echo "<BR>";
6        echo $arr[1];
7  ?>
```

[예제 5-2]의 결과는 [예제 5-1]과 같습니다.

이처럼 C나 자바같은 프로그래밍 언어와는 다르게 PHP에서의 배열은 인덱스 배열과 연관 배열을 지원합니다. 간혹 이러한 차이 때문에 다른 프로그래밍 언어에서 지원하는 인덱스 배열이 보다 빠르고 연관 배열이 더 느리다고 생각하는 경우가 많으나 실질적으로 PHP에서 이 두 배열 방식에 대한 차이는 전혀 없습니다. 오히려 매우 직관적인 연관 배열을 쉽게 사용할 수 있어서 매우 편리하면서도 유용하게 배열을 사용할 수 있습니다.

단, 배열을 생성할 때 키는 정수나 문자열의 형식을 따라야 합니다. 그러나 배열의 값은 PHP의 어떤 변수형이라도 가능합니다. 즉, 배열의 값으로 배열을 등록할 수 있습니다.

[예제 5-3] 연관 배열

```
1  <?
2    $arr = array("name"=>"brown", "브라운");
3
4    echo $arr[name];
5    echo "<BR>";
6    echo $arr[0];
7  ?>
```

array() 언어 구조를 이용하면 위의 두 가지 방법을 섞어서 사용할 수도 있습니다. [예제 5-3]과 같이 key=value 쌍과 함께 value 값만 지정하더라도 정상적으로 배열이 생성됩니다. 단, 키를 지정하지 않은 부분에 대해서는 인덱스가 '0'부터 순서대로 지정됩니다.

[예제 5-3]의 결과는 [예제 5-1]과 같습니다.

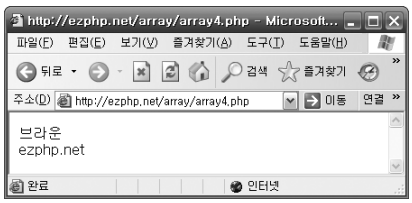
위의 예제에서 인덱스를 지정하지 않으면 자동으로 '0'부터 인덱스가 지정된다고 하였습니다. 그렇다면 인덱스를 지정하면 어떻게 될까요?

[예제 5-4] array() 언어 구조를 이용하여 배열 생성

```
1  <?
2    $arr = array("name"=>"brown", 5=>"브라운", "ezphp.net");
3
4    echo $arr[5];
5    echo "<BR>";
6    echo $arr[6];
7  ?>
```

위의 예제에서처럼 두 번째 원소에 인덱스를 0이 아닌 5로 지정했습니다. 거기다 0에서 4까지는 건너뛰었습니다. 위와 같이 중간에 숫자를 인덱스로 하는 원소가 있으면 그 후에 인덱스를 설정하지

않은 모든 원소는 그 번호에 이어서 순서대로 인덱스가 설정됩니다.



[그림 5-2] 예제 5-4의 실행결과

대괄호 문법을 이용한 배열 생성

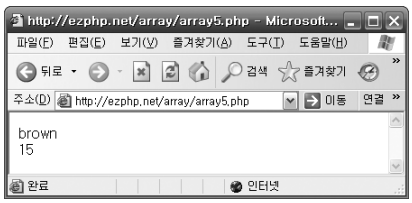
array() 언어 구조를 이용하지 않고 바로 대괄호를 이용하여 배열을 생성할 수도 있습니다. 이는 PHP의 유연한 변수 정책에 의한 것으로 보입니다. PHP에서는 프로그래머의 편의를 위해서 배열에 대한 매우 유연한 동작을 지원합니다. 대표적으로 변수를 선언하지 않아도 된다는 것이나 변수형이 다른 값을 대입하거나 추가해도 알아서 처리해주는 것 등을 말합니다. 대괄호 문법은 대괄호를 사용하여 직접 키를 지정하거나 혹은 키를 지정하지 않고 배열의 값을 생성하거나 수정하는 방법을 말합니다.

```
$arr[키] = 값;
$arr[] = 값;
```

대괄호 안에 키를 지정하지 않으면 array() 언어 구조에서 배열을 생성한 것과 마찬가지로 0부터 인덱스가 자동으로 지정됩니다. 이 방법은 배열을 생성하는 것뿐만 아니라 이미 생성된 배열에 값을 추가하거나 수정할 수도 있습니다. 앞서 array() 언어 구조를 이용하는 것보다 훨씬 직관적으로 사용할 수 있어서 매우 유용합니다. 그러나 여러 개의 원소를 추가하고자 할 때는 일일이 추가해주어야 하기 때문에 이런 경우 대괄호 문법을 사용하지 말고 array() 언어 구조를 사용하는 것이 좋습니다.

[예제 5-5] 대괄호 문법을 이용한 배열 생성

```
1  <?
2    $arr['name'] = 'brown';
3    $arr[] = 15;
4
5    echo $arr['name'];
6    echo "<BR>";
7    echo $arr[0];
8  ?>
```



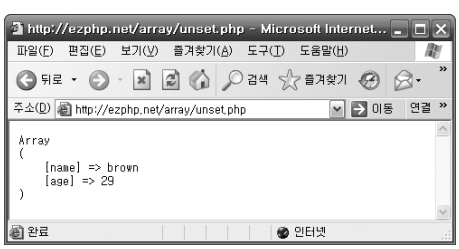
[그림 5-3] 예제 5-5의 실행결과

배열의 제거

배열을 생성하고 추가하여 사용하다가 배열을 메모리에서 제거하고자 한다거나 특정 원소를 제거하고자 할 때에는 일반적인 변수 제거 함수인 unset() 함수를 사용합니다.

[예제 5-6] 배열과 배열의 원소 제거

```
1  <?
2    $arr = array('name'=>'brown', 'age'=>29, 'sex'=>'male');
3
4    echo "<pre>";
5    //배열의 한 원소 제거
6    unset($arr['sex']);
7    print_r($arr);
8
9    //배열 전체 제거
10   unset($arr);
11   print_r($arr);
12   echo "</pre>";
13 ?>
```



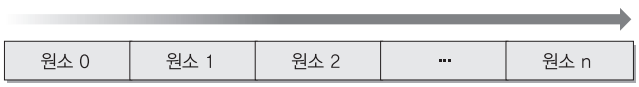
[그림 5-4] 예제 5-6의 실행결과

위의 예제를 보면 unset() 함수를 이용하여 어떻게 배열의 요소를 제거하고 또한 배열 전체를 제거하는지 알 수 있습니다. 여기서 print_r() 함수는 배열의 요소를 위의 결과와 같은 형식으로 출력해 주는 함수입니다.

다차원 배열

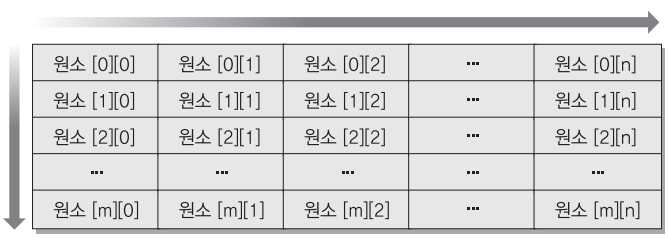
배열은 변수의 보따리라고 말씀드렸습니다. 큰 보따리 안에 작은 보따리가 여러 개 들어갈 수 있듯이 배열 속에 배열을 저장할 수 있습니다. 이렇게 배열 안에 배열이 있는 것을 다차원 배열이라고 합니다.

1차원 배열은 우리가 지금까지 다루어 온 배열입니다. 1차원은 직선과 같이 길이라는 개념만 존재하는 차원입니다. 그래서 아래 그림과 같이 가로(사실 방향의 의미는 없습니다.)로 하나씩 늘어가는 배열이 1차원 배열입니다.



[그림 5-5] 1차원 배열

2차원 배열은 우리가 자주 사용하는 표와 같습니다. 2차원은 x축과 y축이 존재하기 때문에 넓이라는 개념이 존재합니다.



[그림 5-6] 2차원 배열

3차원 배열은 2차원 배열이 여러 겹 쌓여 있는 배열입니다. 우리가 살아가는 공간처럼 부피가 존재하는 차원이 바로 3차원입니다.

[그림 5-7] 3차원 배열

그럼 본격적으로 2차원 배열을 만들어 봅시다.

매우 식상하지만 2차원 배열을 배울 때 가장 많이 사용하는 예를 들어보도록 하겠습니다. 바로 이름하여 국영수 배열입니다. 일단은 과목이나 학생의 이름을 제외하고 생각해 봅시다.

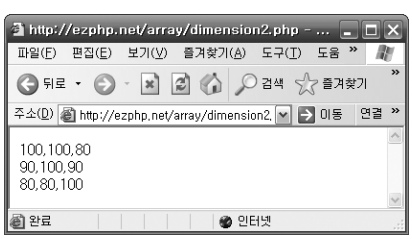
100	100	80
90	100	90
80	80	100

위의 표에는 과목과 학생의 이름이 지정되어 있지 않으므로 왼쪽에서 오른쪽으로 또한 위에서 아래로 순서대로 번호가 매겨지게 됩니다. 이를 배열로 표현하면 다음과 같습니다.

[예제 5-7] 2차원 배열 만들기

```
<?
   $arr = array( array(100,100,80),
       array(90,100,90),
       array(80,80,100)
      );

   echo $arr[0][0] .','.$arr[0][1] .','.$arr[0][2];
   echo '<BR>';
   echo $arr[1][0] .','.$arr[1][1] .','.$arr[1][2];
   echo '<BR>';
   echo $arr[2][0] .','.$arr[2][1] .','.$arr[2][2];
?>
```

[예제 5-7]에서 볼 수 있듯이 배열 속에 배열이 저장되는 구조입니다.



[그림 5-8] 2차원 배열 만들기

표의 값을 이처럼 2차원 배열에 저장할 수 있습니다. 그러나 과목 이름과 학생의 이름이 없어서 누가 어떤 과목의 점수를 받았는지 알기가 힘이 듭니다. 우선 다음과 같이 과목 이름을 추가하여 어떤

과목에 대한 점수인지를 알 수 있도록 해봅시다.

국어	영어	수학
100	100	80
90	100	90
80	80	100

각각 추가된 과목 이름을 바탕으로 배열을 생성해보면 다음과 같습니다.

[예제 5-8] 2차원 배열 만들기

```
<?
   $arr = array( array(  '국어' => 100,
                         '영어' => 100,
                         '수학' => 80
                      ),
                 array(  '국어' => 90,
                         '영어' => 100,
                         '수학' => 90
                      ),
                 array(  '국어' => 80,
                      '영어' => 80,
                      '수학' => 100
                      )
               );

   echo $arr[0]['국어'] .','.$arr[0]['영어'] .','.$arr[0]['수학'];
   echo '<BR>';
   echo $arr[1]['국어'] .','.$arr[1]['영어'] .','.$arr[1]['수학'];
   echo '<BR>';
   echo $arr[2]['국어'] .','.$arr[2]['영어'] .','.$arr[2]['수학'];
?>
```

배열의 원소로 추가하기 위해서 위와 같이 키를 등록하여 연관 배열을 만들어 줍니다. 그러면 키(국어, 영어, 수학)를 통해서 값에 접근할 수 있게 됩니다.

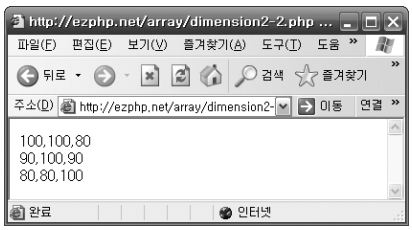
[그림 5-9] 예제 5-8의 실행결과

이제 학생의 이름까지 추가하여 국영수 배열을 완성해 봅시다.

	국어	영어	수학
이태희	100	100	80
박희선	90	100	90
김동건	80	80	100

[예제 5-9] 2차원 배열 만들기

```
<?
   $arr = array(  '이태희' =>
                  array('국어' => 100,
                        '영어' => 100,
                        '수학' => 80
        ),
                  '박희선' =>
                  array('국어' => 90,
                        '영어' => 100,
                        '수학' => 90
        ),
                  '김동건' =>
                  array('국어' => 80,
                        '영어' => 80,
                        '수학' => 100
        )
   );

   echo $arr['이태희']['국어'].','.$arr['이태희']['영어'].','.$arr['이태희']['수학'];
   echo '<BR>';
   echo $arr['박희선']['국어'].','.$arr['박희선']['영어'].','.$arr['박희선']['수학'];
   echo '<BR>';
   echo $arr['김동건']['국어'].','.$arr['김동건']['영어'].','.$arr['김동건']['수학'];
?>
```



[그림 5-10] 예제 5-9의 실행결과

[예제 5-9]와 같이 연관 배열을 이용하면 직관적이면서 아주 쉽게 배열 값을 저장하거나 읽어 올 수 있습니다. 그러나 과목의 수가 수십 개가 된다든가 학생의 수가 매우 많이 늘어나면 연관 배열을 이용하는 방법은 한계가 있습니다. for와 같은 루프를 이용하여 배열의 원소에 접근하고자 할 때는 연관 배열을 사용하지 않고 인덱스 배열을 사용하는 것이 매우 편리합니다.

3차원 배열은 2차원 배열과 마찬가지로 $arr[ ][ ][ ]와 같은 방식으로 생성하거나 접근할 수 있습니다. 이와 마찬가지로 4차원 5차원 등등 자신이 상상할 수 있는 차원만큼 배열의 차원을 늘려갈 수 있습니다. 그러나 우리가 3차원의 세계에 살고 있기 때문에 4차원 이상을 생각하기가 쉽지 않습니다. 만약 그 이상의 차원을 상상할 수 있는 능력이 있더라도 프로그램을 그와 같이 작성해서는 안 됩니다. 왜냐하면 프로그램의 작성에서 가장 중요한 것 중 하나가 "얼마나 알기 쉽게 표현하느냐?" 이기 때문입니다. 매우 복잡한 수식으로 작성한 1줄짜리 코드보다는 누구나 알기 쉬운 10줄짜리 코드가 더 나은 경우가 많기 때문입니다. 두 경우에서 성능 차이가 극히 차이 나지 않는 이상 천재 프로그래머나 이해할 수 있는 1줄짜리 코드는 쓸모가 없습니다. 그래서 매우 특별한 경우를 제외하고는 상상하기 힘든 4차원 이상의 배열은 사용하지 않습니다.

여기서 잠깐

연관 배열에 접근함에 있어서 키 값을 기입할 때 $arr[brown]과 같이 사용할 수 있습니다. 그러나 PHP에서는 $arr['brown']으로 사용하기를 권장합니다. 필자를 포함하여 수많은 프로그래머가 연관 배열에 접근할 때 따옴표나 작은따옴표를 잊어버리고 사용하는 경우가 많습니다. 그러나 따옴표를 사용하지 않는 방법은 아주 특별한 경우에 문제를 일으킬 수 있습니다. 그 특별한 경우는 해당 키가 상수로 정의되어 있을 때를 말합니다. 'brown'이라는 이름에 '조명진'이라는 값을 갖는 상수를 정의해두고 사용한다고 하였을 때 $arr[brown]은 $arr[조명진]으로 바뀌기 때문에 올바른 결과를 얻을 수 없게 됩니다. 상수를 자주 사용하는 프로그래머는 상수를 모두 대문자로 정의하여 쉽게 구분 지을 수 있게 하거나 배열을 사용하면서 주의를 기울일 필요가 있습니다.

배열의 연산

배열도 숫자처럼 연산자를 사용할 수 있습니다. 배열을 위한 특별한 연산자에 대해 알아보도록 하겠습니다.

연산자	사용법	결과
+	$a + $b	배열 $a와 $b의 합집합 (중복을 허용하지 않음)
==	$a == $b	배열 $a와 $b이 같은 원소를 갖는다면 참(true)
===	$a === $b	배열 $a와 $b가 원소와 그 순서까지 동일하면 참
!=	$a != $b	배열 $a와 $b의 원소가 다르면 참(true)
〈〉	$a 〈〉 $b	배열 $a와 $b의 원소가 다르면 참(true)
!===	$a !=== $b	배열 $a와 $b가 원소와 그 순서가 동일하지 않으면 참

[표 5-1] 배열의 연산자

배열의 연산자는 숫자처럼 사칙연산이 되는 것은 아니지만 배열을 더하거나 같은 원소들을 가졌는지 또한 두 배열이 완벽히 일치하는지 등을 검사할 수 있는 배열만의 특별한 연산자입니다. 더하기 연산자는 배열 $a 뒤에 배열 $b의 원소를 덧붙이는데 만약 같은 키를 갖는 원소가 있다면 그 원소는 추가되지 않습니다. 왜냐하면 동일한 키의 원소가 두 개 이상이 되면 배열의 원소에 접근하고자 할 때 어떤 원소의 값인지 알 수 없기 때문입니다.

배열에서는 특이하게 '같다'라는 의미가 두 가지입니다. 하나는 같은 원소를 갖고 있는 것이고 다른 하나는 같은 원소와 그 순서까지 동일한 경우입니다. 여기서 같은 원소를 갖는다는 의미는 키와 값이 모두 동일하지만 배열에 저장된 순서가 다를 수 있음을 의미합니다.

[예제 5-10] 배열의 등위 연산자

```
1  <?
2    $a = array(1, 2);
3    $b = array(2, 1);
4    if ($a == $b) echo '$a와 $b는 같은 원소를 갖는다.';
5    else echo '$a와 $b는 다른 원소를 갖는다.';
6  ?>
```

[예제 5-10]의 결과는 "다른 원소를 갖는다" 입니다. 두 배열 모두 1과 2라는 값을 갖고 있어서 배열의 원소가 같다고 생각할 수가 있습니다. 이 경우는 배열의 원소가 같은 것이 아니라 원소의 값이 같을 뿐 키가 다르므로 배열의 원소가 같다고 할 수 없습니다.

[예제 5-11] 배열의 등위 연산자

```
<?
  $a = array(1, 2);
  $b = array(1 => 2, 0 => 1);

  if ($a == $b) echo '$a와 $b는 같은 원소를 갖는다.';
  else echo '$a와 $b는 다른 원소를 갖는다.';
?>
```

[예제 5-11]과 같이 키와 값이 모두 같지만 순서가 다른 경우에 같은 원소를 갖는다고 말합니다. 상식적으로 인덱스가 1, 0이니까 순서대로 0, 1로 저장될 것 같지만 실제로는 입력한 순서 그대로 저장되어 버립니다. 그래서 원소는 동일하지만 순서가 다른 것입니다. 만약 [예제 5-12]와 같이 '===' 연산자를 이용하여 검사해보면 거짓으로 나타납니다.

[예제 5-12] 배열의 동일 연산자

```
<?
  $a = array(1, 2);
  $b = array(1 => 2, 0 => 1);

  if ($a === $b) echo '$a와 $b는 동일한 배열이다.';
  else echo '$a와 $b는 동일하지 않은 배열이다.';
?>
```

[예제 5-12]에서 참의 결과를 얻으려면 완벽히 동일한 원소를 가져야 합니다. PHP에서 배열의 인덱스가 반드시 순서대로 저장되는 것이 아니므로 생기는 현상입니다.

나머지 연산자들은 설명 드린 연산자의 역에 해당하는 것이므로 설명을 생략합니다.

배열의 정렬

PHP에서는 매우 간단하게 배열을 정렬할 수 있습니다. PHP에서는 기본적으로 다양한 배열 정렬 함수를 제공하고 있는데 퀵 정렬, 버블 정렬과 같은 다양한 정렬 알고리즘을 제공하는 것이 아니라 배열의 상황에 따른 정렬 함수를 제공합니다. 그중에서 몇 가지를 정리해 보았습니다.

함수	설명
sort	값을 기준으로 정렬하여 순서대로 인덱스 배열을 구성
rsort	값을 기준으로 내림차순 정렬하여 순서대로 인덱스 배열을 구성
asort	값을 기준으로 정렬하되 키와 값의 관계를 유지
arsort	값을 기준으로 내림차순 정렬하되 키와 값의 관계를 유지
ksort	키를 기준으로 정렬하되 키와 값의 관계를 유지
krsort	키를 기준으로 내림차순 정렬하되 키와 값의 관계를 유지
natsort	natural order 알고리즘을 사용하여 정렬하되 키와 값의 관계를 유지
natcasesort	대소문자 구분없이 narsort 실행

[표 5-2] 배열 정렬 함수

sort() 함수는 가장 기본적인 정렬 함수로 값을 기준으로 오름차순 정렬을 합니다. sort() 함수의 특징은 연관 배열을 정렬하면 키 값에 상관없이 정렬된 순서대로 인덱스 배열을 형성하게 된다는 것입니다.

[예제 5-13] sort 함수를 이용한 정렬

```
<?
   $fruits = array ('a'=>"lemon", 'b'=>"orange", "banana", "apple");
   sort ($fruits);

   echo "<pre>";
   print_r($fruits);
   echo "</pre>";
?>
```

sort() 함수를 이용하여 정렬하면 키와 값의 관계가 유지되지 않기 때문에 다음과 같이 무조건 순서대로 인덱스 배열을 형성하게 됩니다.

```
Array
(
    [0] => apple
    [1] => banana
    [2] => lemon
    [3] => orange
)
```

rsort() 함수는 sort() 함수와 동일하나 내림차순으로 배열을 정렬합니다. 이와 마찬가지로 비슷한 이름을 가지면서 'r'이 추가된 함수는 원래 정렬 함수와 기능이 같지만 모두 내림차순으로 정렬되는

함수입니다.

asort() 함수는 sort() 함수와 달리 키와 값의 연관 관계를 유지하면서 정렬을 합니다. asort에서 a는 연관 배열(associative array)을 뜻합니다.

[예제 5-14] asort 함수를 이용한 정렬

```
<?
   $fruits = array ('a'=>"lemon", 'b'=>"orange", "banana", "apple");
   asort ($fruits);

   echo "<pre>";
   print_r($fruits);
   echo "</pre>";

?>
```

[예제 5-14]의 결과는 다음과 같습니다.

```
Array
(
    [1] => apple
    [0] => banana
    [a] => lemon
    [b] => orange
)
```

sort() 함수로 정렬했을 때와 달리 키와 값이 여전히 연결되어 있음을 알 수 있습니다. arsort() 함수는 asort() 함수를 내림차순으로 정렬한 것입니다.

ksort() 함수는 값이 아닌 키를 기준으로 정렬합니다. 그래서 이름도 키 정렬(key sort)입니다.

[예제 5-15] ksort 함수를 이용한 정렬

```
<?
   $fruits = array ('a'=>"lemon", 'b'=>"orange", 'c'=>"banana", 'd'=>"apple");
   ksort ($fruits);

   echo "<pre>";
   print_r($fruits);
   echo "</pre>";
?>
```

키를 이용하여 정렬했으므로 결과는 다음과 같습니다.

```
Array
(
    [a] => lemon
    [b] => orange
    [c] => banana
    [d] => apple
)
```

앞서 sort(), asort() 함수는 값을 기준으로 정렬했으나 ksort() 함수는 결과에서도 알 수 있듯이 키 값을 기준으로 정렬합니다. 사람 이름을 키로 갖고 값으로 시험 점수를 갖는 배열이 있다면 사람 이름 순서대로 출력하고자 할 때가 있을 것입니다. 이럴 때 ksort() 함수를 이용하면 편리하게 정렬할 수 있습니다.

natsort() 함수는 natural order 알고리즘을 이용하여 정렬합니다. natural order 알고리즘은 문자열 정렬을 할 때 숫자 정렬이 제대로 되지 않는 것을 보완한 것입니다. 예를 들면 sort() 정렬은 apple1, apple2, …, apple11, apple12와 같이 값이 있을 경우에 정렬을 해보면 apple1, apple11, apple12, apple2, …와 같이 정렬됩니다. apple1 다음에는 apple2이지만 apple 다음에 오는 숫자를 숫자로 비교하지 않고 문자로 생각하고 정렬하기 때문에 apple11이 먼저 정렬되는 것입니다. 이러한 문제를 보완하고자 만든 정렬이 바로 natsort() 정렬입니다.

[예제 5-16] natsort 함수를 이용한 정렬

```
1  <?
2    $arr = array ("img12.png", "img10.png", "img2.png", "img1.png");
3    natsort($arr);
4
5    echo "<pre>";
6    print_r($arr);
7    echo "</pre>";
8  ?>
```

앞서 배운 정렬 함수로 정렬하면 img1.png, img10.png, img12.png, img2.png 순으로 정렬되어야 하지만 natsort() 함수로 정렬하면 다음과 같이 문자열 속의 숫자도 구분하여 정렬하고 있음을 알 수 있습니다.

```
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

이와 마찬가지로 natcasesort() 함수는 대소문자 구분없이 natrual order정렬을 합니다.

정렬 함수가 여기에 정리된 것 외에도 더 많이 존재합니다. 비록 많아서 뭐가 뭔지 잘 모르겠다고 생각하는 분이 계실지 모르겠으나 잘 생각해보면 규칙이 있는 것을 알 수 있습니다.

우선 배열에서 값으로 정렬하고자 할 때는 sort 함수를 사용하고 배열이 연관 배열이어서 키와 값의 관계를 유지시켜줘야 하면 asort 함수를 사용하면 됩니다. 그리고 값이 아니라 키를 기준으로 정렬하고자 할 때는 ksort를, 문자열에 숫자가 섞여 있는 값을 정렬하고자 할 때는 natsort 함수를 사용하면 됩니다. 또한 내림차순으로 정렬하고자 할 때는 앞서 배운 정렬 함수에 'r'이 추가된 함수를 사용하면 됩니다.

Section 02 객체

프로그래밍 기법 중에는 절차식 프로그래밍과 객체지향 프로그래밍이 있습니다. 우리가 지금까지 작성해온 프로그램들이 절차식 프로그래밍 기법을 사용한 것인데 절차식 프로그래밍은 기능에 초점을 맞추어 순차적으로 어떤 기능들을 수행해야 하는지를 표현해가는 방식을 말합니다. 즉, 어떤 작업을 하면서 수행해야 하는 기능을 함수로 만들고 그 기능들의 처리 순서를 고려하여 작성해둔 함수를 호출하는 방식입니다. 이에 반해 객체지향 프로그래밍은 기능이 아닌 데이터에 초점을 두고 객체라는 단위로 모든 처리를 표현해 가는 방법을 말합니다.

최근에는 객체지향 프로그래밍이 대세라고 불릴 만큼 주목받고 있으며, 근래에 주목받는 언어들은 대부분 객체지향 언어입니다. 이에 발맞추어 PHP에서도 앞서 PHP4에서부터 객체지향 프로그래밍 기법을 도입하였으나 실제로 부족한 점이 많이 있었고 PHP5가 되면서 보다 객체지향에 가까운 언어로 변모하게 되었습니다.

객체지향 프로그래밍이란 말은 말 자체에서부터 무슨 소린지 알 수 없는 포스를 갖고 있습니다. 시

작도 하기 전에 겁을 잔뜩 먹게 하는 객체지향이란 말은 "Object-Oriented"라는 말을 번역한 것인데 단순히 말해서 모든 사물을 객체로 바라본다는 의미입니다. 객체는 기능(method)과 속성(property)으로 구성되어 있습니다. 그러니까 모든 사물을 기능과 속성을 가진 객체로 생각하자는 것이 바로 객체지향 프로그래밍의 핵심입니다.

인간을 객체로 생각해 보겠습니다. 인간은 어떠한 기능을 할까요? 객체지향 프로그래밍에서 말하는 기능은 객체가 할 수 있는 동작 또는 행위를 의미합니다. 따라서 걷고 달리고 말하고 음식을 먹고 잠을 자는 등의 모든 동작이 바로 기능이 됩니다. 속성은 객체가 갖는 데이터로 인간의 경우 이름이나 나이, 주소 등등이 바로 속성이 됩니다. 바로 '이러한 기능과 속성을 갖는 것이 인간'이라고 인간을 정의하는 것을 인간 클래스(class)라고 합니다.

클래스는 판화와 같습니다. 사람 모양의 판화를 만들고 잉크를 발라서 똑같은 모양을 계속해서 찍어낼 수 있듯이 인간이란 클래스를 만들어 찍어내면 철수니 영희니 돌쇠니 하는 사람들이 나오게 되는 것입니다. 이렇게 클래스를 통해 만들어진 철수를 객체라고 합니다.

클래스를 정의하려면 우선 사물을 추상화(abstraction) 해야 합니다. 인간이라는 클래스를 정의하고자 할 때 프로그래머가 인간의 모든 기능과 속성들을 정의할 수가 없습니다. 너무나 당연하지만 인간을 표현하기에는 인간이 아는 지식은 터무니없이 부족합니다. 그뿐만이 아니라 그 부족한 지식마저도 매우 방대합니다. 그래서 인간을 정의하기가 쉽지 않습니다. 그래서 인간을 표현할 수 있게 중요하고 관계있는 부분만 분리해서 간결하고 이해하기 쉽게 정리하는 것을 바로 추상화라고 합니다.

인간을 간단히 추상화하여 인간 클래스를 만들어 보았습니다.

```
class Human {  //인간 클래스를 정의합니다.

        var $Name;
        var $Age;
        var $Height;
        var $Weight;

        function Eat ( $foods ) {
                //먹는 행위를 함수로 정의
                }
        function Walk ( $destination ) {
                //걷는 행위를 함수로 정의
                }
        function Work ( $job ) {
                //일하는 행위를 함수로 정의
                }
        function Talk ( $words ) {
                //말하는 행위를 함수로 정의
                }
}
```

간단히 이름, 나이, 키, 몸무게의 속성을 가지고 먹고 걷고 일하고 말하는 행위를 기능으로 정의하였습니다. 객체지향에 대한 개념이 머릿속에 그려지시나요? 이제 좀 더 깊이 배워봅시다.

생성자

인간이 태어나면 일단 이름을 정해주고 나이를 한 살 먹는 등 인간의 속성값이 초기화되어야 합니다. 이를 위해서 생성자(constructor)라는 것이 존재합니다. 생성자는 객체를 초기화하기 위해서 생겨난 것으로 일반적으로 클래스의 이름과 동일한 이름의 함수가 생성자가 되는데 PHP5에서는 특별히 __construct()라는 이름의 함수를 따로 만들어 두었습니다. 생성자는 반드시 필요한 것은 아니므로 생략해도 상관없습니다. 인간에 대한 생성자를 정의해보면 다음과 같습니다.

```
function __construct($hname)
{
  $Name = $hname;
  $Age = 1;
}
```

아기가 태어나면 이름을 짓고 나이를 한 살 먹게 합니다. 몸무게나 키도 생성자의 파라미터를 통해서 전달하여 초기화할 수 있습니다. PHP5에서는 생성자의 함수 오버로딩(Function Overloading)을 지원하지 않기 때문에 여러 개의 생성자를 사용할 수가 없습니다. 그렇기 때문에 인자의 초기값 설정을 이용하여 여러 개의 생성자가 존재하는 것과 같은 효과를 얻을 수 있습니다.

```
function __construct($pName, $pAge=1, $pHeight=50, $pWeight=3.5)
{
  $Name = $pname;
  $Age = $pAge;
  $Height = $pHeight;
  $Weight = $pWeight;
}
```

소멸자

생성자와 마찬가지로 소멸자(destructor)도 존재합니다. 객체가 소멸하면서 해야 하는 일들을 정의하는 함수입니다. __destruct()라는 이름을 가지며 인간의 경우에 죽으면서 유언을 남기는 것과 같은 일을 소멸자에 정의할 수 있습니다. 소멸자는 생성자와 달리 인자를 가지지 못합니다.

```
function destruct()
{
  Talk( "창문을 닫아 주오!");
}
```

유명한 방랑시인 김삿갓의 유언을 한번 인용해 보았습니다. 돌아가실 때 찬바람이 불었나 봅니다. 이렇게 소멸자를 만들어두면 모든 인간이 죽으면서 하나같이 이 말을 하게 될 것입니다. 따라서 유언이라는 속성을 만들어서 사람마다 다른 유언을 하도록 하는 것을 생각해 볼 수 있습니다. 생성자와 소멸자를 추가한 인간 클래스는 다음과 같습니다.

human.php

```
<?

  class Human { //인간 클래스를 정의합니다.

    var $Name;
    var $Age;
    var $Height;
    var $Weight;
    var $DyingWish;

    function __construct($pName, $pAge = 1, $pHeight=50,
    $pWeight=3.5)
    {
      $Name = $pname;
      $Age = $pAge;
      $Height = $pHeight;
      $Weight = $pWeight;
    }

    function destruct()
    {
      Talk( $DyingWish );
    }

    function Eat ( $foods ) {
      //먹는 행위를 함수로 정의
    }
    function Walk ( $destination ) {
      //걷는 행위를 함수로 정의
    }
    function Work ( $job ) {
      //일하는 행위를 함수로 정의
    }
    function Talk ( $words ) {
      //말하는 행위를 함수로 정의
    }
  }

?>
```

인스턴스 생성하기

인간 클래스를 정의하였으니 이제 이 클래스를 이용하여 철수, 영희와 같은 객체를 생성해 보겠습니다. 클래스를 통해 객체를 만들어내는 일을 인스턴스(Instance)를 생성한다고 말합니다. 객체는 new 키워드를 사용하여 생성할 수 있습니다.

```
<?
  /* Human 클래스를 여기에 삽입 */

  $charles = new $Human('철수');
  $younghee = new $Human('영희', 1, 50, 3.5);
  $noname = new $Human();
?>
```

위의 예처럼 new 키워드를 이용하여 객체를 판화 찍어내듯이 계속해서 찍어낼 수 있습니다. 철수와 영희는 인자의 수가 다릅니다. 철수는 이름만 전달되기 때문에 몸무게와 키 정보는 초기값으로 설정됩니다.

객체의 속성과 기능 사용하기

이제 객체를 생성하였으니 생성한 객체를 이용하는 방법에 대해 알아봅시다. 객체의 속성과 기능에 접근하려면 '->' 기호를 사용합니다.

```
<?
  /* Human 클래스를 여기에 삽입 */

  //5살짜리 철수를 생성합니다.
  $charles = new $Human('철수', 5);

  //철수는 몇 살?
  $charles->Talk($charles->$Age);

  //철수야! 밥 먹자~
  $charles->Eat("dinner");

  //밥을 먹고 난 후 철수의 키와 몸무게가 늘었다.
  $charles->$Height = 110; // 110 cm
  $charles->$Weight = 22; // 22 Kg

?>
```

위와 같이 '->' 기호를 이용하여 객체의 변수와 함수들에 접근할 수 있습니다. 철수는 몇 살인지 대답을 하고 밥을 먹습니다. 밥을 먹고 난 후 키와 몸무게가 증가하여 110cm와 22kg이 되었습니다.

클래스 상속하기

객체지향 프로그래밍의 장점 중 하나인 상속에 대해 배워보겠습니다. 절차식 프로그래밍에서는 새로운 모듈을 개발하기 위해서 보통 비슷한 모듈의 소스 코드를 복사해다가 수정하여 원하는 기능을 구현하는 방법을 사용하였습니다. 그러나 객체지향 프로그래밍에서는 클래스를 수정하는 것이 아니라 기존 클래스의 속성과 기능을 상속받아 새로운 클래스를 정의할 수 있게 해줍니다.

아기는 인간의 한 형태이기 때문에 인간의 속성과 기능을 가집니다. 만약 아기 클래스를 새롭게 정의하고자 한다면 인간의 모든 속성과 기능을 다시 정의해주고 아기만의 속성과 기능을 추가해야 할 것입니다. 하지만 상속을 이용하면 인간의 모든 속성과 기능을 새로 정의해줄 필요가 없습니다. 즉, 아기 클래스는 인간 클래스의 속성과 기능을 모두 물려받을(상속) 수 있습니다.

```
<?
  /* Human 클래스를 여기에 삽입 */

  class Baby Extends Human {   //인간 클래스를 상속받아 아기 클래스를 정의합니다.

        function 모유먹기 () {
      //모유를 먹는 행위를 함수로 정의
        }
        function 천사와대화 () {
      //천사와 대화하는 행위를 함수로 정의
        }
  }

      //아기 클래스를 이용해 재민이 객체를 생성
  $재민 = new Baby('재민');
  $재민->천사와대화(); // 옹알~ 옹알~
?>
```

PHP에서 상속을 받으려면 Extends라는 키워드를 사용합니다. Extends 뒤에 오는 클래스의 모든 기능과 속성을 그대로 상속받아 사용할 수 있기 때문에 생성자와 소멸자를 비롯하여 객체의 기능과 속성을 새롭게 정의하지 않아도 모두 사용할 수 있게 됩니다.

C++와 자바와 같은 객체지향 프로그래밍 언어에서는 다중 상속을 지원합니다. 다중 상속이란 하나 이상의 클래스에 대해 상속을 받을 수 있는 것을 말합니다. 그러나 PHP에서는 다중 상속을 지원하지 않기 때문에 하나의 클래스만을 상속받을 수 있습니다.

객체지향 프로그래밍이 필수인가?

최근의 프로그래밍 트렌드는 객체지향 프로그래밍입니다. 그래서 PHP에서도 서둘러 객체지향 프

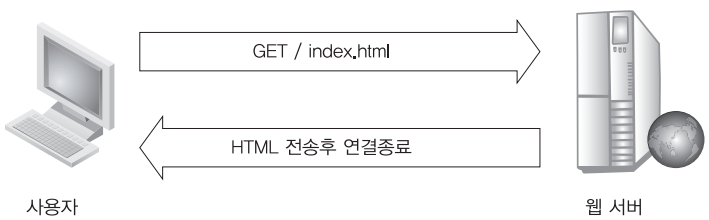
로그래밍 기법을 도입했습니다. 그러나 객체지향 프로그래밍은 대규모 프로젝트를 위해서 태어났다고 해도 과언이 아닙니다. PHP는 대부분의 프로그램이 짧게 금방 작성할 수 있을 만큼 쉽고 PHP 스크립트 자체가 한 줄 한 줄 처리되는 방식이기 때문에 사실 어떻게 보면 PHP와 객체지향 프로그래밍은 어울리지 않는다고 볼 수 있습니다. 무조건적인 유행을 따라가는 것이 선두를 향해가는 것이라고 생각하는 것은 오산입니다. 필자는 개인적으로 웹이라는 환경의 제약 때문에 객체지향 프로그래밍의 장점이 빛을 발하기 쉽지 않다고 생각합니다. 그래서 많은 PHP 프로그래머가 아직 객체지향 프로그래밍이 아닌 절차식 프로그래밍 방식을 사용하고 있습니다. 하지만 점점 웹 프로그램도 규모가 방대해지고 있어서 여러 명의 개발자가 참여하는 프로젝트가 많아지고 있습니다. 이에 따라 객체지향 프로그래밍 기법도 더 많이 활용되지 않을까 생각합니다.

Chapter

06

쿠키와 세션

HTTP 프로토콜은 사용자의 웹 서비스에 대한 요구를 매번 새롭게 받아들입니다. 이 말은 사용자가 사이트에 접속 후 다른 페이지에 접근하면 사이트에서는 이 두 접속을 한 사용자의 접속으로 여기는 것이 아니라 완전 별개의 접속으로 생각한다는 것을 의미합니다. 그 이유는 다음과 같이 HTTP 프로토콜에서 사용자의 컴퓨터가 서버에 접속하여 페이지를 요청하고 나서 서버에서 해당 파일의 정보를 전송하고 나면 연결을 끊어버리기 때문입니다.



[그림 6-1] HTTP 프로토콜의 방식

사용자의 요청에 대해 HTML 문서를 전송하고 나면 웹 서버는 연결을 종료합니다. 일반적으로 데이터 통신에서는 한번 연결되면 1대 1로 연결된 실을 통해서 지속적으로 대화하는 종이컵 전화기처럼 1대 1로 연결된 가상의 회선을 통해 계속해서 통신을 합니다. 그러나 웹 서비스에서는 사용자가 웹 서버에 데이터를 요청하여 수신받은 후 수신받은 정보를 읽어 보는데 시간이 걸리게 됩니다. 따라서 지속적으로 통신을 연결해두어도 실제로 통신을 하는 시간이 얼마 되지 않는 경우가 많습니다. 만약 100명의 사용자가 웹 서버에 이런 방식으로 회선을 연결해두고 있다면 웹 서버는 이 많은 접속을 관리하기가 매우 힘들어지게 됩니다. 그래서 보다 효율적인 서비스를 위해서 필요한 정보를 전달하고 나면 바로 접속을 끊어서 다음 요청을 기다리는 방식을 사용하게 되었습니다.

그렇지만 이러한 방식의 단점은 지속적으로 서버에 정보를 요청하는 사용자의 경우 이전 접속과 현재의 접속 간의 연관성을 알지 못한다는 것입니다. 예를 들면, 사용자의 인증을 하고 난 후에 사용자가 다른 페이지에 접근하면 웹 서버는 이미 인증된 사용자의 접속이란 것을 인지할 수 없다는 의미입니다.

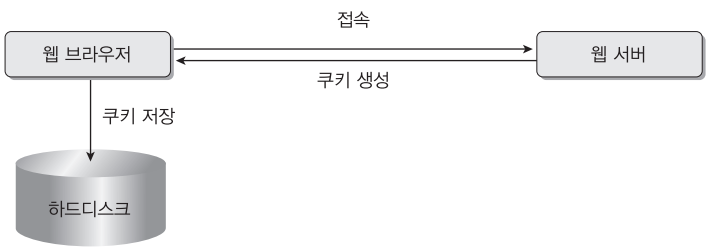
바로 이러한 HTTP 프로토콜의 단점을 이를 보완하고자 쿠키와 세션이 탄생하게 되었습니다.

Section 01

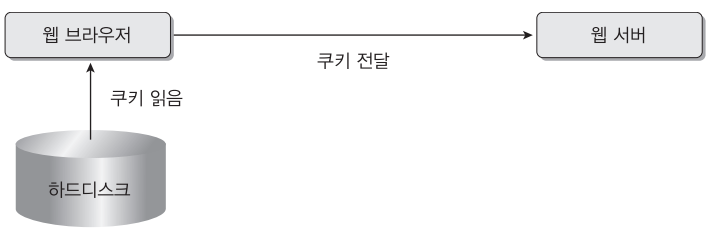
쿠키

우리가 들어가려고 하는 사이트를 놀이동산이라고 하면 쿠키는 자유이용권입니다. 처음에 놀이동산에 들어갈 때 자유이용권을 사면(쿠키는 무료이지만) 어느 놀이기구를 타더라도 그것만 보여주면 다 탈 수 있습니다. 그렇듯이 사이트에서 사용자에게 쿠키를 만들어주면 사용자는 쿠키를 보여주며 사이트를 마음껏 구경할 수 있는 것입니다.

그렇다면 쿠키에 대해 좀 더 자세히 알아봅시다.



[그림 6-2] 쿠키의 저장



[그림 6-3] 쿠키의 전달

쿠키는 웹 서버에서 웹 브라우저를 통해 접속자의 하드디스크에 저장하는 정보입니다. 웹 서버에서 쿠키를 생성할 때 쿠키 이름과 값을 웹 브라우저에 전달해 줍니다. 이때 웹 브라우저는 쿠키를 건네준 사이트의 이름으로 쿠키를 하드디스크에 저장하게 됩니다. 쿠키가 접속자의 하드디스크에 저장되기 때문에 웹 서버가 쿠키를 원한다고 해서 마음껏 쿠키정보를 가져올 수는 없습니다. 단지 웹 브라우저가 건네주는 쿠키만을 받을 수 있습니다. 그래서 웹 브라우저가 웹 서버에 접속할 때 해당 사이트 이름으로 된 쿠키가 있는지 먼저 하드디스크에서 검색합니다. 만약 있다면 쿠키를 웹 서버에 전달합니다. 이러한 방식 때문에 웹 브라우저가 쿠키를 지원하지 않거나 사용자가 쿠키의 생성을 제한해 놓았다면 쿠키를 사용할 수 없습니다.

PHP에서 쿠키 다루기

이제 PHP에서 쿠키를 어떻게 사용하는지를 알아봅시다. PHP에서는 쿠키를 생성하기 위하여 다음과 같은 함수를 제공하고 있습니다.

```
setcookie (쿠키이름, 쿠키값, 만료시간, 경로, 도메인, 보안);
```

setcookie() 함수는 다음과 같이 6개의 인자를 가집니다.

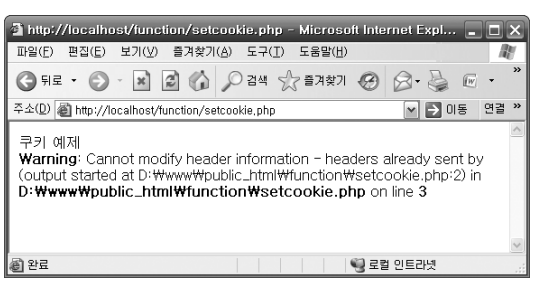
인자	필수	설명
쿠키 이름	예	쿠키의 이름
쿠키 값	아니오	쿠키의 값
만료 시간	아니오	언제 쿠키가 만료될지 시간을 설정
경로	아니오	이 쿠키를 사용할 수 있는 서버 경로
도메인	아니오	이 쿠키를 사용할 수 있는 도메인
보안	아니오	HTTPS 에서만 사용할지 여부를 설정

[표 6-1] setcookie 함수의 인자

또한 setcookie() 함수는 쿠키를 성공적으로 생성한 경우에는 TRUE를, 실패한 경우 FALSE를 되돌려 줍니다. 예제를 통해 함수의 사용법을 알아봅시다.

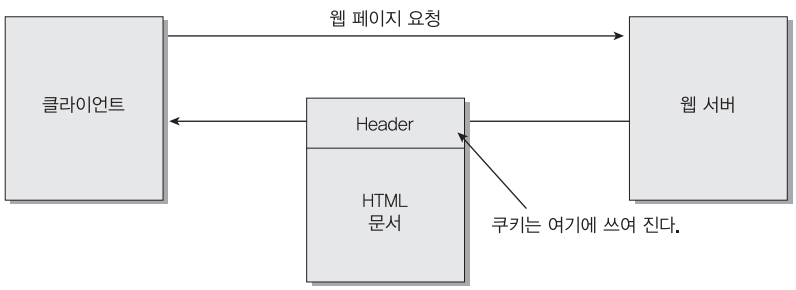
[예제 6-1] 쿠키의 생성

```
1  <?
2     echo '쿠키 예제';
3     setcookie('php');
4  ?>
```



[그림 6-4] 예제 6-1의 실행결과

위의 예제는 문법상 틀린 부분이 없음에도 경고 메시지가 출력되었습니다. 이 경고는 헤더 정보를 수정할 수 없어서 발생한 것입니다. HTTP 요청(request)과 응답(response)은 헤더(header)와 바디(body)로 이루어져 있습니다. 먼저 헤더 정보가 쓰이고 그다음에 바디가 쓰입니다. 쿠키는 이들 중 헤더에 쓰여지고 HTML 문서가 바디를 통해 전달됩니다. 그래서 쿠키를 생성하려면 헤더에 쓰여야 합니다. 다시 말해 setcookie() 함수가 앞에 있어야 합니다.



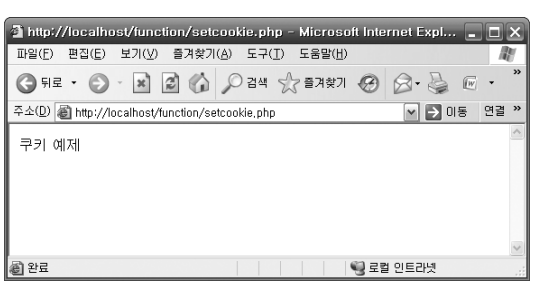
[그림 6-5] 쿠키의 전달방식

[예제 6-1]의 헤더 정보를 살펴보면 헤더 어디에도 쿠키 정보가 없음을 알 수 있습니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 00:52:23 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Content-Length: 231
Content-Type: text/html
```

[예제 6-1]의 소스 코드를 올바르게 수정하면 다음과 같습니다.

```
<?
  setcookie('php');
  echo '쿠키 예제';
?>
```



[그림 6-6] 예제 6-1의 수정된 코드 실행결과

단순히 출력의 순서를 바꾸는 것만으로 경고가 말끔히 사라졌습니다. 제대로 쿠키가 헤더에 쓰였는지 확인을 해보면 다음과 같습니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 01:03:08 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: php=
Content-Length: 9
Content-Type: text/html
```

헤더를 보면 Set-Cookie 항목에 php라는 이름의 쿠키가 등록되어 있음을 알 수 있습니다. "php="로 표시되는 이유는 우리가 php라는 이름의 쿠키를 생성하였으나 값을 부여하지 않았기 때문입니다. 쿠키 정보가 헤더에 제대로 쓰였다면 내 컴퓨터에 쿠키 파일이 저장되었단 소리이니 한번 찾아보도록 하겠습니다.

Windows XP에서는 C:\Documents and Settings\아이디\Cookies 디렉토리에 쿠키가 저장됩니다.

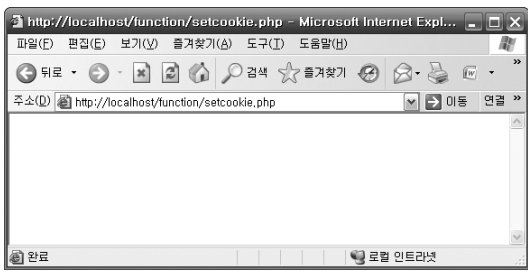
[그림 6-7] 쿠키 저장 폴더

그러나 저장되어야 할 쿠키 파일이 보이질 않습니다. 무언가 잘못된 것일까요? 아닙니다. 생성이 안 되어야 정상입니다. 앞에서 쿠키는 사용자의 하드디스크에 저장된다고 말해놓고 무언가 앞뒤가 안 맞는 것 같습니다. 그러나 이 경우에는 쿠키가 단지 헤더에서만 존재하기 때문입니다. 쿠키가 파일로 만들어지는 경우는 잠시 후에 다루도록 하겠습니다.

setcookie() 함수가 에러 없이 잘 동작하고 있으니 이제 쿠키에 값을 설정하고 설정된 값을 읽어보겠습니다.

[예제 6-2] 쿠키 값 읽기

```
1  <?
2    setcookie('php' , 'Cool~');
3    echo $_COOKIE['php']; //이름이 php인 쿠키의 쿠키 값을 불러와서 출력
4  ?>
```

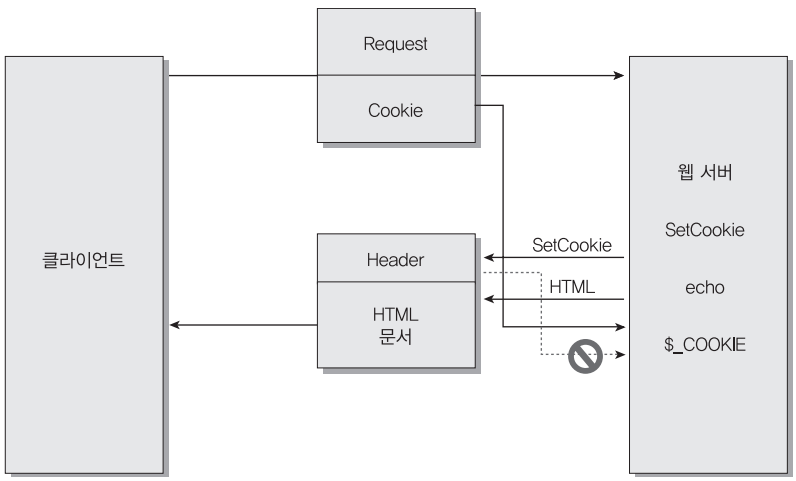


[그림 6-8] 예제 6-2의 실행결과

그런데 쿠키 값을 출력했는데 값이 출력되지 않았습니다. 쿠키가 제대로 쓰이지 않은 걸까요?

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 01:16:23 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: php=Cool%7E
Content-Length: 0
Content-Type: text/html
```

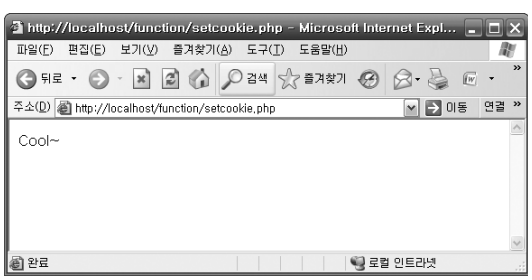
헤더 정보에는 분명 php라는 이름에 Cool%7E(%7E는 틸드(~)가 인코딩된 값입니다)라는 값으로 쿠키가 제대로 쓰여 있습니다. 쿠키에 정보가 제대로 쓰여 있음에도 그 값이 출력되지 않는 이유는 생성 직후에는 쿠키 값이 보이질 않기 때문입니다.



[그림 6-9] 쿠키 전달 및 저장

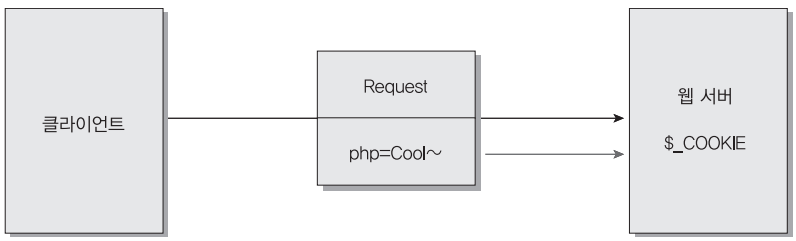
위 그림에서 보다시피 우리가 setcookie() 함수를 통해서 작성한 쿠키는 헤더에 쓰입니다. 그러나 읽으려고 시도하는 $_COOKIE['php'] 값은 우리가 쓴 헤더에서 읽어들이는 것이 아니라 클라이언트가 웹 서버로 페이지를 요청하면서 보낸 쿠키 정보를 읽어들이는 것입니다. 그래서 새로 만들어진 쿠키가 바로 적용되지 않는 것입니다. 만약 다른 페이지로 이동하거나 새로 고침을 해보면 이미 클라이언트의 웹 브라우저에 헤더 정보가 쓰여 있으므로 제대로 된 쿠키 값을 볼 수 있게 됩니다.

그럼 새로 고침을 해봅시다.



[그림 6-10] 새로 고침을 통한 쿠키 확인

앞서 언급한 대로 제대로 출력되는 것을 확인할 수 있습니다. 이는 다음 그림과 같이 클라이언트에서 페이지를 요청하면서 쿠키 정보를 전달해 주었기 때문입니다.



[그림 6-11] 클라이언트 쿠키의 전달

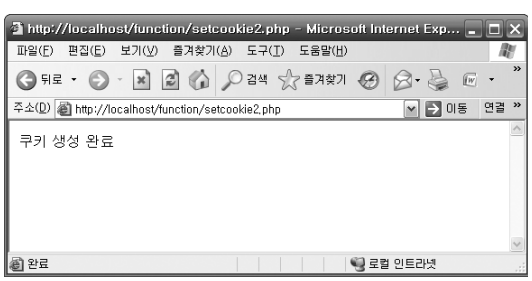
생성된 쿠키 값을 읽고자 할 때는 $_COOKIE['쿠키이름']과 같은 방법으로 가능합니다. PHP 설정(php.ini 파일)에서 register_globals가 on이면 $쿠키이름과 같은 방법으로도 접근할 수 있지만 되도록이면 위와 같은 방법을 사용하는 걸 권장합니다. 단, PHP 버전 4.1 이전에서는 $HTTP_COOKIE_VARS['쿠키이름']과 같은 방법을 사용해야 합니다.

쿠키 파일이 저장되는 디렉토리를 확인해보면 이번에도 쿠키 파일이 생성되지 않은 것을 알 수 있습니다.

이번에는 쿠키의 만료 시간을 설정해 보겠습니다.

[예제 6-3] 만료 시간 설정

```
1  <?
2     $result = setcookie('php' , 'Cool~' , time()+1000);
3
4     if ($result)
5     echo '쿠키 생성 완료';
6     else
7     echo '쿠키 생성 실패';
8  ?>
```



[그림 6-12] 예제 6-3의 실행결과

setcookie() 함수의 세 번째 인자는 만료 시간입니다. 만료 시간은 쿠키가 언제까지 남아있어야 하는지를 알려주는 것으로 이 인자 값이 있으면 주어지 시간 동안 쿠키가 살아있게 되고 인자 값을 주

지 않으면 기본값으로 브라우저가 닫힐 때까지 쿠키가 살아있게 됩니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 02:16:56 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: php=Cool%7E; expires=Tue, 30-Oct-2007 02:33:36 GMT
Content-Length: 14
Content-Type: text/html
```

헤더 정보를 살펴보면 expires 항목이 추가된 것을 알 수 있습니다. 이렇게 만료 시간을 지정하여 쿠키를 언제 제거할지를 판단하게 합니다.

예제에서 주어진 만료 시간을 살펴봅시다.

```
time()+1000
```

이는 앞서 배운 time() 함수를 이용하여 현재시간 + 1000초를 의미합니다. 즉, 지금부터 1000초까지만 쿠키가 살아있게 됩니다. 하루 동안만 살아있게 하고 싶다면,

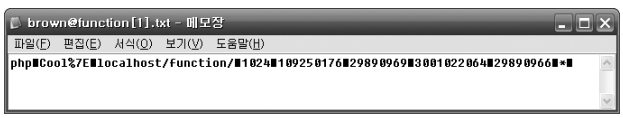
```
time() + 60*60*24
```

60초 곱하기 60은 1시간, 1시간 곱하기 24는 24시간으로 하루를 나타냅니다.

setcookie() 함수에서 만료 시간을 지정하는 경우 브라우저를 닫더라도 쿠키가 계속해서 살아 있게 됩니다. 이는 만료 시간을 지정한 경우 계속해서 쿠키를 사용할 수 있도록 파일로 저장해두기 때문입니다. 앞의 두 예제에서 쿠키를 생성하였음에도 쿠키 파일이 생성되지 않았던 이유가 바로 이것입니다. 쿠키의 만료 시간을 지정하지 않으면 쿠키 값을 파일로 저장하지 않고 웹 브라우저의 헤더에만 저장하였다가 브라우저가 닫히면서 쿠키가 사라지게 되는 것입니다. [예제 6-3]의 경우 만료 시간이 지정되었으므로 해당 디렉토리에 쿠키 파일이 반드시 존재하게 됩니다.

[그림 6-13] 하드디스크에 저장된 쿠키 파일

생각한 것처럼 쿠키 파일이 생성된 것을 확인할 수 있습니다. 쿠키 파일을 열어서 쿠키에 어떠한 정보가 저장되는지 엿보도록 합시다.



[그림 6-14] 쿠키 파일에 기록된 정보

쿠키 파일에 낯익은 값들이 보이는군요. 쿠키이름, 쿠키 값, 경로 등이 쿠키 파일에 저장됩니다.

일반적으로 쿠키를 생성하고자 할 때 만료 시간까지 3개의 인자를 사용합니다. 만약 현재 웹 페이지가 존재하는 디렉토리를 벗어나서 다른 디렉토리에 있는 웹 페이지가 이 쿠키 값을 사용하려면 네 번째 경로 인자를 사용해야 합니다. 왜냐하면 경로가 지정되지 않으면 현재 파일이 존재하는 디렉토리 내에서만 쿠키를 사용하도록 정해져 있기 때문입니다.

```
<?
    echo $_COOKIE['php'];
?>
```

위와 같은 코드를 다른 디렉토리에 저장하여 실행해보면 다음과 같이 쿠키가 출력되지 않음을 알 수 있습니다.

[그림 6-15] 쿠기가 없는 디렉토리

만약 다음과 같이 경로를 지정해주면 지정된 경로 이하의 모든 디렉토리에서 쿠키에 접근할 수 있게 됩니다.

[예제 6-4] 경로 설정

```
1  <?
2    $result = setcookie('php' , 'Cool~' , time()+1000, '/');
3
4    if ($result)
5    echo '쿠키 생성 완료';
```

```
6     else
7     echo '쿠키 생성 실패';
8   ?>
```

[그림 6-16] 예제 6-4의 실행결과

위와 같이 다른 디렉토리에서 접근한 때도 쿠키 값이 제대로 출력되는 것을 확인할 수 있습니다. 이는 쿠키를 설정할 때 경로 인자에 최상위 디렉토리를 지정했기 때문입니다. 그래서 해당 사이트의 모든 디렉토리에서 쿠키를 읽는 것이 가능해 집니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 02:21:56 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: php=Cool%7E; expires=Tue, 30-Oct-2007 02:38:36 GMT;
path=/
Content-Length: 14
Content-Type: text/html
```

헤더 정보에서도 path 항목이 추가된 것을 확인할 수 있습니다.

만약 docs.ezphp.net과 www.ezphp.net처럼 서브 도메인이 존재하는 경우 서브 도메인 간에는 쿠키가 기본적으로 공유되지 않습니다. 그래서 서브 도메인 간에 쿠키 정보를 공유하고 싶다면 다섯 번째 인자인 도메인 인자를 추가하면 됩니다.

```
setcookie('php','Cool~',time()+60,'/','.ezphp.net');
```

여기서 주의할 점은 도메인 앞에 점(.)이 있다는 것입니다. 이 점이 없으면 제대로 동작하지 않게 되니 주의해야 합니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 02:33:01 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
```

```
Set-Cookie: php=Cool%7E; expires=Tue, 30-Oct-2007 02:49:41 GMT;
path=/; domain=.ezphp.net
Content-Length: 14
Content-Type: text/html
```

마지막으로 HTTP 프로토콜이 아니라 보안이 강화된 HTTPS 프로토콜을 이용하는 웹 페이지에서만 쿠키를 사용할 수 있게 하려면 보안 인자를 1로 설정하면 됩니다.

```
setcookie('php','Cool~',time()+60,'/','.ezphp.net',1);
```

헤더에는 다음과 같이 표시됩니다.

```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 02:34:07 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: php=Cool%7E; expires=Tue, 30-Oct-2007 02:50:47 GMT;
path=/; domain=.ezphp.net; secure
Content-Length: 14
Content-Type: text/html
```

여기서 잠깐

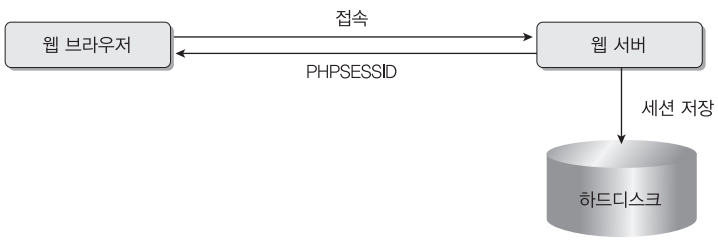
쿠키는 사용자의 하드디스크에 저장되기 때문에 사용자에 의해서 수정될 수 있습니다. 따라서 쿠키에 중요한 정보를 담아두면 사용자가 이를 악용할 우려가 있습니다. 중요한 정보는 쿠키를 이용해 저장하지 마십시오. 자세한 것은 웹 해킹 단원에서 다루도록 합니다.

Section 02

세션

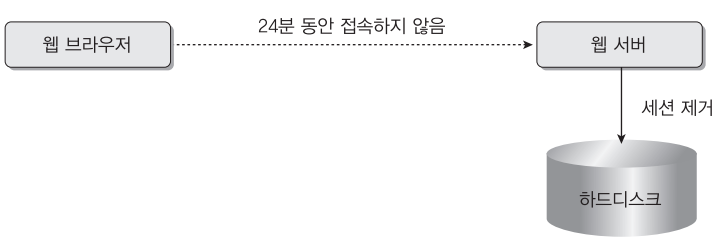
HTTP 프로토콜은 사용자의 접속을 매번 새롭게 인식하여 어떤 요청들이 하나의 사용자로부터 발생한 일련의 요청임을 인지하지 못하는 단점이 있다고 했습니다. 이를 사용자의 상태를 유지할 수 없다고 하여 stateless 프로토콜이라고 합니다. 이러한 HTTP 프로토콜의 단점을 보완하고자 세션이 등장했습니다.

세션은 쿠키와 유사하지만 큰 차이가 있습니다. 쿠키가 접속자의 컴퓨터에 저장되는 반면 세션은 웹 서버에 저장됩니다. 쿠키는 사용자의 컴퓨터에 저장되기 때문에 사용자가 홈페이지에 들어오는 것은 알 수 있어도 로그아웃을 하지 않고 나가 버리면 웹 서버는 사용자가 접속을 끊었는지 아직 웹 페이지를 읽고있는 것인지 알 수가 없습니다. 그러나 세션은 서버 측에 존재하기 때문에 웹 서버가 주기적으로 세션의 상태를 확인할 수 있어서 특정 시간 동안 웹 사이트 내에서 어떠한 이동도 발생하지 않으면 사용자가 나간 것으로 간주하여 세션을 삭제할 수 있습니다.



[그림 6-17] 세션의 저장

사용하지 않는 세션을 주기적으로 제거하는 이유는 세션이 서버 측에 저장된다는 것에 있습니다. 서버 측에 저장되므로 세션을 무작정 쌓아둘 수만은 없습니다. 서버의 하드디스크 용량의 문제도 있겠지만 불필요한 세션이 많아지면 세션의 관리 또한 힘들어질 게 분명하기 때문입니다. 그러한 이유로 주기적으로 세션을 관리하여 필요 없다고 느껴지는 세션을 지우는 작업을 하게 됩니다. 기본적으로 24분을 기준으로 하여 세션을 삭제합니다.



[그림 6-18] 세션의 제거

쿠키와 비교하여 또 하나의 큰 차이는 보안적인 차이입니다. 쿠키는 사용자의 컴퓨터에 저장되므로 사용자가 쿠키를 생성하거나 웹 서버로부터 만들어진 쿠키 값을 변조하는 것이 충분히 가능합니다. 반면에 세션은 서버 측에 저장되므로 세션 값을 사용자가 변조할 수 없습니다. 놀이공원의 예를 들어보겠습니다.

아빠랜드는 자유이용권을 구매한 입장객에게 종이로 만들어진 팔찌를 손목에 채워주었습니다. 이 팔찌를 보고 자유이용권 구매자임을 판단해 놀이기구를 마음껏 사용할 수 있게 하였습니다. 그런데 어느 순간 복제된 가짜 팔찌가 암표상에게서 팔리고 있음을 알게 되었습니다. 그래서 아빠랜드 관

계자들은 회의 끝에 전자팔찌 시스템을 도입하기로 하였습니다. 전자팔찌 시스템은 전자팔찌에 무작위의 숫자를 저장하고 자유이용권 구매자는 시스템에 그 번호를 기록합니다. 그리고는 놀이기구를 사용할 때 카드리더기에 팔찌를 대는 방법으로 자유이용권 구매자임을 확인하는 시스템입니다. 이 시스템에서는 전자팔찌에 기록된 번호가 매우 중요하기 때문에 입장객들이 놀이공원을 나갈 때 모두 회수해서 시스템에서 번호를 삭제하도록 하였습니다. 그런데 팔찌를 반납하지 않고 나가버리는 사람들이 생겨서 부득이하게 "24분 제한 규칙"이란 것을 만들었습니다. 만약 24분 동안 놀이기구를 타거나 놀이공원 시설을 이용하지 않으면 놀이공원을 빠져나간 것으로 생각하고 팔찌의 번호를 시스템에서 지워버리는 규칙입니다. 이 전자팔찌 시스템을 도입하여 아빠랜드는 암표상들의 가짜 자유이용권 판매를 획기적으로 줄일 수 있게 되었습니다.

이 아빠랜드와 마찬가지로 세션은 무작위의 숫자로 이루어진 세션 ID를 사용자에게 발급하고 정보는 모두 서버 측에 기록합니다. 웹 서버는 사용자의 세션 ID를 확인하여 사용자의 상태를 유지할 수 있게 됩니다. 그러나 세션이 사용자의 상태를 확인하는 작업을 주기적으로 시행하고 웹 서버 측에 정보가 저장되기 때문에 저장 공간을 소모하는 등 웹 서버에 부담을 주게 됩니다.

세션 시작하기

PHP에서 세션을 사용하려면 세션을 시작해줘야 합니다. 세션을 시작하지 않은 페이지에서는 세션 값을 사용할 수 없기 때문입니다.

```
session_start();
```

위와 같이 session_start() 함수를 호출하는 방법으로 세션을 사용할 수 있게 해줍니다. 세션을 시작하면 사용자에게 세션 ID가 발급됩니다. 일반적으로 사용자의 웹 브라우저에 PHPSESSID라는 이름으로 쿠키가 기록됩니다.

[예제 6-5] 세션 시작하기

```
1  <?
2    session_start();
3  ?>
```

[예제 6-5]의 실행결과는 웹 브라우저에 나타나지 않습니다. 웹 서버로부터 응답받은 헤더 정보를 살펴보면 다음과 같습니다.

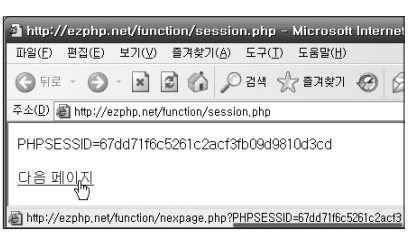
```
HTTP/1.1 200 OK
Date: Tue, 30 Oct 2007 08:02:10 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: PHPSESSID=cb64ea3ac26d9ad9ba31390ef8763835; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 12
Content-Type: text/html
```

[예제 6-5]에서 setcookie() 함수를 사용하지 않았음에도 불구하고 자동으로 PHPSESSID라는 이름의 쿠키가 기록된 것을 알 수 있습니다. 이는 php.ini 파일의 설정 중 session.use_cookies 항목이 기본적으로 활성화되어 있기 때문입니다. 만약 이 항목이 비활성화되어 있다면 쿠키는 자동으로 생성되지 않습니다. 이런 때에는 세션 ID를 URL을 통해서 전달해 주어야 합니다.

[예제 6-6] URL을 이용한 세션 ID 전달

```
1  <?
2    session_start();
3    echo SID . '<P>';
4  ?>
5  <a href="nexpage.php?<?=strip_tags(SID)?>">다음 페이지</a>
```

[예제 6-6]에서 SID라는 상수가 사용되고 있음을 알 수 있습니다. 상수이기 때문에 앞에 변수를 나타내는 '$' 표시가 없습니다. SID는 세션 ID를 가리키는 상수입니다. strip_tags() 함수를 사용하는 이유는 크로스 사이트 스크립팅(XSS)를 방지하기 위해서입니다. 크로스 사이트 스크립팅에 대해서는 웹 해킹 단원에서 자세히 다루도록 하겠습니다.



[그림 6-19] 링크에 GET 방식으로 PHPSESSID를 전달

위의 그림에서 보는 바와 같이 링크에 GET 방식으로 PHPSESSID를 전달하는 것을 볼 수 있습니

다. 이처럼 세션을 URL로 전달하는 때는 세션 ID를 발급받은 후 페이지를 이동하면서 매번 세션 ID를 전달해야 하는 불편함이 있습니다. 그러나 php.ini 파일에서 session.use_trans_sid 항목을 활성화시키는 방법으로 이런 지루한 작업을 피할 수 있습니다. 만약 PHP 4.1.2 이하 버전을 사용하고 있다면 --enable-trans-sid 옵션을 추가하여 컴파일해야 합니다.

그런데 여기서 주의할 점은 세션 ID는 최초 발급될 때만 노출된다는 것입니다. 만약 [예제 6-6]을 실행하고 다시 새로 고침을 하면 SID 상수를 이용하여 세션 ID를 확인할 수 없습니다.

[그림 6-20] 새로 고침을 했을 때 세션 ID를 확인할 수 없다.

여기서 사용하는 URL을 이용한 세션 ID 전달 방법은 권하고 싶지 않습니다. 크로스 사이트 스크립팅에 악용될 소지가 있으니 될 수 있으면 쿠키를 이용한 방법을 사용하길 권합니다.

앞에서 세션을 시작하기 위해서 session_start() 함수를 호출하는 방법을 언급했습니다. 이외에도 세션을 시작하는 방법이 하나 더 있습니다. php.ini 파일의 session.auto_start 항목을 활성화하면 웹 사이트를 방문할 때 자동으로 세션이 시작되게 할 수 있습니다.

세션 변수 등록하기

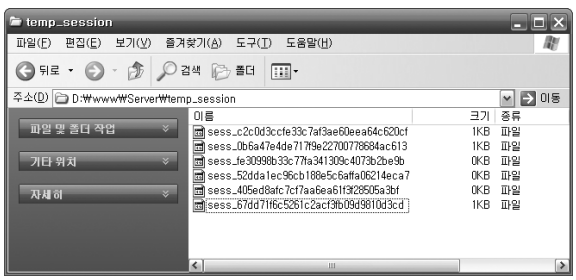
PHP 버전 4.1 이전에는 세션에 사용되는 변수를 별도로 등록해주는 절차가 필요했습니다. 그러나 최신 버전의 PHP에서는 간단하게 세션 변수를 사용할 수 있습니다.

[예제 6-7] 세션 변수 등록

```
1  <?
2    session_start();
3    $_SESSION['message'] = 'Welcome to PHP world!<BR>';
4
5  ?>
6  <a href="read_session.php">다음 페이지</a>
```

[예제 6-7]과 같이 수퍼글로벌 변수인 $_SESSION 배열을 사용하여 세션 변수를 등록하고 값을 저

장할 수 있습니다. [예제 6-7]을 실행하면 세션이 등록되었으니 서버에 파일로 세션 변수의 값이 저장될 것입니다.



[그림 6-21] 세션 정보가 기록되어 있는 세션 파일

세션이 저장된 디렉토리를 살펴보니 기존에 등록된 여러 가지 세션들이 아직 남아있는 것을 확인할 수 있습니다. 우리가 찾아야 할 세션은 [예제 6-6]에서 받은 세션 ID입니다. [예제 6-6]에서 "67dd71f6c5261c2acf3fb09d9810d3cd"라는 세션 ID를 발급받았습니다. 마침 제일 하단에 세션 파일이 존재하는군요. 파일을 열어보겠습니다.

[그림 6-22] 세션 파일의 내용

세션 파일에는 message라는 이름의 변수에 문자열(s)로 21자인 Welcome to PHP world! 라는 값이 저장되어 있다고 기록하고 있습니다. 여기서 우리는 다음과 같은 중요한 사실을 알 수 있습니다.

"**세션은 서버 측에 파일로 저장되기 때문에 만약 서버에 접근할 수 있거나 세션이 저장되는 디렉토리가 외부로 드러나 있으면 누구나 세션 정보를 훔쳐볼 수 있다.**"

따라서 세션을 저장하는 디렉토리는 반드시 외부에서 접근할 수 없는 디렉토리여야 합니다.

세션 변수 사용하기

세션 변수를 등록하고 세션 값을 설정하는 방법을 배웠습니다. 이제는 저장된 세션 변수를 어떻게 사용하는지에 대해 알아봅시다.

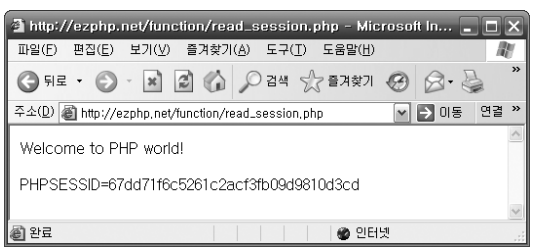
[예제 6-8] 세션 변수 사용

```
1  <?
2    session_start();
3
4    echo $_SESSION['message'];
5    echo "<P>";
6    echo "PHPSESSID=" .  session_id();
7  ?>
```

세션을 등록할 때와 마찬가지로 $_SESSION 배열을 이용하여 세션 변수를 사용할 수 있습니다. register_globals=on 상태라면 수퍼글로벌 변수를 사용하지 않고 직접적으로 $message 변수를 사용할 수 있습니다. 그러나 $message 변수를 사용하는 경우 일반적으로 출처(GET이나 POST 방식으로 전달된 변수인지)를 확인하지 않기 때문에 보안적으로 문제가 있을 수 있습니다.

session_id() 함수는 현재 사용자의 세션 ID를 알려주는 함수입니다. 세션 ID를 다루는 방법을 알려드리기 위해서 출력을 해두었으나 실제 사용 시에는 절대 세션 ID를 공개해서는 안 됩니다.

[예제 6-7]을 실행하고 다음 페이지 링크를 클릭하면 [예제 6-8]의 결과를 확인할 수 있습니다.



[그림 6-23] 예제 6-8의 실행결과

세션 변수 제거하기

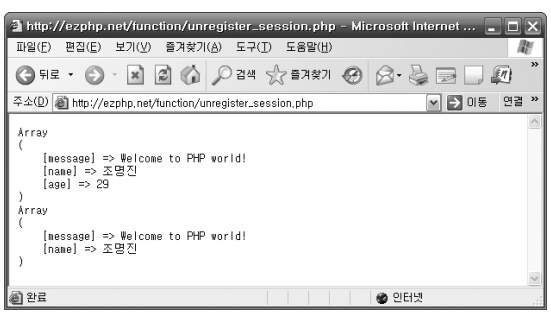
세션에 등록한 변수가 더는 필요하지 않다면 세션 변수를 제거해야 합니다.

[예제 6-9] 세션 변수 제거하기

```
1  <?
2    session_start();
3
4    //세션에 변수를 등록한다.
5    $_SESSION['name'] = '조명진';
```

```
6     $_SESSION['age'] = 29;
7
8     //보기 좋게 출력하기 위해서 <PRE> 태그를 사용한다.
9     echo '<PRE>';
10
11    //현재 세션에 저장된 변수를 출력한다.
12    print_r($_SESSION);
13
14    //세션 변수를 제거한다.
15    unset($_SESSION['age']);
16
17    //현재 세션에 저장된 변수를 출력한다.
18    print_r($_SESSION);
19
20    echo '</PRE>';
21 ?>
```

[예제 6-9]는 비밀스럽고 중요한 값인 필자의 이름과 나이를 세션에 기록하였다가 세션 값이 출력되자 나이가 노출되는 것을 꺼려 얼른 나이 항목을 제거하는 소스입니다. 오호통재라~ 내일모레면 서른이군요.



[그림 6-24] 예제 6-9의 실행결과

그런데 예상치 않은 변수(message 변수)가 출력되었습니다. 만약 위와 같이 출력된다면 분명히 열심히 공부하는 학생이고 위와 같이 출력되지 않고 이름과 나이만 출력된다면 게으른 학생임이 분명합니다. 왜냐하면 세션 ID가 아직 살아있어서 앞서 [예제 6-7]에서 등록한 message 변수가 출력되었으니 쉬지 않고 열심히 공부를 하고 있다는 증거가 아니고 뭐겠습니까? ^^

앞서 세션 변수의 등록과 사용 그리고 제거하는 방법을 곰곰이 생각해보면 이 방법은 모두 배열을 다루는 방법과 일치하는 것을 알 수 있습니다. 단지 세션 변수를 사용하기 위해서 세션을 시작해 주어야 한다는 것 이외에는 차이점을 찾을 수 없습니다. 따라서 특별히 세션이라고 별다른 방법이 있겠거니 생각하지 말고 배열과 똑같다고 생각하시면 쉽게 세션을 다룰 수 있을 것입니다.

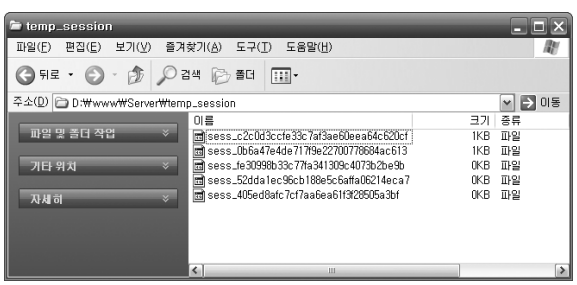
세션 종료하기

앞서 세션 시작하기가 있었으니 당연히 세션을 종료하는 것이 있게 마련입니다. 이제 마지막으로 세션을 종료하는 방법을 배워봅시다.

[예제 6-10] 세션 종료하기

```
1  <?
2     session_start();
3
4     //세션에 저장된 변수를 모두 제거한다.
5     session_unset();
6
7     //세션을 종료한다.
8     session_destroy();
9  ?>
```

세션에 저장된 값은 session_unset() 함수를 통하여 모두 제거할 수 있습니다. 또한 세션을 종료하려면 session_destroy() 함수를 호출합니다. 세션을 종료시키면 세션에 등록된 모든 변수가 제거되고 세션 ID도 제거됩니다.



[그림 6-25] 제거된 세션 파일

세션이 저장되는 디렉토리를 살펴보면 [그림 6-21]에서 봤던 "67dd71f6c5261c2acf3fb09d9810d3cd"라는 세션 ID를 갖는 세션 파일이 없는 것을 알 수 있습니다. 이는 session_destroy() 함수가 세션을 종료하면서 세션 파일을 제거하였기 때문입니다.

그런데 이상한 것이 세션을 종료하면 세션 안의 모든 변수가 제거되는데 왜 굳이 session_unset() 함수를 통해서 세션 변수를 제거하는 것일까요?

[예제 6-11] session_name을 사용한 경우 세션 종료하기

```
1  <?
2    session_start();
3    session_name('sess_name');
4
5    $_SESSION['name']='brown';
6
7    //세션을 종료한다.
8    session_destroy();
9
10   //세션 변수 출력
11   print_r($_SESSION);
12 ?>
```

session_name() 함수는 세션의 이름을 반환하거나 변경하는 데 사용하는 함수입니다. 위와 같이 session_name() 함수의 파라미터에 이름을 입력하면 세션의 이름이 지정된 이름으로 변경됩니다. 여기서 세션의 이름이 무엇을 말하는 것이냐 하면 앞서 우리가 자주 보았던 'PHPSESSID'를 말합니다. 즉, session_name() 함수를 이용하여 'PHPSESSID'를 다른 값으로 변경할 수 있다는 뜻입니다.

그런데 session_name() 함수를 사용하면 예제의 결과와 같이 session_destroy() 함수가 제대로 동작하지 않을 수 있습니다.

[그림 6-26] 예제 6-11의 실행결과

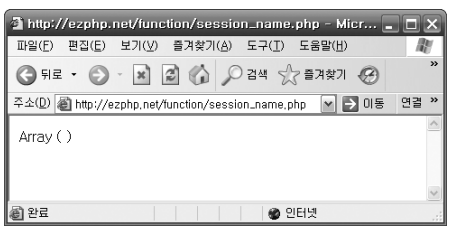
그래서 session_name() 함수를 사용했을 때는 반드시 세션 변수를 다음의 예제와 같이 모두 제거해 주어야 합니다.

[예제 6-12] session_name() 함수 사용 시 세션 제대로 종료하기

```
1  <?
2    session_start();
3    session_name('sess_name');
4
```

```
5     $_SESSION['name']='brown';
6
7     //세션 변수를 모두 제거한다.
8     session_unset();
9
10    //세션을 종료한다.
11    session_destroy();
12
13    //세션 변수 출력
14    print_r($_SESSION);
15 ?>
```

[예제 6-12]의 결과를 확인하면 제대로 세션이 종료되었음을 알 수 있습니다.



[그림 6-27] 예제 6-12의 실행결과

Section 03 쿠키냐? 세션이냐?

쿠키와 세션을 배우면서 유사한 점과 차이점을 알아보았습니다. 간략히 다시 정리를 해보면 쿠키와 세션은 HTTP 프로토콜의 사용자 상태를 유지할 수 없는 단점을 보완하고자 등장하게 되었고 쿠키는 사용자의 시스템에 저장되는 반면 세션은 웹 서버 측에 저장됩니다. 이러한 차이로 쿠키보다 세션이 보다 안전하며 중요한 정보를 저장하고자 할 때는 쿠키보다는 세션을 이용하는 것이 바람직합니다. 그러나 세션은 웹 서버에 무리를 주기 때문에 무조건 사용할 수 없습니다. 따라서 상황에 따라 쿠키나 세션 중에서 하나를 선택하는 것이 좋습니다.

--

--

Chapter

07

데이터베이스

데이터베이스는 PHP와 매우 긴밀하게 연결되어 있고 속칭 "절친"으로 불릴 만큼 가까운 관계입니다. 혼자만 있을 때보다 이 둘이 같이 있을 때가 더 빛이 나는, 서로 빛내주는 관계입니다. 그래서 PHP를 배울 때 데이터베이스를 빼놓고 이야기할 수 없습니다. PHP를 더 깊게 배우려면 데이터베이스를 알아야 합니다. 그러나 데이터베이스는 하나의 학문이기 때문에 이 조그만 책에 모든 것을 담을 수가 없습니다. 만약 데이터베이스에 대해 자세히 들여다보고자 한다면 오히려 이 책의 두께보다 더 두꺼운 분량을 할애해야 할지도 모릅니다. 그래서 데이터베이스의 가장 기본적인 개념과 반드시 알고 넘어가야 할 부분에 대해서만 간략하게 설명을 해보았습니다. 이 책에서 다루지 못하는 많은 데이터베이스 개념들과 새로운 기술들은 이 책을 독파한 후 꼭 시간을 내서 데이터베이스 전문 서적을 읽어보길 권해 드립니다. 당장은 필요성을 못 느낄지도 몰라도 나중에 게시판과 그보다 더 고수준의 프로그램을 작성할 때면 데이터베이스를 좀 더 자세히 알고 싶다는 생각을 자연스럽게 하게 될 것입니다. 처음에는 데이터베이스 개념과 이론들로 다소 어렵고 지루하게 여겨질 수 있습니다.

Section 01

파일 관리 시스템과 데이터베이스 관리 시스템

세상에는 셀 수 없이 많은 데이터(Data)가 있습니다. 매일 오르락내리락하는 환율이나 주가의 지수를 비롯하여 각 나라의 인구 수나 도시의 1년 예산, 한 학급 학생들의 키, 몸무게, 우리가 블로그나 싸이월드에 올린 사진 등이 모두 데이터입니다. 데이터는 단순한 사실이나 그 기록을 뜻하기 때문에 일반적으로 데이터 자체만으로 큰 의미가 있지는 않습니다. 그러나 수많은 데이터가 어떠한 의도에 의해서 가공되어 특별한 의미를 가지게 될 때, 이를 정보(Information)라고 합니다. 예를 들어 전국 17세 청소년 모두의 키 자체는 우리에게 어떤 의미가 있지 않지만 이를 바탕으로 17세 남자와 여자의 평균 신장을 계산한다면, 매해 계산된 평균 신장을 바탕으로 대한민국 청소년들의 성장 속도와 발육상태의 변화 등을 알 수 있는 좋은 정보가 될 수 있을 것입니다. 그뿐만 아니라 이렇게 생산된 정보는 다시 다른 정보를 재생산하는 데이터가 될 수 있습니다.

이처럼 의미 있는 정보를 생산해내려면 정보의 밑천이 되는 수많은 데이터를 수집할 필요가 있습니다. 구글(google)의 경우 수십억 개의 웹 페이지 데이터를 수집하여 저장하고 있는데 10억 개의 페이지를 보관하고 있다고 가정하면 대략 한 페이지가 300KB 정도라고 했을 때 약 280TB(테라바이트) 용량의 데이터를 가지고 있는 것입니다. 구글은 이러한 웹 페이지들을 수많은 사용자에게 제공하고 있으며 단 하루의 정보 처리량이 20PB(페타바이트, 1024TB)에 이른다고 합니다. 이렇게 많은 데이터를 만약 뒤죽박죽 보관해 두었다면 내가 원하는 웹 페이지를 찾아내기 위해서 아마도 평생의 시간을 들여도 모자랄지 모릅니다.

그래서 데이터를 체계적으로 관리하기 위하여 파일 관리 시스템이나 데이터베이스 관리 시스템(DBMS, Database Management System)을 사용하게 되었습니다. DBMS는 파일 관리 시스템의 발전된 형태이며, 논리적인 데이터베이스라는 구조를 사용하고 있기 때문에 데이터베이스를 관리하는 시스템이라는 이름을 갖게 되었습니다.

파일 관리 시스템

파일 관리 시스템(FMS, File Management System)은 범용적인 시스템이라고 하기보다는 개발자가 프로그램에 따라서 데이터의 형식을 정하고 이를 용도에 맞게 파일 형태로 저장하는 시스템 방식입니다. 예를 들어 컴퓨터학과 학생들의 신상정보를 관리하는 프로그램은 다음과 같은 형식으로 데이터를 저장할 수 있습니다.

```
1|강호동|27|178|서울시 강남구 도곡동
2|유재석|25|173|서울시 중구 신당동
```

3|박명수|23|182|서울시 금천구 시흥동
4|김제동|25|168|서울시 서대문구 아현동

데이터는 구분자(|)를 이용하여 번호, 이름, 나이, 신장 그리고 주소 순서로 기록하고 있으며 프로그램은 데이터가 저장된 형식을 알고 있기 때문에 파일을 읽어들여 형식에 따라 데이터를 구분하고 번호나 이름 항목을 통해서 해당 학생의 나이나 신장 그리고 주소의 정보를 검색할 수 있습니다. 반대로 서울에 거주하는 학생의 목록이나 나이가 25세 미만인 학생의 목록 등을 검색할 수도 있습니다.

이렇게 데이터를 관리하고 검색하기가 용이해진 이유는 몇 가지 규칙을 두고 그 형식에 맞춰서 데이터를 입력하도록 제약을 주고 있기 때문입니다. 이 데이터를 입력할 때는 반드시 이름, 나이, 신장, 주소를 각각 구분하여 입력해야 하며 입력된 값은 데이터 형식에 맞춰서 저장해야 합니다. 또한 각 항목을 구분하는 용도로 사용되는 구분자(|)를 사용자가 입력할 수 없도록 해야 합니다.

항목의 값을 구분하여 입력받지 않고 한꺼번에 입력받게 된다면 사람에 따라서 다양한 형태로 데이터가 저장될 것입니다. 만약 데이터를 입력할 때, "이름, 나이, 키, 주소를 입력하세요"라고 써두었다면 위의 데이터는 아래와 같이 기록되었을지도 모릅니다.

① 이름은 강호동이고 나이는 27살이며 키는 178 그리고 주소는 서울시 강남구 도곡동이다.
② 서울시 중구 신당동에 사는 25살 유재석입니다. 키는 173cm입니다.
③ 박명수 23살 키 182 주소 서울시 금천구 시흥동
④ 이름 : 김제동
나이 : 25
키 : 168
주소 : 서울시 서대문구 아현동

위와 같이 항목을 구분하지 않고 자유롭게 기록되었을 때 이름을 통해서 주소를 알고 싶다면 어떻게 해야 할까요? 어느 것이 이름이며 어느 것이 주소인지는 알 수 있을까요? 사람은 금세 어느 항목이 어떤 것인지 알 수 있겠지만 프로그램은 사람의 언어를 알지 못하기 때문에 이렇게 자유롭게 데이터가 저장되면 그 데이터를 관리하거나 검색하기가 힘들어집니다. 그래서 입력 시에 제약을 주어 입력된 값이 어느 항목에 대한 값인지를 알 수 있게 만들어야 합니다.

만약 데이터의 형식을 자유롭게 저장하였다면 어떻게 될까요?

① 1|강호동|27|178|서울시 강남구 도곡동
② 2+유재석+25+173+서울시 중구 신당동
③ 3|박명수|서울시 금천구 시흥동|23|182
④ 4#김제동#서울시 서대문구 아현동#25#168

위와 같이 구분자가 일관성이 없다거나 저장된 순서가 다르다면 프로그램은 값이 어느 항목에 해당하는지를 알 수 없습니다. 그래서 반드시 같은 구분자와 같은 순서로 기록되어야 합니다.

마지막으로 구분자를 "|"로 지정하였는데 사용자가 주소나 이름을 입력할 때 "|" 기호를 입력할 수 있다면 어떻게 될까요?

1|강|호|동|27|178|서울시 강남구 도곡동

위와 같이 기록된 경우 프로그램은 구분자를 기준으로 번호, 이름, 나이, 키, 주소라고 인식하게 되는데 이름에 구분자와 같은 기호를 입력하게 되면 다음과 같이 프로그램은 인식하게 됩니다.

번호	이름	나이	키	주소	X	X
1	강	호	동	27	178	서울시 강남구 도곡동

[표 7-1] 구분자에 따른 입력 형태

구분자가 나타나면 다음 항목이라고 인식하기 때문에 위와 같이 오류가 발생합니다. 따라서 약간의 제약을 두고 그 형식에 맞는지 확인하여 데이터를 일관성 있게 기록한다면 파일 관리 시스템으로 데이터를 효율적이고 체계적으로 관리할 수 있습니다.

그러나 파일 관리 시스템은 몇 가지 단점이 있습니다.

데이터의 종속성 (Data Dependency)

데이터의 종속성은 프로그램의 구조가 데이터의 구조에 영향을 받는 것을 의미합니다. 즉, 데이터의 구조가 프로그램의 데이터 저장방식을 결정하고 반대로 프로그램의 데이터 저장방식에 따라 데이터의 구조가 변경되는 것을 말합니다. 데이터의 종속성 때문에 데이터 구조가 변경되면 프로그램까지 같이 바꾸는 비용이 들기 때문에 프로그램의 개발과 유지보수가 어려워집니다.

데이터의 무결성 (Data Integrity) 침해

위에서 구분자를 사용자가 입력하게 되는 경우 나이 항목임에도 불구하고 문자가 입력되었습니다. 이렇게 데이터의 내용이 본래의 의도와 다른 형식을 갖게 될 때 데이터 무결성을 침해하였다고 말합니다. 데이터 무결성이 깨지면 잘못된 정보(거짓 정보)를 생산해 내기 때문에 2차적인 문제가 발생할 수 있습니다.

데이터의 중복성(Data Redundancy)

파일 시스템은 프로그램마다 데이터 종속성 등으로 인해서 공유가 안 되는 경우가 많아서 프로그램

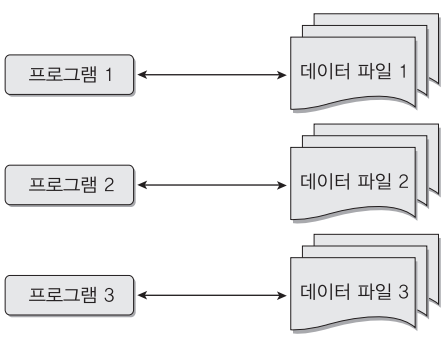
마다 같은 정보를 중복해서 저장하는 경우가 많습니다. 이는 저장 공간의 낭비이기도 하지만 데이터를 관리하는 측면에서 같은 정보를 여러 곳에서 보관하면 수정 시에 모든 데이터를 수정해야 하는 문제가 발생합니다.

데이터 불일치(Data Inconsistency)

데이터 중복성에서 언급하였듯이 여러 곳에 같은 정보가 저장되기 때문에 모두 수정이 되지 않는 경우, 예를 들어 학생의 주소를 한 곳에만 수정했을 때 어떤 주소가 올바른 주소인지 알 수 없습니다. 이렇게 중복된 데이터가 서로 일치하지 않는 경우를 데이터 불일치라고 합니다.

데이터 보안성(Data Security)의 결여

데이터가 저장된 파일은 일반적인 형태의 텍스트 파일이기 때문에 간단히 파일을 열어서 내용을 확인할 수 있습니다. 만약 기업 기밀이나 일기와 같은 사생활 정보를 기록한 경우 내용에 대한 보안을 유지하기가 힘듭니다. 실제로 대학 신입생 시절에 사용하던 텔넷 비비에스에 쪽지와 메일 서비스가 있었는데 이 내용이 모두 텍스트 형식으로 저장되어 있어서 많은 사람이 쪽지를 엿봤다는 소문이 있었습니다. 거기엔 밀회의 흔적들이나 아주 비밀스러운 것들이 많았을 텐데 말입니다.

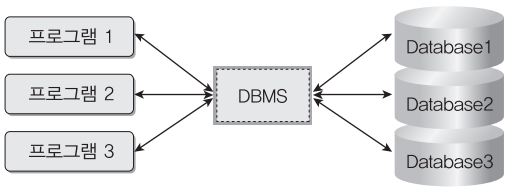


[그림 7-1] 파일 관리 시스템

데이터베이스 관리 시스템

파일 관리 시스템의 한계로 보다 효율적이고 편리하며 대용량의 데이터를 처리할 수 있는 시스템이 요구되면서 데이터베이스가 등장했습니다. 데이터베이스는 다수의 응용 프로그램에서 데이터를 공유하여 사용할 수 있도록 통합하여 저장한 데이터의 묶음을 의미합니다. 이러한 데이터베이스를 생성하거나 관리하고 사용자의 질의(쿼리, Query)에 대해 응답하는 프로그램의 집합을 데이터베이스 관리 시스템(DBMS)이라고 합니다. 그리고 데이터베이스와 DBMS를 통칭하여 데이터베이스 시스

템이라고 합니다. 그러나 정의와 달리 DBMS나 데이터베이스 시스템을 데이터베이스라고 짧게 말하는 경우가 많아서 용어에 주의할 필요가 있습니다.



[그림 7-2] 데이터베이스 관리 시스템

데이터베이스는 파일 관리 시스템의 단점을 극복하고자 만들어진 것이기 때문에 파일 시스템의 단점이 곧 데이터베이스의 장점이 됩니다. 데이터베이스의 대표적인 특징을 간추려보면 다음과 같습니다.

자기 기술성(Self-describing)

데이터베이스에는 데이터에 대한 데이터인 메타데이터(Meta-data)가 있습니다. 메타데이터는 데이터의 연관 정보를 저장하는 데이터로 데이터베이스가 어떻게 정의되어 있는지에 대한 정보를 가집니다. 메타데이터를 통하여 데이터베이스가 데이터베이스를 기술하는 것이 가능해지기 때문에 하나의 DBMS가 여러 개의 데이터베이스를 관리할 수 있게 해줍니다.

프로그램과 데이터의 분리

다수의 응용 프로그램은 데이터베이스 내의 데이터에 접근하기 위해서 DBMS를 사용합니다. DBMS는 각 응용 프로그램의 요청에 의해 데이터베이스에 데이터를 추가하거나 조회하는 등의 기능을 수행합니다. 이러한 DBMS의 중계 덕분에 데이터베이스의 구조가 변경되더라도 DBMS에서 그것을 대신 처리해주기 때문에 응용 프로그램 개발자는 프로그램을 수정할 필요가 없습니다. 예를 들어 MySQL의 기본적인 저장엔진인 MyISAM을 사용하던 시스템을 InnoDB로 변경한다고 해서 프로그램을 변경할 필요는 없습니다. 이처럼 프로그램과 데이터를 분리함으로써 데이터가 프로그램에 의존적이지 않고 독립성을 가질 수 있게 됩니다.

데이터 중복의 최소화

파일 관리 시스템은 중복된 데이터로 인해서 불필요한 공간을 소비하는 단점이 있었습니다. 이러한 데이터의 중복을 피하고자 정규화라는 과정을 통해서 데이터베이스를 구조화하고 뒤에서 배우게 될 키(Key)를 이용하여 중복된 데이터의 추가를 방지할 수 있습니다.

데이터의 무결성

데이터베이스는 데이터의 무결성을 보장하기 위해서 여러 가지 제약 조건(Constraints)을 부여할 수 있도록 하고 있으며 대표적으로 도메인 제약 조건(Domain Integrity Constraint)의 경우 숫자 형식과 문자 형식처럼 타입이 다른 값을 입력하지 못하도록 합니다. 데이터의 무결성을 보장함으로써 데이터의 정확성을 보장할 수 있게 됩니다.

데이터 공유 및 보호

데이터베이스는 여러 응용 프로그램이나 다수의 사용자에 의해서 접근되기 때문에 같은 데이터가 동시에 사용되거나 변경될 수 있도록 데이터의 일관성을 보장하고 있습니다. 또한 허용된 권한을 통해서만 데이터로 접근할 수 있기 때문에 저장된 데이터를 보호하고 안전하게 유지되도록 할 수 있습니다.

파일 관리 시스템이 보다 적절한 경우

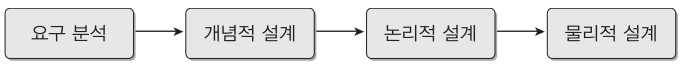
파일 관리 시스템의 단점이 곧 DBMS의 특징이자 장점입니다. 그러나 모든 경우에서 DBMS가 파일 관리 시스템보다 나은 것은 아닙니다. 만약 아주 단순하고 소량의 데이터가 기록되고 수정될 가능성이 매우 적으며 단일 프로그램에 의해서 접근된다면 파일 관리 시스템이 DBMS를 사용하는 것보다 성능이 우수할 수 있습니다. 왜냐하면 DBMS는 데이터베이스를 관리하기 위해서 다양한 관리 프로그램을 실행해야 하기 때문에 이러한 작업들이 부하(overhead)로 작용하기 때문입니다. 예를 들어 응용 프로그램의 버전 정보를 기록하고자 "1.xx"와 같이 단순한 몇 자리의 문자열을 저장하려고 데이터베이스를 사용하는 것은 과용이라고 할 수 있습니다. 웹 프로그램 중에서도 방문자의 수를 헤아리는 카운터의 경우 데이터가 단순하기 때문에 동시 사용자가 수십 수백 명이 되지 않는 이상, 데이터베이스를 이용하는 것보다는 파일을 이용하는 것이 훨씬 효율적입니다.

Section 02

데이터베이스 설계

데이터베이스 설계 과정

데이터베이스를 설계하는 과정은 다음과 같습니다.



[그림 7-3] 데이터베이스 설계 과정

요구 분석

어떤 서비스 혹은 기능을 구현하기 위해서 사용자(고객)로부터 업무에 대한 요구 사항을 전달받습니다. 이 과정은 매우 중요한데 일반적으로 사용자는 개발자가 아니므로 개발자와 의사표현의 문제에서 큰 어려움을 겪게 됩니다. 개발자와 사용자의 미묘한 시각의 차이는 추후 데이터베이스를 다시 설계하게 만들기도 합니다. 따라서 초기에 사용자가 원하는 것이 어떤 것인지 확실하게 정의를 하고 개발자가 이해하고 작성한 요구 사항 명세서를 바탕으로 사용자와 여러 차례에 걸쳐 논의하고 수정 보완하는 과정이 필요합니다.

개념적 설계

사용자의 요구 분석이 완료되면 개념적 데이터 모델을 이용하여 개체(Entity)와 속성(Attribute)을 정의하고 개체 간의 관계를 표현합니다. 이 단계는 특정 데이터베이스나 응용 프로그램에 종속적이지 않으며 사용자의 요구 사항을 추상화하는 과정입니다. 개념적 데이터베이스의 설계는 예를 들어 대학교를 모델링하는 경우 대학, 단과대학, 학과, 교수, 학생, 수업 등의 개체를 정의하고 학교의 이름이나 개교일, 단과대학 이름 등과 같은 각 개체의 속성을 정의하고 교수와 학생, 학생과 수업 등의 개체 간의 관계를 표현하는 일련의 과정을 의미합니다.

논리적 설계

논리적 설계 과정은 앞서 작성한 개념적 데이터베이스 모델을 바탕으로 실제 구현할 DBMS에 맞게 논리적 데이터 모델로 변형합니다. 이 과정을 데이터 모델 사상(Data Model Mapping) 과정이라고도 하며 개체는 테이블, 속성은 컬럼, 식별자는 기본키(primary key) 그리고 관계는 외부키(Foreign Key)로 표현하여 논리적 스키마(Schema)를 생성합니다.

물리적 설계

물리적 설계 과정은 논리적 모델을 바탕으로 실제 사용할 데이터베이스에 물리적으로 자료구조를 사상하는 단계입니다. 데이터베이스 성능 향상을 위한 인덱스를 생성하는 것도 물리적 설계 과정입니다.

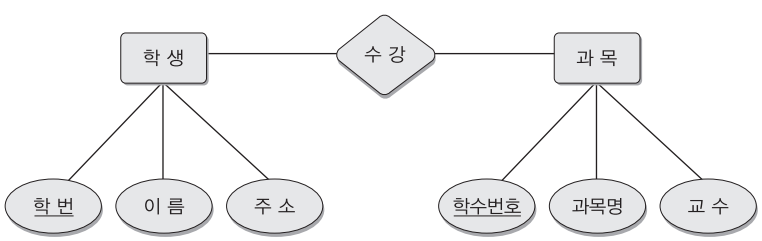
데이터베이스의 설계 과정은 사용자의 요구 사항을 바탕으로 개념적으로 개체와 관계를 표현하고 표현된 모델을 테이블이나 키 등을 통해서 실제 데이터베이스에서 사용할 수 있는 형태로 변환한

후 인덱스 등을 통해서 데이터베이스에 최적화하는 절차로 이루어집니다.

개체-관계 모델

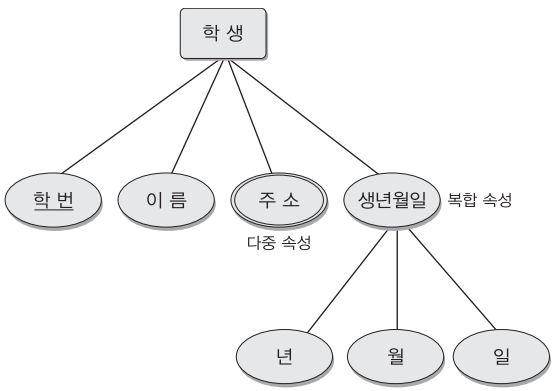
개념적 설계에서 사용되는 개념적 데이터 모델에는 여러 가지가 있지만 그중 가장 널리 사용되는 모델이 개체-관계 모델(ER Model, Entity-Relationship Model)입니다. 이 데이터 모델은 데이터베이스의 구조를 개체(Entity)와 속성(Attribute) 그리고 개체 간의 관계(Relationship)를 구성요소로 표현한 모델로 도형을 이용하여 표현한 것을 E-R 다이어그램이라고 합니다.

개체는 실세계를 추상화한 것으로 학생, 교수, 과목, 학과와 같이 유형적인 사물이나 무형적인 개념 모두를 의미합니다. 개체는 인스턴스(Instance)의 집합이며 학생 중 홍길동, 학과 중 컴퓨터학과, 과목 중 웹 프로그래밍과 같이 개체의 실제 값을 인스턴스라고 합니다. 속성은 개체의 특성을 나타내는 것으로 학생 개체는 이름, 나이, 학번, 소속학과 등과 같은 것입니다. 마지막으로 관계는 개체 간의 연관성을 의미하는 것으로 학생과 지도교수와의 관계, 학생과 학과와의 관계 등 개체 간의 연결 형태를 표현하는 것입니다.



[그림 7-4] 학생과 과목에 대한 E-R 다이어그램

그림에서 볼 수 있듯이 개체는 사각형으로 표현하고 그 속성은 타원형으로 표현합니다. 속성은 만약 허용한다면 주소에 여러 개의 주소, 즉 다중 값을 가질 수 있으며 생년월일과 같이 한 속성에 연도, 월, 일의 복합 속성을 가지는 것도 가능합니다. 특별히 다중 속성은 테두리가 두 줄인 타원을 사용합니다.



[그림 7-5] 다중 속성과 복합 속성

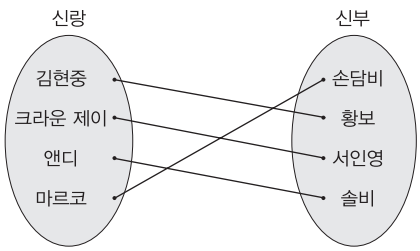
여러 가지 속성 중에서 밑줄로 표시한 "학번" 속성은 식별자를 의미하는데, 식별자란 한 개체에 대한 여러 개의 인스턴스를 서로 구분 지어주는 속성을 의미합니다. 예를 들면 학생을 한 명 찾고자 했을 때 어떤 속성을 통해서 그 사람만을 찾을 수 있는지를 생각해보면 됩니다. 학생의 이름을 통해서 찾고자 한다면 한 학과에 같은 이름을 가진 사람이 있을 때 단 한 명을 특정지을 수 없습니다. 따라서 학생의 이름은 식별자가 되지 못합니다. 만약 학번을 통해서 학생을 찾았다면 어떻게 되었을까요? 학번은 학생 1인당 하나씩 부여받은 번호이기 때문에 절대 중복되지 않습니다. 따라서 학번을 통해서 학생을 찾았다면 한 명을 특정지을 수 있을 것입니다. 학번과 같이 각각의 인스턴스를 식별할 수 있도록 해주는 속성이 식별자입니다. 식별자는 학번과 주민등록번호와 같이 한 개체에 여러 개가 있을 수 있으며 이 중에서 "이 속성을 통해서 인스턴스를 식별하겠다."고 지정한 속성을 기본키(Primary Key)라고 합니다. 다시 말해, 학번이나 주민등록번호는 모두 키가 될 수 있지만(이를 후보키(Candidate Key)라고 합니다.) 편의를 위해서 학번이나 주민등록번호 중 하나를 기본키로 선택하는 것입니다.

마지막으로 관계는 마름모꼴로 표현되며 3가지 형태의 관계 유형을 갖습니다.

일대일(1:1) 관계

일대일 관계는 두 개체의 인스턴스들이 각각 1대1 대응으로 연결된 것을 의미합니다. 예를 들면 신랑과 신부의 객체가 있고 결혼이라는 관계가 있을 때, 일부 일처제인 우리나라의 법으로는 반드시 신랑은 1명의 신부를 또한 신부는 단 1명의 신랑을 가질 수 있습니다. 따라서 신랑과 신부의 결혼 관계는 1대 1 관계를 가집니다.

그림을 보면 신랑과 신부는 각각 한 명의 배우자를 선택할 수 있습니다. 신랑이 두 명의 신부를 선택하거나 반대로 신부가 두 명의 신랑을 선택할 수 없습니다.



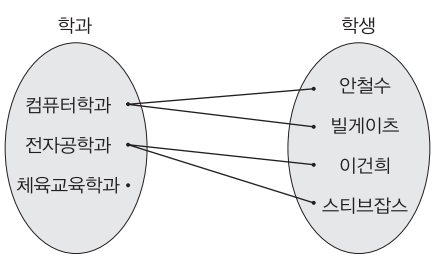
[그림 7-6] 1:1 관계의 인스턴스

인스턴스 간의 관계가 이러한 모습을 나타낼 때 우리는 1:1 관계라고 하며 E-R 다이어그램으로 다음과 같이 표현합니다.

[그림 7-7] 1:1 관계의 E-R 다이어그램

일대다(1:N) 관계

일대다 관계는 한 개체의 인스턴스가 다른 개체의 인스턴스와 여러 개의 연결을 가질 수 있을 때를 말합니다. 예를 들어 학과와 학생의 소속 관계를 생각해보면 학과에는 여러 명의 학생이 소속되어 있지만 한 학생은 하나의 학과에 소속되어 있습니다.



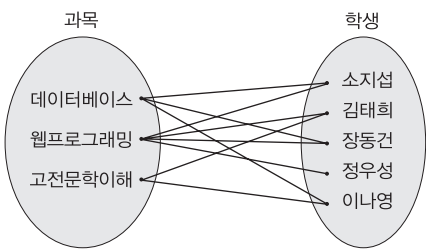
[그림 7-8] 1:N 관계의 인스턴스

1:N의 관계는 하나(1)의 인스턴스에 여러(N) 개의 인스턴스가 연결된다는 것을 의미합니다. 이 관계는 다음과 같이 E-R 다이어그램으로 표현할 수 있습니다.

[그림 7-9] 1:N 관계의 E-R 다이어그램

다대다(N:M) 관계

다대다 관계는 개체 간에 서로 중복적으로 연결을 갖는 경우를 의미합니다. 예를 들어 학생과 과목 간의 수강 관계를 생각해보면, 한 학생은 여러 개의 과목을 수강하고 한 과목에는 여러 명의 학생이 수업에 참여하게 됩니다.

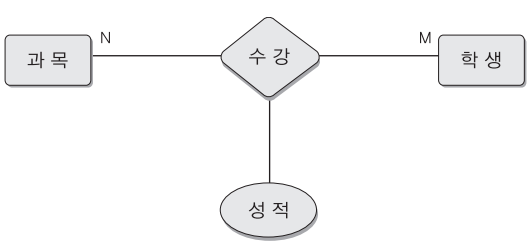


[그림 7-10] N:M 관계의 인스턴스

이처럼 각 개체의 인스턴스가 서로 다중의 연결을 허용하는 경우가 N:M 관계이며 이는 다음과 같이 E-R 다이어그램으로 표현됩니다.

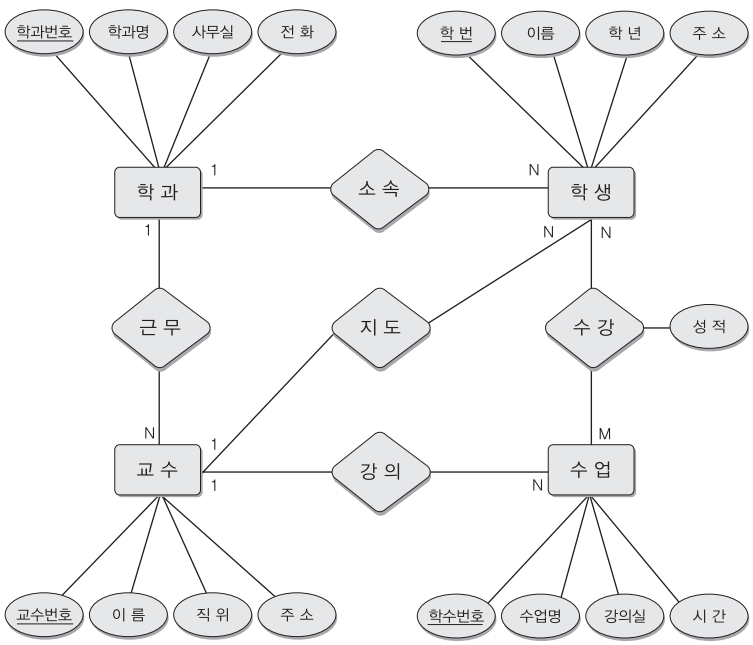
[그림 7-11] N:M 관계의 E-R 다이어그램

그리고 하나 더 알고 넘어가야 할 것이 있는데 개체뿐만이 아니라 관계 역시 속성을 가질 수 있다는 것입니다. 학생의 과목 성적을 생각해보면 성적은 과목과 학생의 연결로 인해서 발생하는 부가적 속성이므로 수강 관계에 성적 속성을 가지는 것이 바람직합니다. 만약 과목에 성적 속성을 추가하면 과목에는 수강하는 모든 학생에 대한 성적을 기록할 수 있는 다중 값 속성이 되어야 하고 마찬가지로 학생 개체의 속성으로 추가하려면 모든 수강 과목에 대한 성적을 기록할 수 있는 다중 값 속성이 되어야 합니다. 그래서 수강 관계에 속성을 추가하여 다시 E-R 다이어그램을 그려보면 다음과 같습니다.



[그림 7-12] 관계의 속성

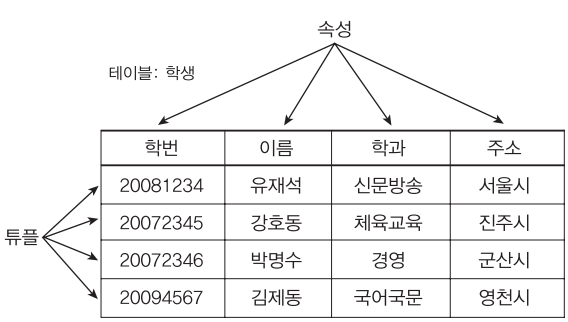
이러한 세 가지 관계의 유형은 앞으로 실제 데이터베이스에 적용하면서 테이블을 어떻게 생성해야 할지를 결정하는 데 많은 도움을 줍니다. 따라서 개체 간의 관계가 어떠한 유형을 갖는지를 잘 파악해두는 것이 매우 중요합니다. 학과, 교수, 학생, 수업 개체와 관계에 대한 간략한 E-R 다이어그램을 그려보면 다음과 같습니다.



[그림 7-13] 학과, 교수, 학생, 수업 개체에 대한 E-R 다이어그램

관계형 데이터 모델

개념적 데이터 모델에 개체-관계 데이터 모델이 있듯이 논리적 데이터 모델에는 관계형 데이터 모델(Relational Data Model)이 있습니다. 이 모델은 개체와 관계를 모두 2차원의 테이블 형태로 표현하는 방식으로 간단하고 직관적이어서 널리 이용되고 있습니다.



학번	이름	학과	주소
20081234	유재석	신문방송	서울시
20072345	강호동	체육교육	진주시
20072346	박명수	경영	군산시
20094567	김제동	국어국문	영천시

[그림 7-14] 관계형 데이터 모델의 표현

그림과 같이 테이블의 열은 속성을 나타내고 행은 튜플이라는 개체의 실제 인스턴스를 말합니다. 이러한 용어들은 다양한 형태로 불리는데 테이블은 릴레이션(Relation), 튜플은 레코드나 행(로우, row) 그리고 속성은 열(컬럼, column)이나 필드(field)로 불립니다.

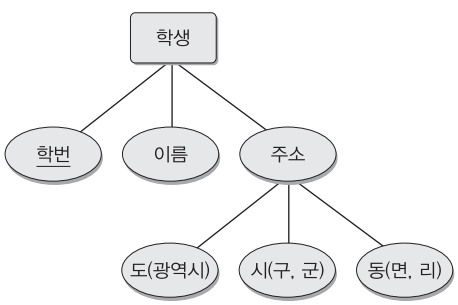
E-R 모델에서 관계형 데이터 모델로의 변환

개체-관계 데이터 모델에서 관계형 데이터 모델로 변환하는 방법에 대해서 알아보겠습니다. 관계형 데이터 모델은 앞서 말씀드린 바와 같이 개체와 관계를 테이블로 표현하는 방법입니다. 그런데 개체-관계 데이터 모델은 개념적인 접근이기 때문에 실제 테이블로 표현할 때 다음과 같은 규칙을 따라야 합니다.

규칙 1

개체는 각각 하나의 테이블을 형성한다.

이때 복합 속성은 부분 속성을 각각 하나의 속성으로 변환해야 합니다.



학번	이름	주소	도(광역시)	시(구, 군)	동(면, 리)

[그림 7-15] 개체의 테이블 변환

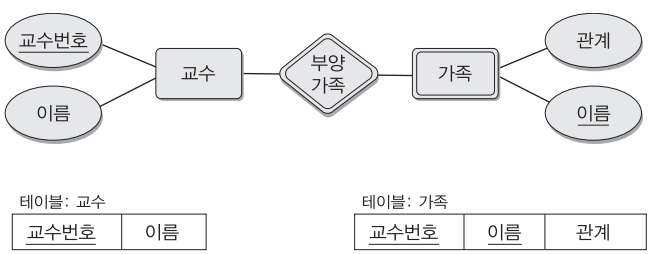
규칙 2

종속적인 개체를 테이블로 변환할 때는 소유 개체의 기본키를 속성으로 추가한다.

교수의 부양가족이라는 개체를 고려하였을 때 교수가 부양가족이 없는 경우에 개체 자체는 존재할 수 없게 됩니다. 또한 교수가 정년퇴직해서 삭제되면 부양가족인 해당 가족들이 기록에 남아 있으면 안 되는 게 당연합니다. 이처럼 어느 개체에 종속적인 개체를 약한 개체(Weak Entity)라고 합니다. 이와 대비하여 약한 개체를 소유한 개체를 강한 개체(Strong Entity)라고 부릅니다. 약한 개체

는 강한 개체와 구별하고자 테두리가 두 줄인 사각형 모양을 갖습니다. 그리고 약한 개체와 강한 개체를 연결해주는 관계를 식별 관계(Identifying Relationship)라고 하며 이도 마찬가지로 테두리가 두 줄인 마름모로 표시합니다.

강한 개체는 일반적인 개체와 같기 때문에 규칙 1번을 이용하여 테이블로 변환할 수 있습니다. 그러나 약한 개체는 반드시 강한 개체의 기본키를 새롭게 속성으로 가져야 합니다. 이러한 외부에서 온 키를 외래키(Foreign Key)라고 합니다. 바로 이 외래키가 교수와 가족 간의 관계인 부양가족이란 관계(식별 관계)를 표현하는 방법입니다. 가족 테이블에서는 한 인스턴스를 식별하기 위해서 두 개의 키를 사용해야 합니다. 이름은 언제나 중복할 수 있기 때문에 한 인스턴스를 식별할 수 없습니다. 그러나 교수의 가족 중에는 같은 이름을 가진 사람이 존재하지 않으므로 교수번호와 이름을 같이 이용하면 한 인스턴스를 식별할 수 있습니다.

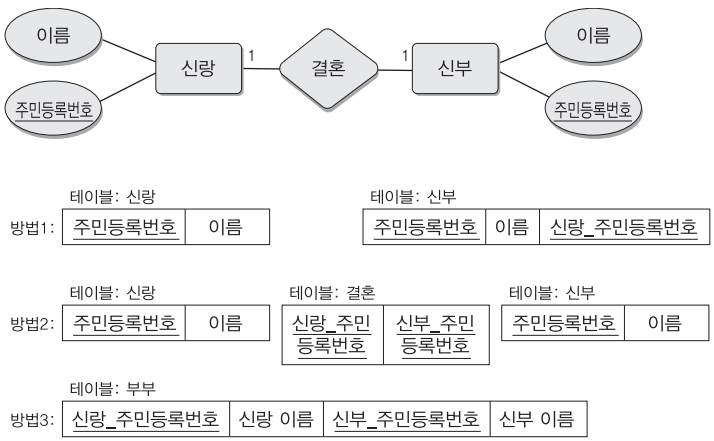


[그림 7-16] 약한 개체의 변환

규칙 3

1:1 관계는 어느 한 쪽의 테이블에 외래키를 추가하거나 관계 정보만을 갖는 관계 테이블을 생성한다.

1:1 관계를 가지는 두 개체는 관계를 표현하기 위해서 어느 한 쪽에 상대방의 기본키를 속성으로 추가하면 됩니다. 그렇지 않다면 관계를 각 개체의 기본키를 속성으로 갖는 테이블로 생성할 수 있습니다. 만약, 두 개체의 인스턴스들이 모두 참여하고 있다면 즉, 연결되지 않은 인스턴스가 없다면 두 개체를 하나의 개체로 합치는 것도 고려할 수 있습니다.

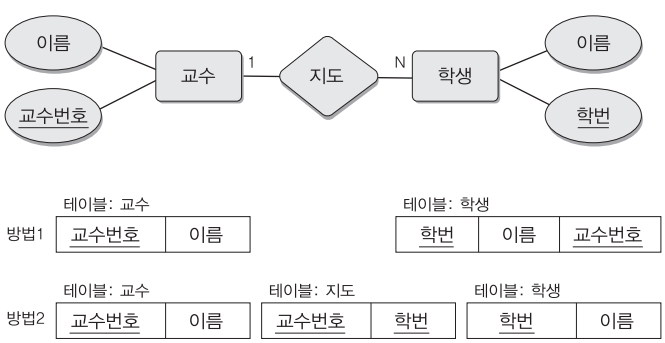


[그림 7-17] 1:1 관계의 테이블 변환

규칙 4

1:N 관계는 N측의 개체에 외래키를 추가하거나 관계 정보를 갖는 관계 테이블을 생성한다.

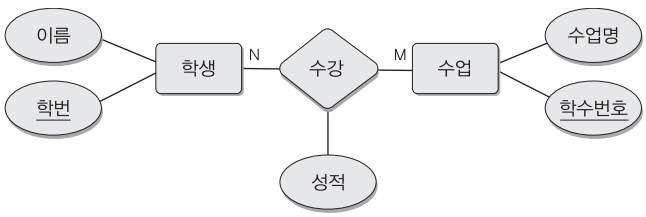
한 개의 인스턴스가 여러 개의 인스턴스와 연결되므로 1측에 외래키를 추가하면 한 속성에 여러 개의 값을 입력하기 위해 다중 값을 허용해야 합니다. 그러나 관계 데이터베이스의 특성 중의 하나는 한 속성은 반드시 하나의 값을 가져야 하므로 이 방법은 바람직하지 않습니다. 따라서 N측에 1측 기본키를 추가하여 관계를 표현하도록 합니다. 이외에 1:1 관계에서와 마찬가지로 두 개체의 기본키를 속성으로 가지는 관계 테이블을 생성하는 방법을 사용할 수도 있습니다.



[그림 7-18] 1:N 관계의 테이블 변환

규칙 5

M:N 관계는 각 개체의 기본키를 속성으로 하는 관계 테이블을 생성한다.



테이블: 학생

학번	이름

테이블: 수강

학번	학수번호	성적

테이블: 수업

학수번호	수업명

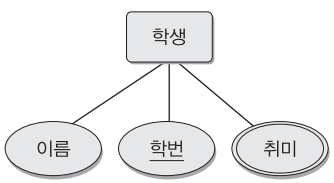
[그림 7-19] N:M 관계의 테이블 변환

만약 그림과 같이 관계에 속성이 존재한다면 속성은 관계 테이블의 속성으로 추가하면 됩니다.

규칙 6

다중 값 속성은 기본키를 외래키로 갖는 별도의 테이블을 생성한다.

앞에서도 잠깐 언급하였지만 관계형 데이터베이스에서 한 속성에는 하나의 값만을 가질 수 있기 때문에 다중 값 속성을 제거해야 합니다. 이를 위해서 기본키를 외래키로 가지는 다중 값 속성을 위한 새로운 테이블을 생성합니다.



테이블: 학생

학번	이름

테이블: 취미

학번	취미

[그림 7-20] 다중 값 속성의 테이블 변환

정규화

데이터베이스의 사용 목적은 중복된 데이터를 최소화하고 이를 통해서 저장 공간의 낭비를 줄여 데이터를 보다 효율적으로 기록하고 검색할 수 있게 해주는 것입니다. 그런데 잘못된 데이터베이스 설계는 이러한 장점들을 모두 무색하게 만들어 버리기 때문에 정규화(Normalization)를 통해서 보다 향상된 형태의 데이터베이스 스키마를 설계할 필요가 있습니다.

정규화는 기본적으로 올바르지 못한 테이블을 분할하여 더 작은 형태의 올바른 테이블로 분해하

는 것을 의미합니다. 올바르지 못한 테이블이란 데이터를 삽입하거나 갱신하거나 삭제할 때 데이터의 중복이나 손실 또는 모순이 발생하는 테이블을 의미합니다. 이러한 상황을 각각 삽입 이상(Insertion Anomaly), 갱신 이상(Update Anomaly) 그리고 삭제 이상(Deletion Anomaly)이라고 합니다.

예를 통해서 어떠한 경우에 이러한 현상이 나타나는지 알아보도록 하겠습니다. 다음은 학생 정보와 수강하는 과목 정보를 기록하는 테이블입니다.

학번	학년	이름	수강과목	강사
200701	2	김국진	A100	배삼룡
200602	3	김용만	B101	김한국
200701	2	김국진	B101	김한국
200801	1	박수홍	C100	김미화
200802	1	김수용	D100	이경규
200602	3	김용만	E109	이경규
200801	1	박수홍	D100	이경규
200701	2	김국진	D100	이경규

[그림 7-21] 학생-수강 테이블

① 삽입 이상

테이블에 새로운 튜플을 추가하고자 할 때 발생하는 이상 증상을 삽입 이상이라고 합니다. 새로운 신입생이 입학했을 때, 아직 수강신청을 하지 않은 상태이므로 수강과목은 존재하지 않습니다. 그런데 이 테이블에서 키는 학번, 수강과목의 조합이므로 수강과목에는 빈값(NULL)이 입력될 수 없습니다. NULL 값을 허용하게 되면 NULL 값을 가지는 여러 개의 튜플을 더는 식별할 수 없기 때문입니다. 그래서 신입생이 수강 신청을 하기 전까지는 테이블에 튜플을 추가할 수 없는 상황이 발생합니다. 만약 반드시 입력해야 하는 상황이라면 수강과목 필드에는 임시적인 과목 번호를 입력해야 할 것입니다. 이렇게 삽입을 하고자 할 때 발생하는 이상 현상을 삽입 이상이라고 합니다.

② 갱신 이상

튜플의 특정 필드 값을 수정하고자 할 때 발생하는 이상 증상을 갱신 이상이라고 합니다. 위의 테이블에서 김용만 학생이 유급으로 2학년으로 다시 되돌아갔다고 하면, 모든 김용만의 튜플을 찾아서 수정을 해주어야 합니다. 그런데 만약 모두 수정하지 않고 하나만 수정하였다면 튜플 간에 데이터가 일관성이 없어져서 정보에 모순이 생길 수 있습니다. 즉, 어떤 튜플을 검색하면 2학년이라고 나오고 또 다른 튜플에는 3학년이라고 나오기 때문에 어떤 정보도 신뢰할 수 없게 됩니다. 이렇게 한 튜플을 수정하면서 데이터의 일관성이 없어지는 경우를 갱신 이상이라고 합니다.

③ 삭제 이상

특정 튜플을 삭제하고자 할 때 발생하는 이상 증상을 삭제 이상이라고 합니다. 위의 테이블에서

> 김수용이 D100 수업을 수강 취소하면 해당 튜플이 삭제되고 그로 인해서 김수용 학생의 정보가 없어지므로 김수용이라는 학생이 존재하지 않는 것과 같아집니다. 이렇듯 특정 튜플을 삭제할 때, 삭제되어서는 안 되는 다른 정보까지 제거되는 경우를 삭제 이상이라고 합니다.

이러한 이상 증상을 제거하고 데이터의 중복을 최소화하기 위해서 정규화를 시행하며 정규화를 통해 제1, 제2, 제3 정규형과 보이스-코드 정규형, 제4, 제5 정규형, 도메인-키 정규형 그리고 제6 정규형까지 다양한 정규형으로 테이블을 분할할 수 있습니다. 정규형의 차수가 높아질수록 데이터의 중복은 최소화되고 이상 증상은 없어지며 테이블은 더 작은 단위로 분할됩니다. 그러나 너무 작게 나누어진 테이블을 조회하게 될 때 이를 다시 잇는 작업에 부하가 걸리게 되어 성능의 저하가 발생합니다. 그래서 성능을 위해서 높은 차수의 정규형을 다시 낮은 차수의 정규형으로 되돌리는 역정규화(denormalization)를 수행하기도 합니다. 즉, 높은 차수의 정규형이 무조건 좋은 것이 아니라 성능과 데이터의 중복을 고려하여 알맞은 정규형으로 변환하는 것이 중요합니다. 일반적으로 제3 정규형 혹은 제3 정규형의 특별한 형태인 보이스-코드 정규형 정도를 염두해두고 정규화를 하는 것이 바람직합니다. 따라서 모든 정규형을 다루지 않고 보이스-코드 정규형까지만 다루도록 하겠습니다.

제1 정규형(First Normal Form: 1NF)

테이블의 모든 속성은 원자 값을 갖는다.

앞서 데이터 모델을 배우면서 다중 값을 가지는 속성이나 복합 속성을 갖는 경우를 보았습니다. 이러한 속성들은 한 속성 내에 하나의 값만을 가져야 한다는 제1 정규형 조건에 어긋납니다. 따라서 다중 값 속성이나 복합 속성을 제1 정규형에 맞게 변환해 주어야 합니다. 그런데 우리는 이미 관계형 데이터 모델로 변환하면서 다중 값 속성과 복합 속성을 제거했으므로 그 결과 이루어진 모든 테이블은 제1 정규형을 만족하게 됩니다.

제2 정규형(Second Normal Form: 2NF)

테이블이 제1 정규형을 만족하고 키가 아닌 모든 속성이 기본키에 완전 함수 종속이다.

어떤 속성이 다른 속성 혹은 하나 이상의 속성에 의해서 고유하게 결정된다면 결정짓는 속성을 결정자(Determinant)라고 하고 결정지어지는 속성을 종속자(Dependent)라고 합니다. 이처럼 특정 속성이 다른 속성들을 결정하는 것을 함수적으로 종속되어 있다고 합니다.

예를 들어 학생의 학번을 알면 학생의 이름과 학과, 학년, 주소 등을 알 수 있습니다. 그러나 학생의 이름을 안다고 하면 같은 이름을 가진 학생과 구분할 수 없으므로 그 사람의 다른 정보를 알 수 없습니다. 마찬가지로 학과나 학년, 주소만으로는 다른 속성 정보를 알 수 없습니다. 이렇듯 다른 속

성을 유일하게 결정짓는 속성을 결정자라고 하며 결정자에 의해 결정지어지는 속성들을 종속자라고 합니다. 즉, 학생의 이름, 학과, 학년 등은 학번에 함수적으로 종속되어 있다는 것입니다.

완전 함수 종속이란 결정자에서 속성을 하나라도 제거했을 때 더 이상 어떤 속성도 결정짓지 못하는 경우를 말하며 부분 함수 종속이란 속성이 하나 이상 제거되어도 여전히 다른 속성을 결정지을 수 있을 때를 말합니다.

앞에서 사용한 테이블을 다시 한번 살펴봅시다.

이 테이블은 원자값만을 사용하고 있으므로 제1 정규형입니다. 이 테이블의 키는 {학번, 수강과목}인데 {학번, 수강과목}은 학년, 이름, 강사를 모두 결정 짓고 있지만 학년과 이름은 학번만으로 결정이 가능하고 강사는 수강과목만으로 결정이 가능합니다. 따라서 다른 속성들이 부분적으로 종속되므로 키가 아닌 모든 속성이 결정자에 의해 완전 함수 종속이어야 한다는 제2 정규형을 만족하지 않습니다.

학번	학년	이름	수강과목	강사
200701	2	김국진	A100	배삼룡
200602	3	김용만	B101	김한국
200701	2	김국진	B101	김한국
200801	1	박수홍	C100	김미화
200802	1	김수용	D100	이경규
200602	3	김용만	E109	이경규
200801	1	박수홍	D100	이경규
200701	2	김국진	D100	이경규

[그림 7-22] 제1 정규형 테이블

왜 제2 정규형으로 변환해야 하는지를 살펴보면, 제1 정규형 테이블은 불필요하게 중복된 정보를 가지고 있습니다. 학번, 학년, 이름이 여러 번 반복되고 수강과목과 강사의 정보 또한 중복되어 있습니다. 그래서 앞서 언급했듯이 이 테이블은 삽입이나 수정, 삭제에서 이상 증상을 일으키게 됩니다. 이러한 이상 증상과 중복된 데이터를 없애기 위해서 제2 정규형으로 변환을 해보면 다음과 같습니다.

테이블: 학생

학번	학년	이름
200701	2	김국진
200602	3	김용만
200801	1	박수홍
200802	1	김수용

테이블: 수강

학번	수강과목
200701	A100
200602	B101
200701	B101
200801	C100
200802	D100
200602	E109
200801	D100
200701	D100

테이블: 수업

수강과목	강사
A100	배삼룡
B101	김한국
C100	김미화
D100	이경규
E109	이경규

[그림 7-23] 제2 정규형으로 변환된 테이블

정규화하는 방법은 부분 종속이 없어지도록 각 테이블에 결정자를 하나만 두면 됩니다.

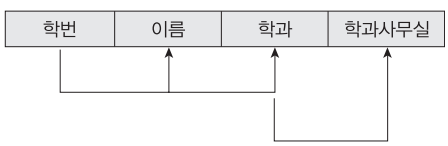
학생 테이블에서 학년과 이름은 학번에 완전 함수 종속이며, 수업 테이블 또한 강사 속성이 수강과목에 대해 완전 함수 종속입니다. 수강 테이블은 결정자가 아닌 속성이 없으며 만약 성적이라는 속성을 추가했다면 성적 속성은 {학번, 수강과목} 결정자에 대해 완전 함수 종속입니다.

제2 정규형으로 변환 결과 테이블에서 불필요하게 중복되었던 데이터들이 줄어든 것을 확인할 수 있습니다.

제3 정규형(Third Normal Form: 3NF)

테이블이 제2 정규형을 만족하고 키가 아닌 모든 속성이 기본키에 이행적으로 종속되지 않는다.

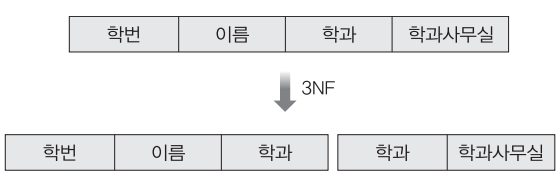
이행 종속이란 결정자 A가 B 속성을 결정짓고 결정지어진 B 속성이 다시 다른 속성 C를 결정짓는 것을 말합니다. 이는 기호로 A → B, B → C라고 표현하며 결국 A → C가 됩니다. 즉, A가 B를 결정짓고 B는 다시 C를 결정지으니 결국 A는 C를 결정짓는 것과 같다라는 의미입니다.



[그림 7-24] 이행 종속의 예

그림에서 보이듯이 학번이 키가 되어 이름과 학과를 결정짓고 학과가 다시 학과 사무실을 결정지으므로 이행 종속 규칙에 따라 학번이 이름, 학과, 학과사무실 모두를 결정짓게 됩니다. 그러나 같은 과의 다른 학생들도 모두 학과와 학과사무실 정보를 갖기 때문에 학과사무실 정보가 중복적으로 입력됩니다.

이상 현상을 없애고자 이행 종속이 일어나는 모든 속성을 새로운 테이블로 분리하면 제3 정규형으로 변환할 수 있습니다.



[그림 7-25] 제3 정규형으로 변환된 테이블

보이스-코드 정규형(Boycd-Codd Normal Form: BCNF)

모든 결정자가 후보키이다.

후보키는 앞서 언급한 적이 있는데 학번과 주민등록번호와 같이 어느 것이나 기본키가 될 수 있는 속성을 의미합니다.

보이스-코드 정규형은 제2, 제3 정규형과는 별개이기 때문에 제2, 제3 정규화를 하지 않아도 곧바로 보이스-코드 정규형을 만들 수 있습니다. 그러나 보이스-코드 정규형을 완성하면 이는 모두 제3 정규형을 만족합니다. 그렇다고 제3 정규형이 모두 보이스-코드 정규형을 만족하는 것은 아닙니다. 왜냐하면 보이스-코드 정규형은 제3 정규형의 특별한 경우이기 때문입니다.

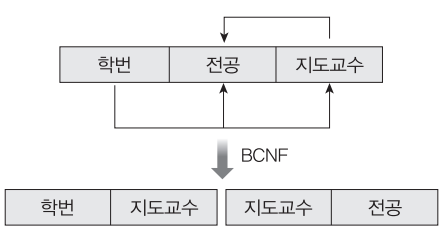
두 정규형의 차이를 알아보기 위해서 제3 정규형이면서 보이스-코드 정규형이 아닌 예를 보며 살펴봅시다.

학번	전공	지도교수
1234	컴퓨터	강호동
1234	경영	유재석
2345	컴퓨터	강호동
3456	컴퓨터	김제동

[그림 7-26] 제3 정규형 테이블

복수 전공 정보를 나타내는 테이블입니다. 이 테이블은 키가 아니면서 완전 종속이 아닌 속성은 없으며 이행 종속도 없으므로 제3 정규형을 만족합니다. {학번, 전공}이 테이블의 키가 되는데 지도교수가 전공을 결정짓는 결정자가 되므로 결정자가 모두 후보키여야 한다는 조건을 위배하게 되어 보이스-코드 정규형은 아닙니다.

이 테이블은 컴퓨터 전공인 강호동 교수처럼 중복된 데이터가 존재하므로 이상 현상이 발생할 수 있습니다. 이상 현상을 제거하고자 제3 정규형을 보이스-코드 정규형으로 변환하면 다음과 같습니다.



[그림 7-27] 보이스-코드 정규형으로의 변환

Section 03

SQL

관계형 데이터 모델을 작성할 수 있게 되었으니 실제 데이터베이스에 물리적으로 데이터 구조를 생성하고 조작할 수 있는 SQL에 대해서 알아봅시다.

SQL(Structured Query Language)은 데이터베이스와 의사소통을 하기 위한 표준 언어로 데이터를 검색하거나 관리하고 또한 데이터베이스를 생성하거나 수정하고 관리하는 목적으로 개발되었습니다. 실제 사용되는 관계형 데이터베이스 대부분이 이를 지원하고 있으며 데이터베이스 개발 업체마다 표준 SQL을 확장하여 사용하기 때문에 약간의 명령어 차이가 있습니다. 그래서 이 책에서는 우리가 실제로 사용할 MySQL에서 이용할 수 있는 SQL 문만을 다루도록 하겠습니다.

기본적으로 SQL 명령어는 기능에 따라서 데이터베이스를 정의하거나 변경하는 데이터 정의어(DDL, Data Definition Language)와 데이터의 삽입, 삭제, 검색, 수정을 처리하는 데이터 조작어(DML, Data Manupulation Language) 그리고 데이터베이스에 대한 권한을 제어하는 데이터 제어어(DCL, Data Control Language)로 분류합니다.

분류	명령어
데이터 정의어	CREATE, DROP, ALTER
데이터 조작어	SELECT, INSERT, UPDATE, DELETE
데이터 제어어	GRANT, REVOKE

[표 7-2] SQL 문의 분류

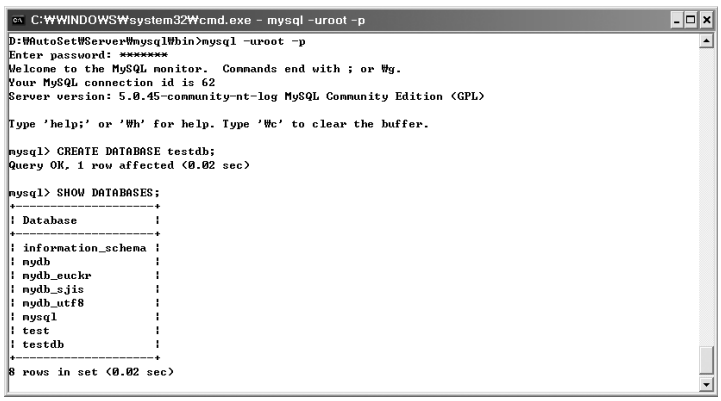
데이터베이스 생성

데이터베이스를 생성하려면 데이터를 정의하기 위한 DDL 명령어 중에서 CREATE 문을 사용해야 합니다. 그런데 CREATE 문을 이용하여 데이터베이스를 생성하려면 우선 데이터베이스를 생성할 수 있는 권한이 있어야 합니다. 권한을 부여하는 방법은 다음에 다루기로 하고 일단은 설치할 때 알고 있는 관리자 계정을 이용하여 생성합니다.

```
CREATE DATABASE testdb;
```

데이터베이스를 생성하였다면 제대로 생성되었는지 확인해 봅시다.

```
SHOW DATABASES;
```



[그림 7-28] 데이터베이스 생성

데이터베이스 목록에 testdb가 새로 추가된 것을 확인할 수 있습니다.

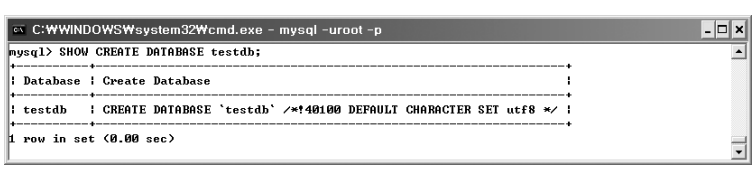
데이터베이스 수정

등록한 데이터베이스의 정보를 수정하기 위해서 ALTER 문을 사용합니다.

우리가 앞서 생성한 데이터베이스는 데이터베이스 기본 설정에 따라서 문자셋이 정해집니다. 설치 프로그램에 따라 설정이 다르지만 대개 utf8이나 latin1 혹은 너무나 배려 깊은 경우 euckr 문자셋을 기본으로 합니다. 일본어나 중국어와 같은 다양한 언어를 지원하려면 utf8 문자셋을 사용해야 하고 영어와 한국어만 사용된다면 euckr 문자셋을 사용하는 것이 좋습니다.

ALTER 문을 이용하여 데이터베이스의 문자셋을 변경해 보겠습니다. 우선 현재 생성된 testdb 데이터베이스가 어떤 문자셋을 사용하고 있는지 알아봅시다.

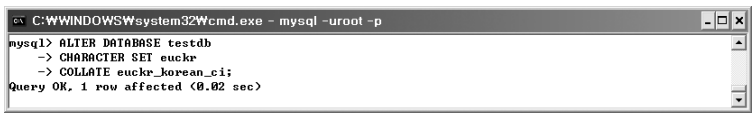
```
SHOW CREATE DATABASE testdb;
```



[그림 7-29] 데이터베이스 문자셋 확인

기본 문자셋이 utf8로 설정되어 있습니다. 이제 한국어 문자셋인 euckr로 변경해 보겠습니다.

```
ALTER DATABASE testdb
CHARACTER SET euckr
COLLATE euckr_korean_ci;
```

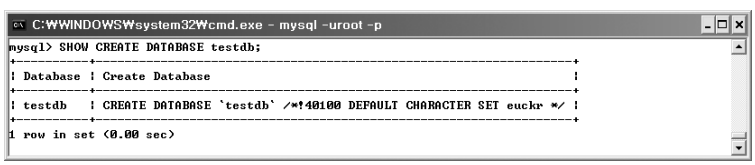


[그림 7-30] 데이터베이스 문자셋 변경

여기서 COLLATE 항목은 콜레이션(Collation)이라는 데이터를 저장하면서 문자를 정렬하는 방법으로 예를 들면 정렬에서 "가" 문자가 "나" 문자보다 우선한다는 것을 표현하는 것입니다. 한국어는 euckr_korean_ci 콜레이션을 사용하면 됩니다.

실제로 문자셋이 변경되었는지 확인해 봅시다.

```
SHOW CREATE DATABASE testdb;
```



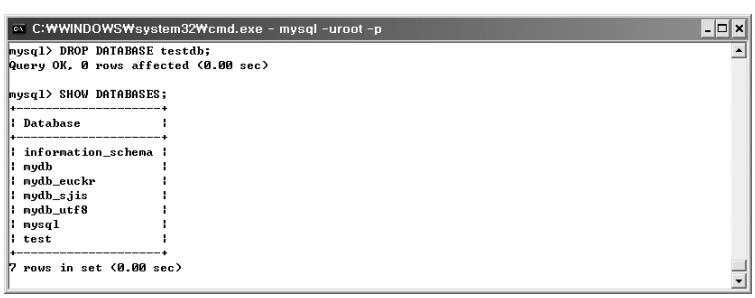
[그림 7-31] 데이터베이스의 수정된 문자셋 확인

아직 테이블을 생성하지는 않았지만 이제 한글 데이터를 입력하면 글자가 깨지지 않고 제대로 보입니다.

데이터베이스 삭제

등록된 데이터베이스를 제거하려면 DROP 문을 사용합니다. DROP 문을 사용하여 데이터베이스를 제거하는 경우 데이터베이스 내의 모든 정보가 사라지기 때문에 삭제하기 전에 반드시 필요한 데이터가 없는지 확인해야 합니다.

```
DROP DATABASE testdb;
```



[그림 7-32] 데이터베이스의 삭제

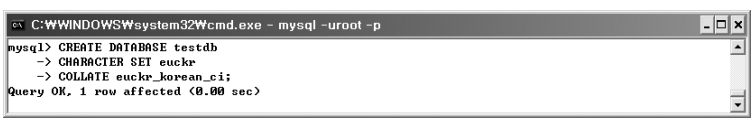
위험한 명령문임에도 불구하고 너무나 간단한 과정이기 때문에 더욱 조심해야 합니다.

테이블 생성

테이블을 추가하려면 데이터베이스와 마찬가지로 CREATE 문을 사용하면 됩니다. 그런데 테이블은 데이터베이스 내에 만들어지는 구조이기 때문에 먼저 데이터베이스가 존재해야 합니다. 우리는 앞서 데이터베이스를 제거했기 때문에 새롭게 데이터베이스를 생성해보도록 하겠습니다.

지금 여러분은 저에게 불만을 토로하고 계실 것으로 생각합니다. "왜? 다시 생성할 데이터베이스를 지우게 했느냐?"라고 말입니다. 그래서 더욱 간단하게 데이터베이스를 생성하는 법을 알려 드리겠습니다.

```
CREATE DATABASE testdb
CHARACTER SET euckr
COLLATE euckr_korean_ci;
```

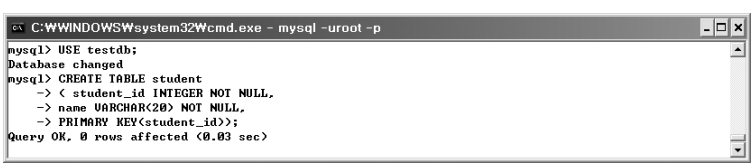


[그림 7-33] 문자셋을 지정하여 데이터베이스 생성

데이터베이스를 생성했으니 데이터베이스를 선택하고 testdb에 간단한 테이블을 추가해 보겠습니다.

```
USE testdb;

CREATE TABLE student
 ( student_id  INTEGER  NOT NULL,
name   VARCHAR(20)  NOT NULL,
PRIMARY KEY(student_id));
```

실제로 데이터베이스에 적용해보면 다음과 같습니다.



[그림 7-34] 테이블의 생성

테이블을 생성하려면 테이블의 이름과 속성을 정의하고 속성별로 어떤 데이터형을 사용할 것인지를 결정해야 합니다. MySQL에서는 약 30개의 데이터형을 제공하고 있으며 숫자, 문자, 텍스트, 날짜, 시간과 같은 다양한 형태로 분류되어 있습니다. 다양하게 데이터형을 제공하는 이유는 데이터베이스의 공간을 보다 효율적으로 사용하기 위해서입니다. 만약 편의를 위해서 모든 값을 넣을 수 있도록 매우 큰 공간을 부여하고 단지 그 속에 두 자리 숫자만을 기록한다면 엄청난 공간 낭비가 아닐 수 없습니다. 그래서 실제 사용 용도에 따라서 알맞은 데이터형을 선택할 수 있도록 다양한 데이터형을 제공하고 있습니다.

실제로 자주 사용하는 데이터형에 대해서 살펴보겠습니다.

데이터형	설명
BOOL, BOOLEAN	참 거짓. 0은 거짓이고 다른 값은 참
TINYINT(M)	매우 작은 정수, –128~127(또는 0~255)
SMALLINT(M)	작은 정수, –32768~32767(또는 0~65535)
MEDIUMINT(M)	중간 크기 정수, –8388608~8388607(또는 0~16777215)
INT(M), INTEGER(M)	일반 정수, –2147483648~2147483647(또는 0~4294967295)
BIGINT(M)	큰 정수, –9223372036854775808~9223372036854775807 (또는 0 ~ 18446744073709551615)
FLOAT(M,D)	작은 부동소수점수, M은 전체 자릿수, D는 소수점 뒷자리 –3.402823466E+38 ~ –1.175494351E–38 0, 1.175494351E–38 ~ 3.402823466E+38
DOUBLE(M,D)	일반 부동소수점수 –1.79769313486231 57E+308 ~ –2.2250738585072014E–308, 0, 2.2250738585072014E–308 ~ 1.7976931348623157E+308

[표 7-3] 숫자 데이터형

숫자 데이터형은 자릿수(M)를 지정할 수 있으며 자릿수를 지정하지 않으면 표에 표시된 허용 범위의 수로 지정됩니다. 숫자형은 UNSIGNED를 추가하여 기호 없는 수로 변경할 수 있으며 ZEROFILL을 추가하여 "0005"와 같이 0으로 채워서 입력할 수도 있습니다.

데이터형	설명
CHAR(M)	0~255 길이의 문자열
VARCHAR(M)	MySQL 5.0.3 버전 이전에는 0~255 길이의 문자열 MySQL 5.0.3 버전 이후에는 0~65535 길이의 문자열
TINYTEXT	255 길이의 텍스트
TEXT(M)	최대 65535 길이의 텍스트
MEDIUMTEXT	최대 16,777,215 길이의 텍스트
LONGTEXT	최대 4GB 크기의 텍스트

TINYBLOB	최대 255 길이의 BLOB
BLOB(M)	최대 65535 길이의 BLOB
MEDIUMBLOB	최대 16,777,215 길이의 BLOB
LOGBLOB	최대 4GB 크기의 BLOB
ENUM('값1', '값2',…)	등록된 값 중에서 하나의 값을 선택하는 데이터형, 최대 65535개의 개별 값을 가질 수 있다.

[표 7-4] 문자열 데이터형

문자열 데이터형도 문자열의 길이를 제한할 수 있으며 만약 허용 범위를 넘어서는 길이를 입력했을 때는 내부적으로 더 큰 단위의 데이터형으로 전환됩니다. 예를 들어 VARCHAR(500)인 경우에 MySQL 5.0.3 이전 버전에서는 255 길이가 최대 길이가 되기 때문에 더 큰 데이터형인 TEXT 타입으로 전환됩니다.

문자열 데이터형에서 BLOB(Binary Large OBject)은 바이너리 형식의 값을 저장할 수 있는 데이터형으로 이미지 파일이나 워드문서 등 바이너리 형태의 데이터를 저장할 수 있습니다. 그러나 이 데이터형은 내부의 값에 대한 검색이 되지 않습니다. ENUM은 어떤 문자열이 지정된 값 중에서 하나가 선택될 때, 매우 유용하게 사용할 수 있습니다. 예를 들어 3개의 도시 중에서 하나를 입력해야 하는데 두 개는 3자리 이름이고 하나가 50자리 이름이라면 이 50자리 도시 이름 때문에 데이터형을 50자리로 잡기에는 공간의 낭비가 심해집니다. 이때 ENUM 항목을 이용하여 세 도시명을 등록해두면 실제 저장될 때는 등록된 도시의 정수형 순번 값이 저장되기 때문에 공간을 보다 효율적으로 사용할 수 있습니다. 또한 요일이나 숫자를 표현하면서 요일을 문자열로 표현하더라도 실제로 정수값으로 저장되기 때문에 보다 편리하게 사용할 수 있습니다. 그러나 ENUM에 등록되지 않은 값을 입력하는 경우 에러가 발생하기 때문에 변동이 심한 경우라면 ENUM을 사용하는 것이 오히려 번거로워질 수도 있습니다.

데이터형	설명
DATE	1000-01-01 ~ 9999-12-31 범위의 날짜
DATETIME	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59 범위의 날짜와 시간
TIMESTAMP	1970-01-01 00:00:01 ~ 2038-01-09 03:14:07 범위의 날짜와 시간
TIME	-838:59:59 ~ 838:59:59 범위의 시간
YEAR	1901~2155 범위

[표 7-5] 날짜 및 시간 데이터형

다시 테이블 생성문으로 되돌아가서 데이터형 외에도 속성값이 빈 값을 허용할지의 여부에 대한 NULL, NOT NULL 옵션과 값을 입력하지 않았을 때 초기값을 지정하는 DEFAULT, 키 등에서 1씩 자동으로 증가하는 번호를 위한 AUTO_INCREMENT 등의 옵션이 있습니다. 그리고 속성이 문자열 데이터형일 때 속성의 콜레이션을 지정할 수도 있습니다.

테이블의 기본키를 설정하려면 PRIMARY KEY 키워드를 이용하면 됩니다.

테이블 수정

테이블 정보를 수정하려면 ALTER 문을 사용합니다. ALTER 문을 이용하면 문자셋을 변경하거나 새롭게 속성을 추가, 삭제하고 속성의 데이터형을 변경하는 등 다양한 작업을 할 수 있습니다.

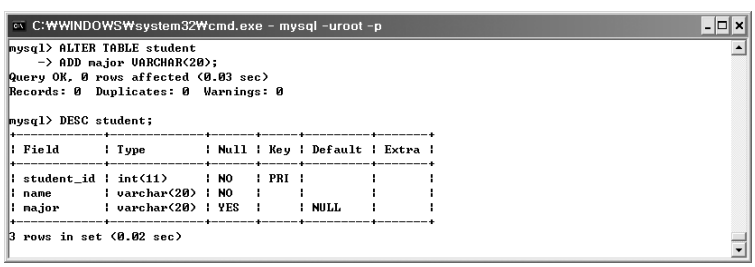
ALTER 문을 이용하여 student 테이블에 새로운 전공 속성을 추가해 보도록 하겠습니다.

```
ALTER TABLE student
ADD major VARCHAR(20);
```

테이블에 새로운 속성이 제대로 추가되었는지 확인하기 위해서 DESC 명령어를 이용하여 테이블 정보를 확인해 봅시다.

```
DESC student;
```

DESC 혹은 DESCRIBE 명령어는 정의된 테이블의 정보를 테이블 형태로 정리하여 보여주기 때문에 매우 유용한 명령어입니다. 아래의 그림에서 보이듯이 속성의 정보와 데이터형 그리고 우리가 추가했던 여러 가지 속성들이 잘 정리되어 보이는 것을 알 수 있습니다.



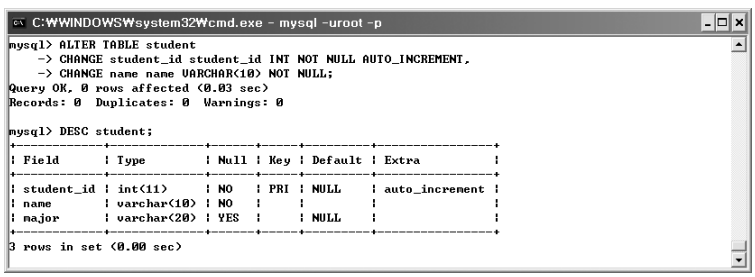
[그림 7-35] 테이블의 필드 추가

이 테이블의 기본키인 student_id 값을 자동으로 1씩 증가하도록 설정하고 name 속성의 길이 제한을 10으로 줄여봅시다.

```
ALTER TABLE student
CHANGE student_id student_id INT NOT NULL AUTO_INCREMENT,
CHANGE name name VARCHAR(10) NOT NULL;
```

속성의 정보를 변경하려면 CHANGE 키워드를 사용하면 되는데 이때 반드시 기존에 옵션으로 주었던 것을 다시 추가해야 합니다. 즉, 수정하고 싶은 속성을 변경한다기보다는 해당 속성을 새롭게 정의한다고 생각하는 것이 낫습니다. 여기서 속성의 이름을 두 번에 걸쳐서 써주는 것은 처음 이름을

두 번째 이름으로 변경한다는 의미입니다. 따라서 이름을 변경하지 않을 때에는 두 번에 걸쳐서 같은 이름을 써주어야 합니다.

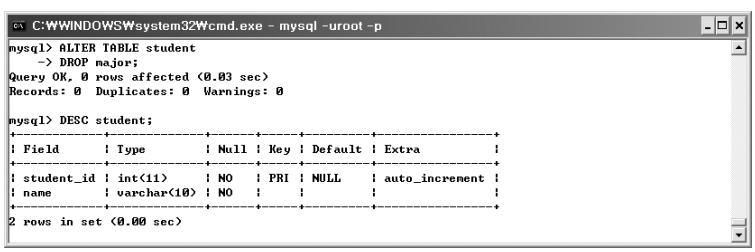


[그림 7-36] 테이블의 필드 수정

이제 마지막으로 테이블에서 속성을 삭제해 보겠습니다.

```
ALTER TABLE student
DROP major;
```

DROP 키워드를 이용하여 해당 속성을 삭제할 수 있습니다.

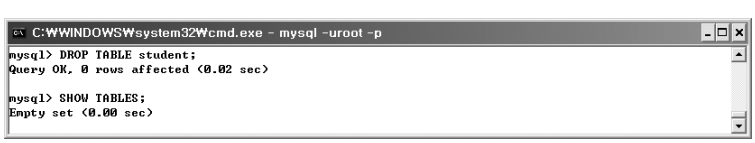


[그림 7-37] 테이블의 필드 제거

테이블 삭제

테이블을 삭제하려면 데이터베이스와 마찬가지로 DROP 문을 사용하면 됩니다. 테이블 역시 별도의 확인 없이 곧바로 테이블을 삭제하므로 실수로 테이블을 지워버리는 일이 없도록 주의를 해야 합니다.

```
DROP TABLE student;
```



[그림 7-38] 테이블의 삭제

여기서 SHOW TABLES는 선택된 데이터베이스 내에 존재하는 테이블의 목록을 보여주는 명령어입니다.

데이터 삽입

테이블에 데이터를 삽입하려면 INSERT 문을 사용하면 됩니다. 테이블에 레코드를 삽입하는 방법은 여러 가지가 있는데 다른 데이터베이스에도 사용할 수 있는 표준 SQL 방법과 MySQL에서만 사용할 수 있는 독특한 방법이 있습니다.

앞으로 데이터를 삽입하고 수정하고 삭제하는 것을 배우기 위해서 먼저 테이블을 정의하여 생성하겠습니다.

필 드	필드 이름	데이터형	크기	비 고
학생번호	student_id	INT		학생 테이블의 키
이름	Name	VARCHAR	10	
학년	Grade	TYNYINT		
학과번호	dept_no	INT		학과 테이블의 키

[표 7-6] 학생 테이블(student)

필 드	필드 이름	데이터형	크기	비 고
학과번호	dept_no	INT		학과 테이블의 키
학과명	dept_name	VARCHAR	20	
사무실	office	VARCHAR	20	
전화번호	office_tel	VARCHAR	13	

[표 7-7] 학과 테이블(department)

테이블을 정의하였으니 테이블 생성문을 작성해 봅시다.

```
CREATE TABLE student
 ( student_id  INT     NOT NULL,
   name      VARCHAR(10)      NOT NULL,
   grade     TINYINT          NOT NULL DEFAULT 1,
   dept_no   INT              NOT NULL,
   major     VARCHAR(20),
   PRIMARY KEY(student_id));

CREATE TABLE department
 ( dept_no         INT     NOT NULL AUTO_INCREMENT,
   dept_name         VARCHAR(20)      NOT NULL,
```

```
office          VARCHAR(20)     NOT NULL,
office_tel      VARCHAR(13),
PRIMARY KEY(dept_no));
```

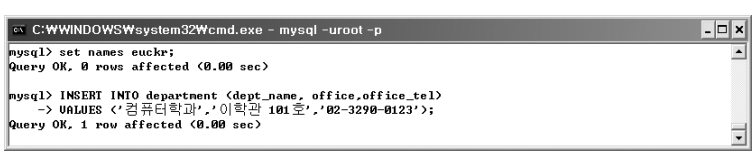
준비가 다 되었으니 표준 SQL 방식의 INSERT 문을 먼저 알아보겠습니다.

```
INSERT INTO 테이블이름 (필드1, 필드2, 필드3, … )
VALUES (필드1의 값, 필드2의 값, 필드3의 값, … );
```

이 방법은 가장 일반적인 방법으로 입력할 필드를 순서대로 기입하고 그 값을 또한 순서에 맞게 입력하는 방법입니다. 이 방법의 장점은 DEFAULT로 설정된 필드나 NULL 값을 허용하는 필드는 생략하여 입력하는 것이 가능하다는 것입니다. 예를 들어 모든 신입생이 학부로 입학해서 2학년에 진입하면서 전공이 결정되는 경우를 생각해 봅시다. 학생 정보 테이블에는 전공 필드가 존재하지만 레코드가 삽입되는 시점(신입생이 입학하는 시점)에서는 전공이 존재하지 않기 때문에 전공을 입력할 필요가 없습니다. 이런 경우에 필수 항목만을 명시하고 그 값을 입력할 수 있는 것입니다.

```
INSERT INTO department (dept_name, office,office_tel)
VALUES ('컴퓨터학과','이학관 101호','02-3290-0123');
```

학과 필드 중에서 학과 번호는 시스템이 자동으로 1씩 증가하는 값을 입력해주는 키값입니다. 따라서 사용자가 입력하지 않아도 되는 값이므로 해당 필드를 생략할 수 있습니다. 그리고 이에 앞서 이 값에는 한글이 포함되므로 set names euckr; 명령을 먼저 수행하여 입력 문자셋을 맞추어줍니다. 만약 기본 문자셋이 euckr이 아닌 상태에서 한글을 입력하게 되면 한글이 모두 깨져서 알아볼 수 없는 형태로 저장됩니다.



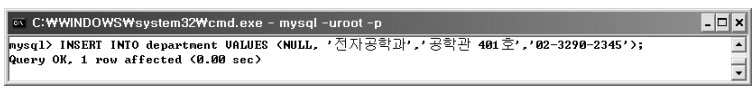
[그림 7-39] 필드 이름을 명시하여 레코드를 추가하는 방법

```
INSERT INTO 테이블이름 VALUES (필드1의 값, 필드2의 값, 필드3의 값, … );
```

이와 더불어 만약 테이블을 생성할 때의 필드 순서를 알고 있다면 필드의 이름은 생략할 수 있습니다. 단, 필드 이름을 명기하지 않기 때문에 특정 필드를 생략할 수 없으며 모든 필드에 대한 값을 입력해야 합니다.

```
INSERT INTO department VALUES
(NULL, '전자공학과','공학관 401호','02-3290- 2345');
```

학과 번호인 dept_no 항목은 1씩 증가하는 항목이므로 특별한 경우가 아닌 이상 시스템이 자동으로 입력하도록 해야 합니다. 그래서 해당 항목은 입력할 필요가 없지만 필드 이름을 명시하지 않았기 때문에 해당 항목을 NULL로 입력합니다. 그러나 추후 테이블의 설계가 변경되어 필드가 추가되거나 삭제되면 필드의 순서가 바뀌어 오류가 발생할 수 있기 때문에 되도록이면 필드 이름을 명시하는 것이 좋습니다.



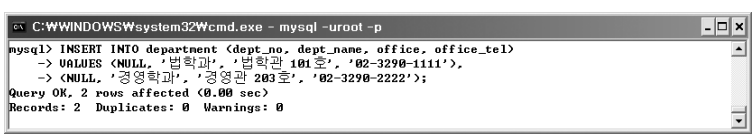
[그림 7-40] 필드 이름을 명시하지 않는 레코드를 추가하는 방법

만약 한 번의 쿼리로 여러 개의 데이터를 추가하고자 한다면 다음과 같은 방법을 이용할 수 있습니다.

```
INSERT INTO 테이블이름 (컬럼1, 컬럼2, 컬럼3, …)
VALUES (컬럼1의 값, 컬럼2의 값, 컬럼3의 값, …),
(컬럼1의 값2, 컬럼2의 값2, 컬럼3의 값2, …);
```

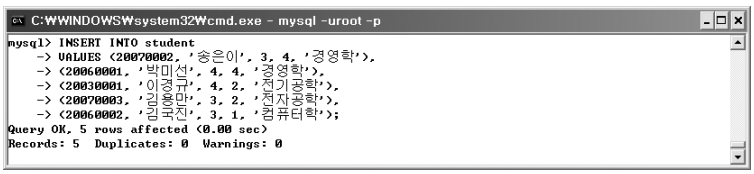
이 방법은 VALUES 항목을 콤마를 이용하여 확장하는 방법으로 여러 개의 레코드를 추가해야 할 경우 각 레코드당 하나의 INSERT 쿼리문을 작성해야 하는 불편함을 없애는 방법입니다. 이 방법 역시 컬럼의 순서를 알고 있다면 컬럼의 이름을 생략할 수 있습니다.

```
INSERT INTO department (dept_no, dept_name, office, office_tel)
VALUES (NULL, '법학과', '법학관 101호', '02-3290-1111'),
       (NULL, '경영학과', '경영관 203호', '02-3290-2222');
```



[그림 7-41] 멀티 레코드 삽입 방법

```
INSERT INTO student
VALUES (20070002, '송은이', 3, 4, '경영학'),
       (20060001, '박미선', 4, 4, '경영학'),
       (20030001, '이경규', 4, 2, '전기공학'),
       (20070003, '김용만', 3, 2, '전자공학'),
       (20060002, '김국진', 3, 1, '컴퓨터학');
```



[그림 7-42] 멀티 레코드 삽입 방법

마지막으로 표준 SQL이 아니라 MySQL에서만 제공하는 특이한 형태의 INSERT 문에 대해서 알아보겠습니다.

```
INSERT INTO 테이블이름 SET 필드1 = 필드1의 값, 필드2 = 필드2의 값 … ;
```

이 방법은 추후 배우게 될 UPDATE 문의 형식에서 따온 것으로 삽입과 수정 시에 거의 같은 SQL 문을 사용할 수 있다는 것과 필드 이름은 이름대로 필드의 값은 값대로 따로 분류하여 입력하는 데서 발생할 수 있는 오류를 해결할 수 있다는 장점이 있습니다. 그러나 표준 SQL 문이 아니므로 다른 데이터베이스를 사용하게 될 경우 SQL 문을 다시 작성하여야 하는 단점이 있습니다.

```
INSERT INTO student SET student_id=20080001, name='강호동', dept_no=1;
INSERT INTO student SET student_id=20080002, name='유재석', dept_no=2,
                                    grade=1;
INSERT INTO student SET student_id=20070001, name='윤종신', dept_no=3,
                                    grade=2, major='법학';
```

[그림 7-43] UPDATE 형식으로 레코드를 추가하는 방법

여기서 주의할 점은 학생 정보를 입력할 때 학과 번호는 학과 테이블의 키이기 때문에 반드시 학과 테이블에 존재하는 값을 입력해야 합니다.

데이터 수정

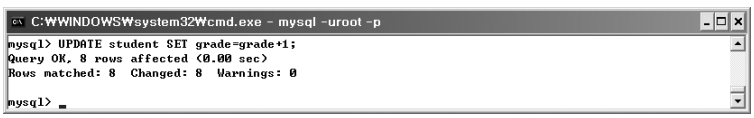
테이블에 삽입된 데이터를 수정하려면 UPDATE 문을 사용합니다.

```
UPDATE 테이블이름 SET 컬럼1=컬럼1의 값, 컬럼2=컬럼2의 값 … WHERE 조건식;
```

새 학년이 되어 학생들이 한 학년씩 진급을 하는 경우를 처리해 봅시다.

```
UPDATE student SET grade=grade+1;
```

위의 UPDATE 문에서 보이듯이 해당 필드의 현재 값을 참조하여 계산과정을 거쳐 데이터를 수정할 수도 있습니다.

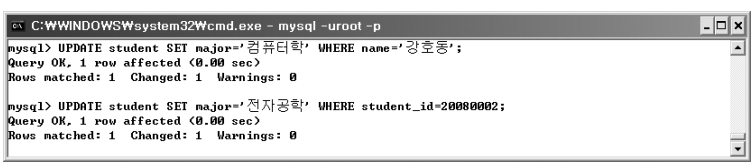


[그림 7-44] 모든 레코드 수정

모든 학생이 한 학년씩 진급을 했기 때문에 4학년인 학생은 5학년이 될 것입니다. 5학년인 사람들은 잠시 후에 졸업시키기로 하고 1학년에서 2학년으로 진급한 학생들은 전공이 결정되었으니 전공 항목을 수정해줍니다.

```
UPDATE student SET major='컴퓨터학' WHERE name='강호동';
UPDATE student SET major='전자공학' WHERE student_id=20080002;
```

WHERE 조건식은 모든 레코드를 수정하는 것이 아니라 특정 레코드를 선택하여 수정하고자 할 경우에 사용합니다. 위의 쿼리문에서 보이듯이 다른 필드의 값을 통해서 레코드를 선택하고 이를 수정합니다. 그런데 강호동의 경우는 동명이인인 탓에 여러 개의 레코드가 변경될 여지가 있으므로 학생 테이블의 키가 되는 학생번호를 이용하여 수정하는 것이 바람직합니다. WHERE 절은 SELECT 문에서 자세히 다루도록 하겠습니다.



[그림 7-45] 특정 레코드의 수정

데이터 삭제

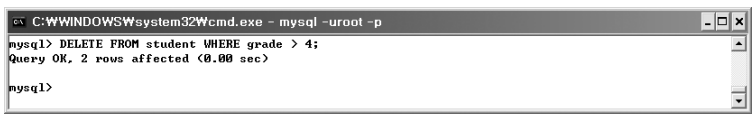
저장된 데이터를 삭제하려면 DELETE 문을 사용합니다.

```
DELETE FROM 테이블이름 WHERE 조건식;
```

새 학년이 되어 졸업 하는 학생들을 테이블에서 삭제해 보도록 합시다.

```
DELETE FROM student WHERE grade > 4;
```

앞서 학생들이 진급하면서 졸업반인 4학년의 경우 5학년으로 변경되었습니다. 이를 이용하여 학년이 4보다 큰 학생들을 테이블에서 삭제합니다.



[그림 7-46] 레코드의 삭제

DELETE 문은 조건식을 사용하지 않는 경우 테이블에 존재하는 모든 레코드를 삭제합니다. 이는 초보들이 자주 하는 실수로 중요한 데이터를 모두 잃어버릴 수 있으므로 DELETE 문을 작성할 때 반드시 조건절이 맞는지 확인하는 것을 잊지 말아야 합니다.

데이터 검색

데이터를 검색하려면 SELECT 문을 사용합니다. 데이터베이스에 저장된 수많은 데이터에서 필요한 정보를 검색하는 이 기능이야말로 데이터베이스의 꽃이라고 할 수 있기 때문에 앞서 배운 SQL 문보다 다양한 기능을 제공합니다.

SELECT 문의 기본 모형은 다음과 같습니다.

```
SELECT 필드1, 필드2, …
FROM 테이블이름
WHERE 절
GROUP BY 절
HAVING 절
ORDER BY 절
LIMIT 절
```

기본 모형이 매우 복잡해 보이고 어려워 보이지만 대부분이 반드시 필요한 항목이 아니라 필요에 따라 추가되는 항목이므로 두려워할 필요가 없습니다. 각 절에 대해서 정리해보면 다음과 같습니다.

절	기능
SELECT	검색할 데이터의 필드를 지정

FROM	데이터를 검색할 테이블을 지정, 여러 개의 테이블인 경우 콤마로 구분
WHERE	데이터의 검색 조건을 지정
GROUP BY	데이터를 어떤 필드를 기준으로 그룹화할지 지정
HAVING	GROUP BY 절에서 그룹화하는 조건을 지정
ORDER BY	검색된 결과를 정렬하는 방법을 지정
LIMIT	검색된 결과의 레코드 수를 제한

[표 7-8] SELECT 문의 구성

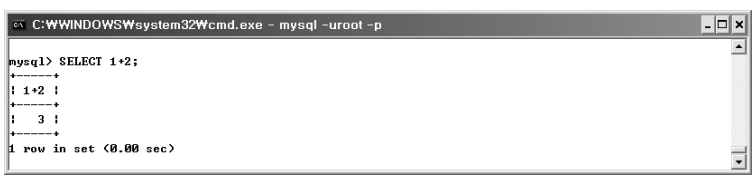
각 절이 어떻게 사용되는지 예제를 통해서 알아보겠습니다.

단순 질의

가장 단순한 형태의 SELECT 문을 작성해 보겠습니다.

```
SELECT 1+2;
```

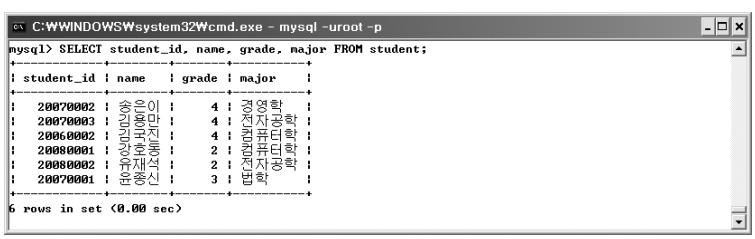
SELECT는 반드시 테이블의 데이터 검색뿐만이 아니라 이처럼 수식 계산도 가능합니다. 이를 통해서 현재의 시간을 구한다거나 수학 내장 함수로 보다 복잡한 수식을 계산할 수도 있습니다.



[그림 7-47] SELECT를 이용한 수식 계산

이제 테이블에 존재하는 레코드를 조회해 보겠습니다.

```
SELECT student_id, name, grade, major FROM student;
```



[그림 7-48] 테이블의 레코드 조회

필드는 테이블이 생성될 때의 순서가 아닌 SELECT 문에 입력한 순서로 출력됩니다. 또한 사용자가 원하는 필드만을 출력되게 할 수 있으며 항목을 선택하지 않고 모든 항목을 보고 싶다면 * 기호

를 사용하면 됩니다.

```
SELECT * FROM student;
```

[그림 7-49] 테이블 전체 필드 검색

WHERE 절

이제 조건절을 추가하여 특정 레코드를 검색해 보겠습니다. WHERE 절에서 사용할 수 있는 검색 연산자는 다음과 같습니다.

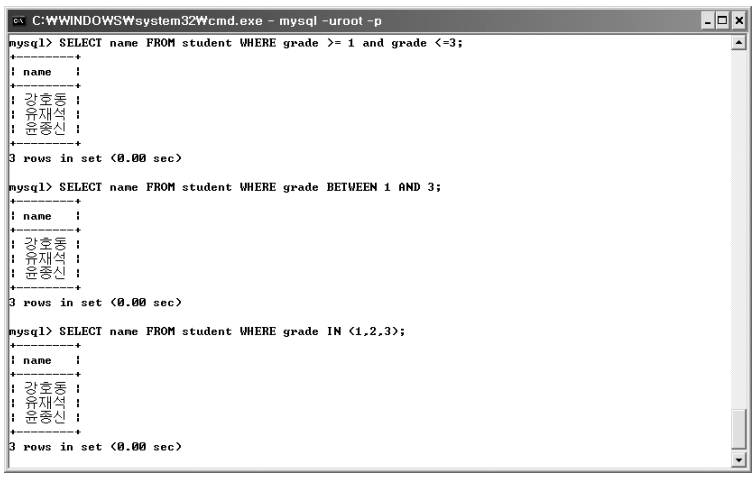
종류	연산자	설명
비교	=, 〈, 〉, 〈=, 〉=, !=, 〈〉	두 값을 비교
논리	AND, OR, NOT	조건과 조건을 결합하거나 참/거짓의 판단
범위	BETWEEN a AND b	a와 b 사이에 존재하는지 검사
집합	IN	해당 값이 목록에 존재하는지 검사
패턴	LIKE	문자열의 패턴 검사

[표 7-9] 검색에 사용되는 연산자

위의 연산자를 이용하여 학년이 1,2,3학년인 학생들의 이름을 검색하는 쿼리문을 작성해보면 다음과 같이 다양하게 만들 수 있습니다.

```
SELECT name FROM student WHERE grade >= 1 and grade <=3;
SELECT name FROM student WHERE grade BETWEEN 1 AND 3;
SELECT name FROM student WHERE grade IN (1,2,3);
```

이 세 가지 쿼리는 모두 같은 역할을 하며 결과는 같습니다.



[그림 7-50] 조건을 이용한 검색

또한 문자열을 검색하고자 할 때는 다음과 같이 LIKE 절을 사용하면 됩니다.

```
SELECT name FROM student WHERE name LIKE '김%';
```

[그림 7-51] 문자열 패턴을 이용한 검색

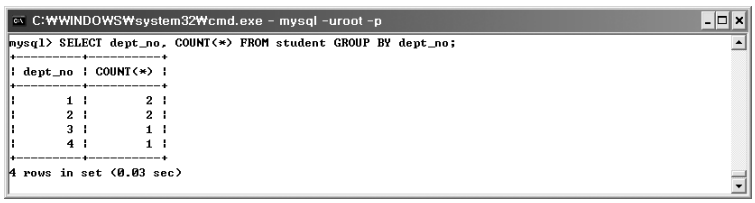
LIKE '김%'는 '김'으로 시작하는 이름을 가진 사람을 검색하는 방법입니다. 이와 마찬가지로 마지막에 '동'자로 끝나는 이름을 검색하고 싶다면 LIKE '%동', '용'이라는 글자를 포함하는 이름을 검색하고 싶다면 LIKE '%용%'라고 하면 됩니다.

GROUP BY 절

레코드를 그룹화하여 결과를 얻고 싶을 때는 GROUB BY 절을 이용하면 됩니다. 만약 학과별 재학생의 수를 알고 싶은 경우를 살펴봅시다.

```
SELECT dept_no, COUNT(*) FROM student GROUP BY dept_no;
```

여기서 COUNT(*)는 검색된 레코드의 수를 계산하는 내장 함수입니다. 이러한 대표적인 내장 함수는 합을 구하는 SUM, 평균을 구하는 AVG, 최소값과 최대값을 구하는 MIN/MAX가 있습니다. 예를 들어 SUM(grade)라고 한다면 검색된 학생들 학년의 총합이 결과로 출력됩니다.



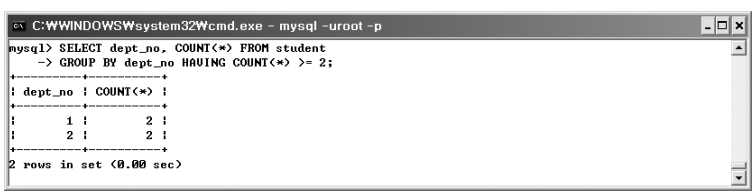
[그림 7-52] 그룹화를 통한 검색

GROUP BY 절은 이처럼 테이블에 존재하는 레코드를 특정 필드를 기준으로 같은 값끼리 그룹화를 합니다. COUNT(*) 함수는 그룹마다 결과를 나타내며 GROUP BY에 지정하지 않은 필드값을 SELECT 절에 추가하는 경우 에러나 잘못된 결과를 얻을 수 있습니다.

HAVING 절

HAVING 절은 GROUP BY 절을 이용하여 레코드를 그룹화하고 여기에 조건을 더 부여하여 그룹화된 결과 중 특정 결과만을 얻고자 할 때 사용합니다. 예를 들면 학과별 재학생 수를 구하되 학생의 수가 2명 이상인 경우를 구하고자 한다면 HAVING 절을 사용하면 됩니다.

```
SELECT dept_no, COUNT(*) FROM student
GROUP BY dept_no HAVING COUNT(*) >= 2;
```

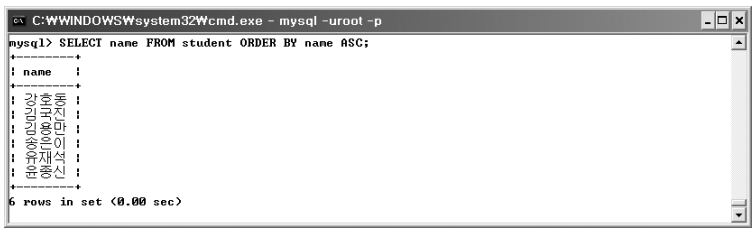


[그림 7-53] HAVING 절을 이용한 GROUP BY 조건 검색

ORDER BY 절

검색된 레코드를 정렬할 때 ORDER BY 절을 사용합니다. ORDER BY 절을 이용하여 오름차순(ASC) 정렬과 내림차순(DESC) 정렬이 가능합니다. 전체 학생을 가나다순으로 정렬하고자 한다면 다음과 같이 쿼리를 작성하면 됩니다.

```
SELECT name FROM student ORDER BY name ASC;
```



[그림 7-54] 레코드의 정렬

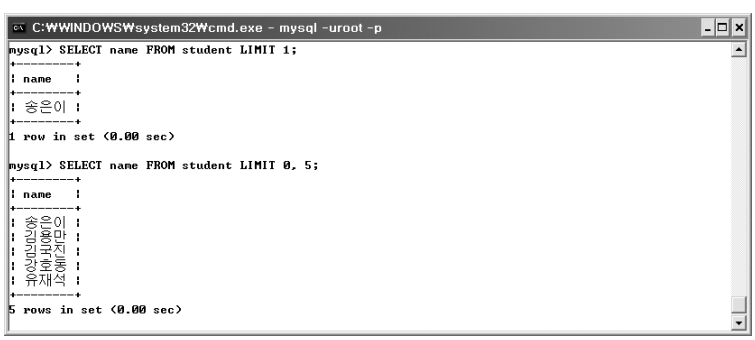
오름차순 정렬의 경우는 기본 정렬 방식이 오름차순이므로 특별히 ASC라고 명시를 하지 않아도 됩니다.

LIMIT 절

LIMIT 절은 검색된 레코드의 수를 제한하는 데 사용합니다. 조건에 의해 검색된 수천 개의 레코드 중에서 특정 위치의 레코드나 특정 개수의 단위로 가져오고 싶을 때 LIMIT를 이용할 수 있습니다.

```
SELECT name FROM student LIMIT 1;
SELECT name FROM student LIMIT 0, 5;
```

위와 같은 방식으로 사용할 수 있으며 LIMIT 1은 첫 번째 레코드만 가져오겠다는 것이며 LIMIT 0, 5는 가장 첫 번째(0번부터 시작) 레코드부터 5개를 가져오겠다는 의미입니다.



[그림 7-55] LIMIT를 이용한 레코드 수의 제한

조인(JOIN)

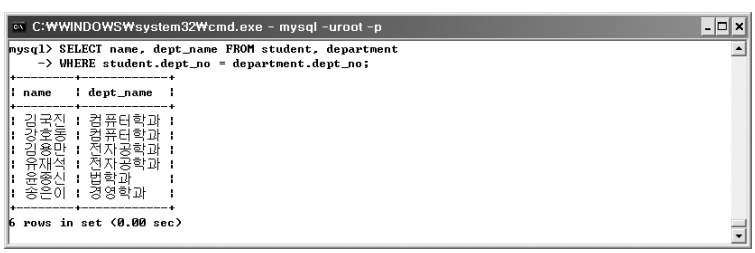
정규화를 통해서 분리된 테이블을 필요에 따라 합칠 때 사용하는 것이 조인(JOIN)입니다. 정규화를 통해서 학생 정보와 학과 정보를 분리했었습니다. 그래서 학생의 소속 학과명을 알려면 학생 테이블의 학과번호를 통해서 학과 테이블을 검색해야 합니다.

```
SELECT name, dept_name FROM student, department
```

```
  WHERE student.dept_no = department.dept_no;
SELECT S.name, D.dept_name FROM student S, department D
  WHERE S.dept_no = D.dept_no;
SELECT S.name, D.dept_name FROM student AS S, department AS D
  WHERE S.dept_no = D.dept_no;
SELECT S.name, D.dept_name FROM student AS S INNER JOIN department AS D
  ON S.dept_no = D.dept_no;
```

여기서 S와 D는 각각 닉네임(Alias)을 설정한 것으로 테이블 이름이나 필드 이름에 별명을 부여할 수 있습니다. 닉네임은 필드 이름이나 테이블 이름 뒤에 스페이스를 두고 지정하거나 특별히 AS라는 닉네임 키워드를 통해서 지정하는 것이 가능합니다. 테이블 이름을 닉네임으로 정하게 되면 해당 테이블의 필드를 표기하기 위해서 테이블이름.필드이름 혹은 닉네임.필드이름 방식을 사용할 수 있습니다.

위의 네 가지 쿼리문은 실질적으로 같으며 WHERE 절에 두 테이블을 합치는 조건을 부여하여 학과번호가 같은 레코드끼리 합치고 있습니다. 이는 두 테이블 레코드의 조합 중에서 조건에 일치하는 레코드만을 가져오는 방법입니다. 이러한 조인을 동등 조인(Equi Join)이라고 합니다. 또한 동등 조인 중에서 두 테이블의 공통 필드인 dept_no를 중복으로 표현하지 않고 하나만 표현할 때 이를 자연 조인(Natural Join)이라고 합니다. 그리고 동등 조인과 자연 조인 등을 통칭하여 내부 조인(Inner Join)이라고 합니다.



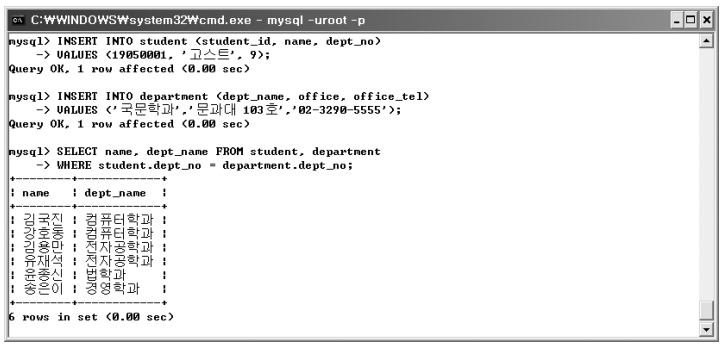
[그림 7-56] 내부 조인을 이용한 테이블의 병합

여기에 학과 테이블에 없는 학과를 다니는 유령 학생과 새로운 학과를 추가해 보겠습니다.

```
INSERT INTO student (student_id, name, dept_no)
VALUES (19050001, '고스트', 9);
INSERT INTO department (dept_name, office, office_tel)
VALUES ('국문학과','문과대 103호','02-3290-5555');
```

두 레코드는 고스트 학생은 학과가 존재하지 않고 국문학과는 학생이 존재하지 않습니다. 이런 경우 내부 조인을 하게 되면 어떻게 될까요?

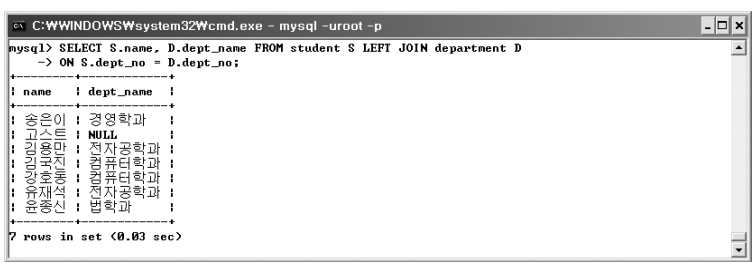
내부 조인은 결합된 레코드의 교집합입니다. 결합 조건을 만족하지 않으면 레코드는 결합하지 않고 무시되어 버립니다. 그래서 고스트 학생과 국문학과는 검색 결과에 나타나지 않게 됩니다.



[그림 7-57] 내부 조인은 테이블의 교집합

내부 조인의 이러한 특징 때문에 고스트와 국문학과는 마치 존재하지 않는 레코드인 것처럼 보입니다. 이것을 방지하고자 외부 조인(Outter Join)이 생겼습니다. 외부 조인에는 내부 조인과 마찬가지로 LEFT JOIN, RIGHT JOIN, CROSS JOIN이 있습니다.

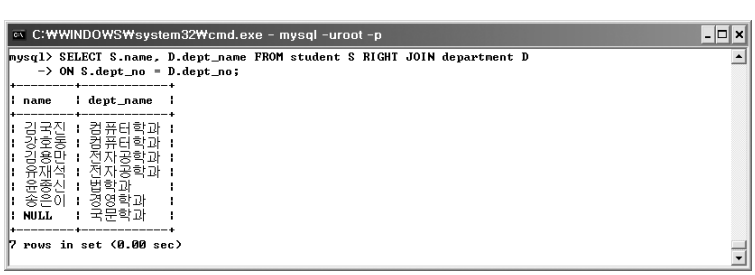
```
SELECT S.name, D.dept_name FROM student S LEFT JOIN department D
ON S.dept_no = D.dept_no;
```



[그림 7-58] LEFT JOIN의 결과

LEFT JOIN은 결과에서 보이듯이 내부 조인의 결과에 왼쪽 테이블에서 남아있는 레코드 즉, 조인 조건에 일치하지 않는 왼쪽 테이블의 레코드를 추가하고 채우지 못하는 정보는 모두 NULL로 대체하는 방법입니다. 학생 테이블에 학과 정보를 추가하는데 고스트 학생은 일치하는 학과가 없으므로 NULL로 대체된 것입니다.

RIGHT JOIN은 LEFT JOIN과 반대로 오른쪽 테이블에 남아있는 레코드를 처리하는 방식입니다.



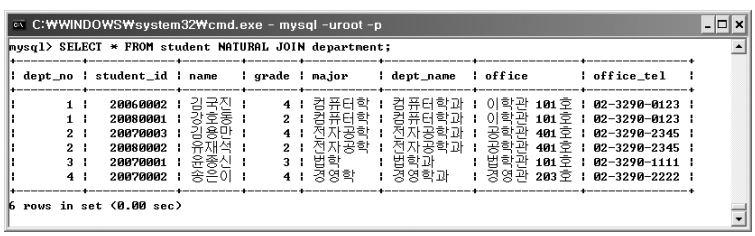
[그림 7-59] RIGHT JOIN의 결과

RIGHT JOIN의 결과를 보면 내부 조인의 결과에 오른쪽 테이블에서 남겨진 국문학과 레코드가 추가된 것을 알 수 있습니다. 또한 해당 학과에는 학생이 없으므로 NULL로 대체되었습니다.

마지막으로 CROSS JOIN은 MySQL에서는 내부 조인과 동일하게 동작하며 조인 조건절 ON을 사용하지 않을 때를 CROSS JOIN이라고 합니다. CROSS JOIN은 두 테이블의 레코드의 수를 각각 N, M이라고 하면 총 N × M개의 레코드가 생성됩니다.

MySQL에는 앞서 잠깐 언급한 자연 조인 명령이 있습니다. NATURAL이라는 키워드를 사용하면 조인 조건을 부여하지 않더라도 같은 필드를 찾아서 동일 필드를 하나 제거한 상태의 결과를 출력합니다. 이는 단순히 정규화를 통해서 분리된 테이블을 병합하고자 할 때 매우 유용하게 사용할 수 있습니다.

```
SELECT * FROM student NATURAL JOIN department;
SELECT * FROM student NATURAL LEFT JOIN department;
SELECT * FROM student NATURAL RIGHT JOIN department;
```



[그림 7-60] 자연 조인의 결과

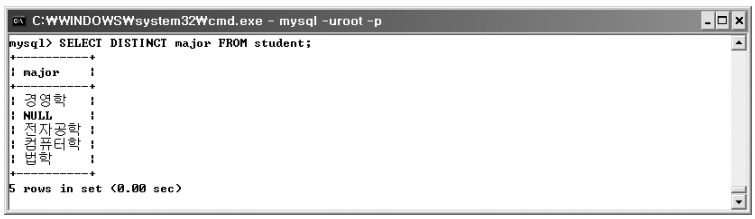
정규화가 진행되면 될수록 자연스럽게 조인의 수는 늘어납니다. 그러나 조인은 쿼리 중에서 가장 부하가 많이 걸리는 작업이기 때문에 너무 잦은 조인은 성능 저하의 원인이 됩니다. 특히 고도의 정규화로 테이블 3, 4개를 조인하게 되는 경우가 잦아진다면 때로는 한 페이지의 결과를 도출하는데 몇 초에서 몇 십 초의 시간이 걸릴 수도 있습니다.

성능 향상을 위해서 고도의 정규화를 일부러 하지 않은 이유가 바로 여기에 있습니다. 고도의 정규화는 곧 테이블의 분할이기 때문에 적당히 정규화를 하여 테이블이 너무 잘게 쪼개지는 것을 막는 것입니다. 만약 테이블을 분할하고 나서 조인으로 인해서 성능 저하가 발생한다면 부득이하게 역정규화를 통해서 테이블을 합치는 것을 고려해야 합니다.

중복된 레코드 제거

전체 학생들 전공의 종류를 파악하기 위해서 학생들의 전공을 출력하고자 한다면 "SELECT major FROM student;"와 같은 방법으로 출력할 수 있을 것입니다. 그러나 이렇게 쿼리를 작성하면 모든 학생의 전공이 출력되므로 반복해서 특정 전공 정보가 나오게 될 것이 분명합니다. 이러한 중복을 없애주는 키워드가 DISTINCT입니다.

```
SELECT DISTINCT major FROM student;
```



[그림 7-61] 중복된 레코드 제거

UNION

UNION은 여러 개의 SELECT 결과를 하나로 합치기 위한 키워드입니다.

테스트를 위해서 지도 교수의 관계를 무시하고 학생 테이블을 이용하여 교수 테이블을 간단하게 작성해 봅시다. 학생과 교수의 차이점은 교수는 학생번호 대신에 교수번호가 있고 학년 대신에 직위가 있다는 것입니다. 따라서 교수 테이블은 다음과 같이 만들 수 있습니다.

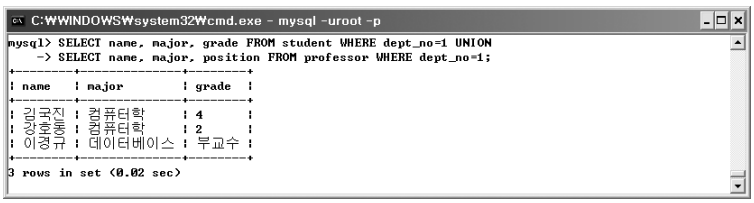
```
CREATE TABLE professor
( professor_id INT      NOT NULL,
  name       VARCHAR(10)    NOT NULL,
  position     VARCHAR(10)  NOT NULL DEFAULT '조교수',
  dept_no    INT          NOT NULL,
  major      VARCHAR(20),
PRIMARY KEY(professor_id));
```

이 테이블에 교수를 몇 명 등록합니다.

```
INSERT INTO professor (professor_id, name, position, dept_no, major)
       VALUES (1000, '이경규', '부교수', 1, '데이터베이스'),
              (2000, '박미선', '조교수', 2, '마이크로아키텍처'),
              (3000, '최양락', '교수', 3, '민사법'),
              (4000, '이봉원', '교수', 4, ' 인적자원관리');
```

이제 다시 UNION으로 돌아와서 컴퓨터학과 학생과 교수의 목록을 출력하고자 한다면 학생목록 한 번과 교수 목록 한 번 모두 두 번에 걸쳐서 쿼리를 실행해야 합니다. 그러나 UNION을 사용하면 한 번의 쿼리만으로 이를 출력할 수 있습니다.

```
SELECT name, major, grade FROM student WHERE dept_no=1 UNION
SELECT name, major, position FROM professor WHERE dept_no=1;
```



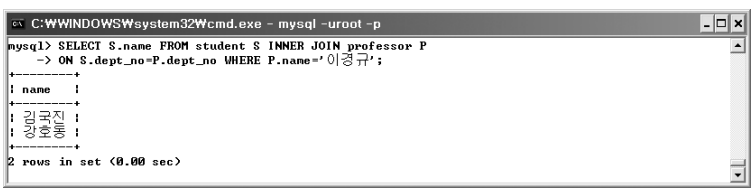
[그림 7-62] UNION을 이용한 레코드의 결합

서브 쿼리(Sub Query)

MySQL은 서브 쿼리를 지원하기 때문에 쿼리 속에 쿼리를 작성할 수 있습니다. 서브 쿼리를 사용하면 불필요한 조인을 많이 제거할 수 있기 때문에 매우 유용한 기능입니다.

이경규 교수가 소속되어 있는 학과의 학생들 수를 한 번의 쿼리를 통해서 구해봅시다.

```
SELECT S.name FROM student S INNER JOIN professor P
     ON S.dept_no=P.dept_no WHERE P.name='이경규';
```

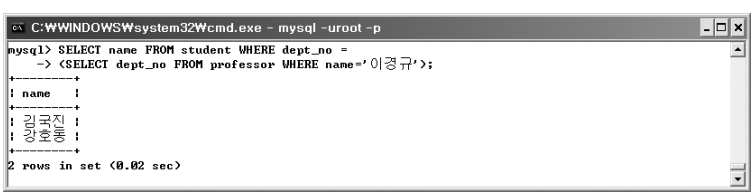


[그림 7-63] 조인을 이용한 결과

물론 교수 테이블에서 이경규의 소속학과 정보를 찾고 다시 학생 테이블에서 해당 학과의 학생들을 검색하면 됩니다. 그러나 한 번의 쿼리를 이용하여 결과를 출력하려면 조인이나 서브 쿼리를 사용해야 합니다.

조인을 이용한 쿼리문은 서브 쿼리를 이용하여 다음과 같이 변경할 수 있습니다.

```
SELECT name FROM student WHERE dept_no =
(SELECT dept_no FROM professor WHERE name='이경규');
```

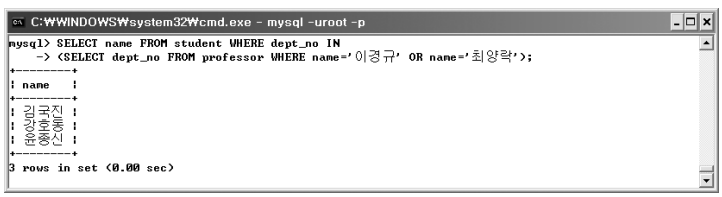


[그림 7-64] 서브 쿼리를 이용한 결과

만약 이경규 교수뿐만이 아니라 최양락 교수의 학생 정보를 알고 싶다면 다음과 같이 수정해야 합

니다. 서브 쿼리 앞에 '=' 기호를 사용하는 경우 같다는 의미이기 때문에 여러 개의 레코드를 가진 서브 쿼리의 경우에는 이를 처리할 수가 없습니다. 그래서 앞에서 배운 IN 키워드를 이용하여 여러 개의 레코드를 갖는 서브 쿼리를 처리할 수 있습니다.

```
SELECT name FROM student WHERE dept_no IN
(SELECT dept_no FROM professor WHERE name='이경규' OR name='최양락');
```



[그림 7-65] 서브 쿼리가 여러 개의 레코드를 갖는 경우

권한 부여

데이터베이스에 대한 접근 및 관리를 위한 권한을 부여하려면 GRANT 문을 사용합니다. GRANT 문을 이용하면 데이터베이스 사용자를 추가하거나 권한을 변경할 수 있습니다.

GRANT 문의 기본 모형은 다음과 같습니다.

```
GRANT 권한 ON 권한레벨 TO 사용자
IDENTIFIED BY '패스워드' WITH GRANT OPTION;
```

IDENTIFIED BY 이후는 필요에 따라 생략할 수 있으며 사용자를 추가하거나 비밀번호를 변경하고자 할 때 사용합니다.

부여할 수 있는 권한을 정리하면 다음과 같습니다.

권한	설명
ALL	GRANT OPTION을 제외한 모든 권한
ALTER	ALTER 명령 허용
CREATE	CREATE 명령 허용
DELETE	DELETE 명령 허용
DROP	DROP 명령 허용
INDEX	CREATE INDEX, DROP INDEX 명령 허용
INSERT	INSERT 명령 허용
SELECT	SELECT 명령 허용

UPDATE	UPDATE 명령 허용
USAGE	권한 없음

[표 7-10] GRANT 권한

권한 레벨은 글로벌 레벨, 데이터베이스 레벨, 테이블 레벨 등 총 5가지가 있습니다. 글로벌 레벨은 모든 데이터베이스와 모든 테이블에 대한 권한을 의미하고 데이터베이스 레벨은 특정 데이터베이스의 모든 테이블에 대한 권한을 의미하며 테이블 레벨은 특정 테이블에 대한 권한을 의미합니다.

글로벌 레벨 : *.*
데이터베이스 레벨 : 데이터베이스이름.*
테이블 레벨 : 데이터베이스이름.테이블이름

다음과 같은 쿼리를 실행함으로써 brown 사용자에게 로컬로 접속하는 경우 GRANT 권한을 제외한 모든 권한을 부여할 수 있습니다. 만약 brown 계정이 존재하지 않는다면 해당 계정이 추가됩니다.

```
GRANT ALL ON *.* TO brown@localhost;
```

GRANT 권한이 없으면 GRANT 문을 사용할 수가 없습니다. 만약 GRANT 권한까지 부여하고자 한다면 다음과 같은 쿼리를 사용하면 됩니다.

```
GRANT ALL ON *.* TO brown@localhost WITH GRANT OPTION;
```

또한 로컬로의 접속뿐만 아니라 모든 곳에서도 접속할 수 있게 하려면 localhost 대신에 '%'를 추가하면 됩니다.

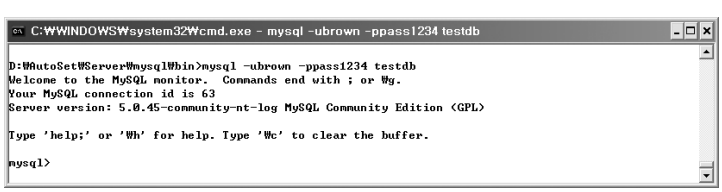
```
GRANT ALL ON *.* TO brown@'%' WITH GRANT OPTION;
```

그리고 brown 계정의 비밀번호를 등록하려면 IDENTIFIED BY 항목을 이용하면 됩니다.

```
GRANT ALL ON *.* TO brown@'%'
IDENTIFIED BY 'pass1234' WITH GRANT OPTION;
```

위의 쿼리는 데이터베이스 사용자에 brown을 추가시켜 모든 곳에서 접속할 수 있으며 데이터베이스에 대한 모든 권한을 갖게 만듭니다.

사용자가 제대로 등록되었는지 확인하기 위해서 다음과 같이 데이터베이스에 접속해 봅시다.



[그림 7-66] 추가된 사용자 로그인 확인

권한 박탈

GRANT를 통해서 부여된 권한을 박탈하려면 REVOKE 문을 사용하면 됩니다. REVOKE 문은 GRANT 문과 유사하며 권한, 권한 레벨은 모두 같습니다.

REVOKE 문의 기본 형태는 다음과 같습니다.

```
REVOKE 권한 ON 권한레벨 FROM 사용자;
```

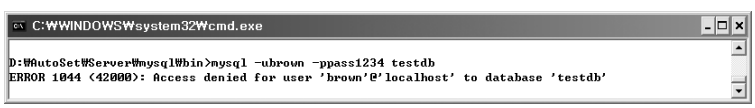
REVOKE 문을 이용하여 특정 권한을 박탈하고자 하는 경우 다음과 같은 쿼리를 사용할 수 있습니다.

```
REVOKE CREATE, DELETE ON *.* FROM brown@'%';
```

만약 모든 권한을 박탈하고자 한다면 ON 절을 생략하는 다음 쿼리를 사용합니다.

```
REVOKE ALL, GRANT OPTION FROM brown@'%';
REVOKE ALL, GRANT OPTION FROM @localhost;
```

brown 계정으로 다시 로그인을 해보면 모든 권한을 박탈당했으므로 더는 데이터베이스에 접속할 수 없음을 알 수 있습니다.



[그림 7-67] 사용자 권한 박탈

Section 04 MySQL 관련 함수

PHP에서 MySQL을 사용하는 방법은 크게 3가지가 있습니다. 첫째, 특정 데이터베이스와 무관한 ODBC(Open Database Connectivity)나 PDO(PHP Data Objects)와 같은 추상적인 함수를 사용하는 방법. 둘째, 초기 PHP 버전부터 사용되어 오던 MySQL 함수를 사용하는 방법. 마지막으로 MySQL 함수를 확장한 MySQLi(MySQL Improved Extension) 함수를 사용하는 것입니다.

추상 계층의 PDO를 사용하면 추후 데이터베이스가 변경된다고 하더라도 쿼리를 표준 SQL로 작성하였다면 소스 코드를 수정할 필요가 없는 장점이 있지만 데이터베이스를 직접 접근하는 것이

아니라 하나의 단계를 더 거쳐서 접근하기 때문에 속도가 저하되는 단점이 있습니다. 그에 반해 MySQL 함수는 속도 측면에서 추상 계층을 사용하는 것에 비해 월등히 빠르지만 MySQL 데이터베이스에 특화된 함수이기 때문에 다른 데이터베이스와 호환되지 않는 단점이 있습니다. 그리고 MySQL 4.1 이상의 버전에서만 사용할 수 있는 MySQLi의 경우, MySQL 함수와 성능적 측면에서는 거의 동등하거나 약간 우위를 차지하면서도 객체 형태의 인터페이스, 다중 쿼리, 미리 준비된 쿼리(Prepared Query)를 지원하는 등의 장점이 있습니다.

이 책에서는 MySQL 함수 중에서 널리 사용되는 함수를 정리하도록 하겠습니다.

MySQL 함수

mysql_connect

[PHP 4, PHP 5]

resource mysql_connect ([string server [, string username [, string password]]])

데이터베이스에 접속한다.

인자	자료형	설명	비고
server	string	MySQL 서버 주소	옵션
username	string	데이터베이스 사용자 계정	옵션
password	string	데이터베이스 사용자 비밀번호	옵션

[표 7-11] mysql_connect 함수

mysql_connect() 함수는 데이터베이스에 접속하고 연결이 되면 연결된 링크 식별자를 되돌려줍니다.

mysql_close

[PHP 4, PHP 5]

bool mysql_close ([resource link_identifier])

데이터베이스의 연결을 종료한다.

인자	자료형	설명	비고
link_identifier	resource	연결된 링크 식별자	옵션

[표 7-12] mysql_close 함수

mysql_close() 함수는 mysql_connect() 함수를 통해서 연결된 데이터베이스 접속을 종료합니다. 링크 식별자가 명시되지 않으면 최근에 접속한 링크를 종료합니다. mysql_connect() 함수를 이용한 연결은 PHP 스크립트의 마지막에서 자동으로 종료되므로 반드시 이 함수를 사용할 필요는 없습니다.

mysql_pconnect

[PHP 4, PHP 5]
resource mysql_pconnect ([string server [, string username [, string password]]])
데이터베이스에 지속적인 연결을 생성한다.

인자	자료형	설명	비고
server	string	MySQL 서버 주소	옵션
username	string	데이터베이스 사용자 계정	옵션
password	string	데이터베이스 사용자 비밀번호	옵션

[표 7-13] mysql_pconnect 함수

mysql_pconnect() 함수는 데이터베이스에 접속하기 전에 이미 연결된 링크가 존재하면 그것을 재사용하고 그렇지 않으면 새로운 접속을 통해서 새 링크 식별자를 되돌려줍니다. mysql_connect() 함수의 경우 PHP 스크립트 종료와 함께 데이터베이스 연결이 종료되지만 mysql_pconnect() 함수는 PHP 스크립트가 종료되어도 연결을 종료하지 않고 다음에 재사용할 수 있도록 남겨둡니다. 이렇게 지속적인 접속을 사용하는 경우는 너무나 잦은 데이터베이스 접속으로 인해서 부하가 생길 때입니다. 데이터베이스와의 연결을 종이컵과 실로 만들어진 종이컵 전화기라고 생각하고, 가끔 대화를 하려고 종이컵 전화기의 한쪽을 친구에게 전해주었다가 전화 놀이를 하고 난 후에는 다시 종이컵 전화기 한쪽을 되돌려받아 놓는다고 합시다. 친구끼리 가끔 전화 놀이를 하는 경우라면 문제가 없겠지만 한 시간에도 몇 십번씩 전화놀이를 한다면 전화기를 가져다주는 것이 더 번거로워지기 때문에 그 친구에게 전화기를 갖고 있으라고 하는 것이 더 나을 것입니다. 이와 마찬가지로 일반적인 경우에는 지속 연결의 장점이 없지만 연결을 생성하는 것이 부담스러운 상황이 되면 지속적인 연결이 더 유용하다는 것입니다.

mysql_select_db

[PHP 4, PHP 5]
bool mysql_select_db (string database_name [, resource link_identifier])
활성화할 데이터베이스를 선택한다.

인자	자료형	설명	비고
database_name	string	선택할 데이터베이스 이름	필수
link_identifier	resource	연결된 링크 식별자	옵션

[표 7-14] mysql_select_db 함수

mysql_select_db() 함수는 MySQL 서버에서 사용할 데이터베이스를 선택합니다. 콘솔에서 use 명령을 사용하는 것과 동일합니다.

mysql_query

[PHP 4, PHP 5]
resource mysql_query (string query [, resource link_identifier])
MySQL 서버에 하나의 쿼리를 전송한다.

인자	자료형	설명	비고
query	string	전송할 쿼리문	필수
link_identifier	resource	연결된 링크 식별자	옵션

[표 7-15] mysql_query 함수

mysql_query() 함수는 MySQL 서버에서 하나의 쿼리를 전송합니다. SELECT, SHOW와 같이 레코드를 생성하는 쿼리는 그 리소스를 제공하고 INSERT, DELETE와 같이 결과 레코드가 없는 쿼리는 성공 여부에 따른 TRUE, FALSE를 되돌려줍니다.

mysql_num_rows

[PHP 4, PHP 5]
int mysql_num_rows (resource result)
mysql_query() 함수의 결과 레코드 수를 구한다.

인자	자료형	설명	비고
result	resource	mysql_query() 함수의 리턴 값	필수

[표 7-16] mysql_num_rows 함수

mysql_num_rows() 함수는 mysql_query() 함수를 통해서 실행된 쿼리에 대한 결과 레코드의 수를 알려줍니다. 만약 결과 레코드가 없이 INSERT, UPDATE 등의 쿼리로 인해 적용된 레코드의 수를 구하고자 한다면 mysql_affected_rows() 함수를 이용해야 합니다.

mysql_affected_rows

[PHP 4, PHP 5]
int mysql_affected_rows ([resource link_identifier])
변경된 레코드의 수를 구한다.

인자	자료형	설명	비고
link_identifier	resource	링크 식별자	옵션

[표 7-17] mysql_affected_rows 함수

mysql_affected_rows() 함수는 INSERT, UPDATE, DELETE와 같이 쿼리의 결과가 적용된 레코드의 수를 알아내려고 할 때 사용할 수 있습니다. 링크 식별자를 기입하지 않으면 마지막 연결 링크를 통해서 작업이 수행됩니다. 쿼리의 결과가 실패했다면 -1, 성공했다면 적용된 레코드의 수를 반환합니다.

mysql_result

[PHP 4, PHP 5]
string mysql_result (resource result , int row [, mixed field])
쿼리의 결과 레코드를 반환한다.

인자	자료형	설명	비고
result	resource	mysql_query() 함수의 반환 값	필수
row	int	검색된 결과의 레코드 번호, 0부터 시작	필수
field	mixed	필드 순서 / 필드 이름 / 테이블.필드	옵션

[표 7-18] mysql_result 함수

mysql_result() 함수는 mysql_query() 함수의 결과 레코드를 한 셀 단위로 가져옵니다. 이 말은 레코드의 결과를 2차원 테이블로 생각했을 때, 한 행(레코드)씩 가져오는 것이 아니라 한 셀, 예를 들면 2행 3열의 칸에 있는 값을 가져온다는 것입니다. 이 함수는 특정 행의 특정 열의 값을 가져오는데 유용하지만 전체 레코드가 필요한 프로그램에서는 사용하지 않는 것이 좋습니다.

mysql_fetch_row

[PHP 4, PHP 5]
array mysql_fetch_row (resource result)
쿼리의 결과 레코드를 배열로 반환한다.

인자	자료형	설명	비고
result	resource	mysql_query() 함수의 반환 값	필수

[표 7-19] mysql_fetch_row 함수

mysql_fetch_row() 함수는 mysql_query() 함수의 결과 레코드를 한 레코드 단위로 가져오며 이 레코드를 배열 형태로 되돌려줍니다. 이 배열에서 필드값을 가져오려면 인덱스로 숫자를 사용합니다. 다음 레코드를 가져오려면 다시 이 함수를 호출하고, 만약 더 이상 레코드가 존재하지 않으면 FALSE를 반환합니다.

mysql_fetch_assoc

[PHP 4, PHP 5]
array mysql_fetch_assoc (resource result)
쿼리의 결과 레코드를 연관 배열로 반환한다.

인자	자료형	설명	비고
result	resource	mysql_query() 함수의 반환 값	필수

[표 7-20] mysql_fetch_assoc 함수

mysql_fetch_assoc() 함수는 mysql_fetch_row() 함수와 거의 동일하며 차이는 숫자 인덱스가 아닌 연관 배열의 형태로 결과를 반환한다는 것입니다. 배열의 인덱스로 필드 이름을 사용할 수 있기 때문에 보다 직관적이어서 사용하기가 편리합니다. 그리고 두 함수의 성능은 차이가 나지 않습니다.

mysql_fetch_array

[PHP 4, PHP 5]
array mysql_fetch_array (resource result [, int result_type])
쿼리의 결과 레코드를 숫자 인덱스 배열이나 연관 배열로 반환한다.

인자	자료형	설명	비고
result	resource	mysql_query() 함수의 반환 값	필수
result_type	int	결과 배열의 형태, (MYSQL_BOTH가 기본값) MYSQL_ASSOC, MYSQL_NUM, MYSQL_BOTH	옵션

[표 7-21] mysql_fetch_array 함수

mysql_fetch_array() 함수는 mysql_fetch_row() 함수와 mysql_fetch_assoc() 함수를 합친 것과 같습니다. result_type 옵션을 통해서 둘 중 하나의 방식을 선택하거나 아니면 둘 모두의 방식을 선택할 수 있습니다. 기본적으로는 두 가지 방식 모두를 지원합니다. 이 세 함수는 성능적으로 차이가 없습니다. 따라서 통합 함수인 mysql_fetch_array() 함수를 이용하는 것이 낫습니다. 그런데 숫자 인덱스 배열은 되도록이면 사용하지 않는 것이 좋습니다. 숫자 인덱스를 사용하게 되면 나중에 테이블의 변경으로 필드의 순서가 변경되었을 때 이를 반영하지 못하기 때문에 엉뚱한 위치에 엉뚱한 값을 출력하게 됩니다. 그러나 만약 for 문을 이용하여 값을 출력하고자 한다면 숫자 인덱스는 매우 유용하게 사용할 수 있습니다.

mysql_real_escape_string

[PHP 4, PHP 5]
string mysql_real_escape_string (string unescaped_string [, resource link_identifier])
특수 문자열을 쿼리에 안전한 문자열로 변환한다.

인자	자료형	설명	비고
unescaped_string	string	변환할 문자열	필수
link_identifier	resource	링크 식별자	옵션

[표 7-22] mysql_real_escape_string 함수

mysql_real_escape_string() 함수는 사용자의 입력에 의해서 동적으로 쿼리가 만들어질 때 특수 문자를 처리하여 쿼리를 안전한 형태로 변환합니다. 사용자의 입력에 의해서 작성되는 쿼리는 SQL

Injection 공격의 대상이 될 수 있기 때문에 이를 방지하고자 이 함수를 사용할 수 있습니다. 단, 이 함수를 사용하려면 반드시 MySQL 서버와 연결된 링크가 존재해야 합니다.

mysql_error

[PHP 4, PHP 5]

string mysql_error ([resource link_identifier])

최근 실행된 MySQL 작업에 대한 에러 메시지를 반환한다.

인자	자료형	설명	비고
link_identifier	resource	링크 식별자	옵션

[표 7-23] mysql_error 함수

mysql_error() 함수는 MySQL 함수를 사용하면서 발생한 에러 메시지를 반환합니다.

MySQLi 함수

MySQLi 함수들은 기존의 함수 방식과 객체 방식 두 가지 형태로 사용할 수 있습니다. MySQLi 함수가 MySQL 함수에 비해서 여러 가지 새로운 기능을 지원하지만 PHP 5 버전 이상에서만 사용할 수 있기 때문에 호환성이 중요하다면 MySQL 함수를 사용하기 바랍니다.

여기에서는 객체 형태가 아닌 함수 형태의 사용법에 대해서 알아보도록 하겠습니다.

mysqli_connect

[PHP 5]

mysqli mysqli_connect ([string host [, string username [, string password [, string dbname [, int port [, string socket]]]]]])

데이터베이스에 접속한다.

인자	자료형	설명	비고
host	string	MySQL 서버 주소	옵션

username	string	데이터베이스 사용자 계정	옵션
password	string	데이터베이스 사용자 비밀번호	옵션
dbname	string	선택할 데이터베이스 이름	옵션
port	int	MySQL 서버 포트 번호	옵션
socket	string	소켓 또는 명명된 파이프	옵션

[표 7-24] mysqli_connect 함수

mysqli_connect() 함수는 데이터베이스에 접속하고 연결이 되면 MySQL 연결 정보를 객체로 되돌려줍니다.

mysqli_close

[PHP 5]
bool mysqli_close (mysqli link)
데이터베이스의 접속을 종료한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-25] mysqli_close 함수

mysqli_close() 함수는 연결된 데이터베이스 접속을 종료합니다.

mysqli_select_db

[PHP 5]
bool mysqli_select_db (mysqli link , string dbname)
사용할 데이터베이스를 선택한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수
dbname	string	선택할 데이터베이스 이름	필수

[표 7-26] mysqli_select_db 함수

mysqli_select_db() 함수는 사용할 데이터베이스를 선택하는 함수입니다. mysqli_connect() 함수

자체에 데이터베이스를 선택하는 옵션이 있기 때문에 실제로는 데이터베이스를 중간에 변경해야 할 경우에 사용합니다.

mysqli_real_query

[PHP 5]
bool mysqli_real_query (mysqli link , string query)
데이터베이스에 쿼리를 전송한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수
query	string	쿼리	필수

[표 7-27] mysqli_real_query 함수

mysqli_real_query() 함수는 데이터베이스에 쿼리를 전송합니다. 쿼리의 결과를 얻으려면 mysqli_use_result() 함수나 mysql_store_result() 함수를 사용해야 합니다.

mysqli_store_result

[PHP 5]
mysqli_result mysqli_store_result (mysqli link)
마지막 쿼리의 결과 레코드를 전송한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-28] mysqli_store_result 함수

mysqli_store_result() 함수는 마지막에 수행된 쿼리의 결과 레코드를 전송합니다. INSERT와 같은 쿼리는 FALSE를 반환하고 결과 레코드를 읽어오지 못한 때에도 FALSE를 반환합니다.

mysqli_use_result

[PHP 5]
mysqli_result mysqli_use_result (mysqli link)
결과 레코드을 조회한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-29] mysqli_use_result 함수

mysqli_use_result() 함수는 마지막에 수행된 쿼리의 결과 레코드를 조회합니다. 이 함수와 mysqli_store_result() 함수 중 하나는 쿼리의 결과를 취득하기 전에 반드시 호출되어야 합니다. 그렇지 않으면 다른 쿼리를 전송할 수 없습니다.

mysqli_query

[PHP 5]
mixed mysqli_query (mysqli link , string query [, int resultmode])
데이터베이스에 쿼리를 전송한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수
query	string	쿼리	필수
resultmode	int	MYSQLI_USE_RESULT / MYSQLI_STORE_RESULT	옵션

[표 7-30] mysqli_query 함수

mysqli_query() 함수는 기능면에서 mysqli_real_query() 함수를 호출한 후 mysqli_use_result() 함수 혹은 mysqli_store_result() 함수를 호출한 것과 같은 역할을 합니다. resultmode의 경우 기본적으로 MYSQLI_STORE_RESULT로 동작하며 MYSQLI_USE_RESULT를 명시하게 되면 결과 레코드를 모두 취득하기 전까지는 MySQL에 다른 쿼리를 전송하는 등의 작업을 할 수 없습니다.

mysqli_multi_query

[PHP 5]
bool mysqli_multi_query (mysqli link , string query)
데이터베이스에 하나 이상의 쿼리를 전송한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수
query	string	쿼리	필수

[표 7-31] mysqli_multi_query 함수

mysqli_multi_query() 함수는 MySQLi 클래스에서 새롭게 지원하는 함수로 여러 개의 쿼리를 한 번에 전송하는 기능을 합니다. 쿼리의 결과 레코드를 얻으려면 mysqli_use_result() 함수나 mysql_store_result() 함수를 사용해야 합니다.

mysqli_next_result

[PHP 5]
bool mysqli_next_result (mysqli link)
mysqli_multi_query() 함수의 다음 결과를 준비한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-32] mysqli_next_result 함수

mysqli_next_result() 함수는 mysqli_multi_query() 함수에 의해서 실행된 결과의 다음 레코드를 가져올 수 있도록 준비합니다.

mysqli_next_result

[PHP 5]
bool mysqli_next_result (mysqli link)
mysqli_multi_query() 함수에 의한 결과의 다음 레코드를 준비한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-33] mysqli_next_result 함수

mysqli_next_result() 함수는 mysqli_multi_query() 함수에 의해서 실행된 결과의 다음 레코드를 가져올 수 있도록 준비합니다.

mysqli_more_result

[PHP 5]
bool mysqli_more_result (mysqli link)
mysqli_multi_query() 함수에 의한 결과 레코드가 더 남아있는지 확인한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-34] mysqli_more_result 함수

mysqli_more_result() 함수는 mysqli_multi_query() 함수에 의해서 실행된 결과가 남아있는지 확인합니다. 결과 레코드가 남아있다면 TRUE를, 비어 있으면 FALSE를 되돌려줍니다.

mysqli_more_result

[PHP 5]
bool mysqli_more_result (mysqli link)
mysqli_multi_query() 함수에 의한 결과 레코드가 더 남아있는지 확인한다.

인자	자료형	설명	비고
link	mysqli	MySQL 연결 객체	필수

[표 7-35] mysqli_more_result 함수

mysqli_more_result() 함수는 mysqli_multi_query() 함수에 의해서 실행된 결과가 남아있는지 확인합니다. 결과 레코드가 남아있다면 TRUE를, 비어 있으면 FALSE를 되돌려줍니다.

mysqli_fetch_array

[PHP 5]
mixed mysqli_fetch_array (mysqli_result result [, int resulttype])
결과 레코드를 숫자 인덱스 배열 혹은 연관 배열로 반환한다.

인자	자료형	설명	비고
result	mysqli_result	mysqli 결과 객체	필수
resulttype	int	MYSQLI_ASSOC, MYSQLI_NUM, MYSQLI_BOTH	옵션

[표 7-36] mysqli_fetch_array 함수

mysqli_fetch_array() 함수는 mysqli_query(), mysql_use_result(), mysql_store_result() 함수의 결과인 mysqli_result 객체를 입력받아 결과 레코드를 배열로 반환합니다. 이는 mysql_fetch_array() 함수와 동일합니다. 또한 mysqli_fetch_row() 함수와 mysqli_fetch_assoc() 함수는 확장 전의 mysql_fetch_row() 함수와 mysql_fetch_assoc() 함수와 동일합니다.

mysqli_free_result

[PHP 5]
void mysqli_free_result (mysqli_result result)
쿼리 결과를 메모리에서 해제한다.

인자	자료형	설명	비고
result	mysqli_result	mysqli 결과 객체	필수

[표 7-37] mysqli_free_result 함수

mysqli_free_result () 함수는 쿼리의 결과를 메모리에서 해제하여 메모리 공간을 확보합니다.

mysqli_autocommit

[PHP 5]
bool mysqli_autocommit (mysqli link , bool mode)
데이터베이스 갱신을 자동으로 커밋할 것인지 여부를 설정한다.

인자	자료형	설명	비고
link	mysqli	mysqli 결과 객체	필수
mode	bool	자동 커밋의 경우 TRUE	필수

[표 7-38] mysqli_autocommit 함수

mysqli_autocommit() 함수는 자동으로 커밋을 실행할지를 결정하는 함수입니다. 커밋은 트랜잭션이 종료되었음을 데이터베이스에 알려주는 것이며 커밋이 되면 갱신된 내용이 실제 데이터베이스에 적용됩니다. 트랜잭션은 일련의 작업 과정을 하나로 묶은 더 큰 단위라고 생각하면 됩니다.

현금 인출기의 동작을 세 가지로 요약해보면 다음과 같습니다.

① 계좌의 잔고를 확인한다.
② 계좌에서 인출금액을 감액한다.
③ 인출금액을 내보낸다.

이러한 세 가지 과정을 거치게 되는데 만약 2번까지 수행하고 에러가 발생하면 어떻게 될까요? 고객은 현금을 받아보지도 못했지만 금액은 이미 빠져나갔기 때문에 손실을 보게 될 것입니다. 이러한 문제가 발생한 것은 현금 인출기의 동작이 세 가지로 나누어져 있지만 실제로는 모두 하나와 같이 동작해야 한다는 것입니다. 즉, 어느것 하나라도 제대로 동작하지 않으면 큰 문제를 일으킵니다. 이런 경우 이 세 가지 과정을 하나의 트랜잭션으로 묶어서 처리를 하게 되는데, 트랜잭션으로 처리하면 2번 과정에서 문제가 발생할 때 트랜잭션 단위의 모든 작업을 초기화하여 처음으로 되돌아갑니다. 이러한 과정을 롤백(RollBack)이라고 합니다. 처음으로 되돌아갈 수 있는 이유는 모든 작업이 가상으로 이루어지기 때문입니다. 즉, 실제 계좌에서 돈이 빠져나가는 것이 아니라 계좌에서 돈이 빠져나갔다고 가정을 하고 처리하는 것입니다. 그래서 모든 과정이 정상적으로 처리되었을 때 그때 확정(커밋)이 되어 실제 시스템에 적용되는 것입니다.

mysqli_commit

[PHP 5]
bool mysqli_commit (mysqli link)
현재의 트랜잭션을 커밋한다.

인자	자료형	설명	비고
link	mysqli	mysqli 결과 객체	필수

[표 7-39] mysqli_commit 함수

mysqli_commit() 함수는 현재의 트랜잭션을 커밋합니다. 커밋을 하면 수행한 트랜잭션이 실제 데이터베이스에 적용되므로 더는 취소할 수 없습니다.

mysqli_rollback

[PHP 5]
bool mysqli_rollback (mysqli link)
현재의 트랜잭션을 롤백한다.

인자	자료형	설명	비고
link	mysqli	mysqli 결과 객체	필수

[표 7-40] mysqli_rollback 함수

mysqli_rollback() 함수는 현재의 트랜잭션을 롤백합니다. 롤백을 하면 트랜잭션 중에서 수행했던 작업들은 모두 원상태로 되돌아갑니다.

--

--

Chapter

08

사용자 인증

우리는 네이버(www.naver.com)나 싸이월드(www.cyworld.com)와 같은 사이트를 방문할 때 사용자 인증(Login)을 하려고 아이디와 비밀번호를 입력하고 들어가는 경우가 많습니다. 이는 메일과 같이 회원의 사적인 서비스를 제공하거나 인터넷 카페와 같이 인증된 회원들만 사용할 수 있는 곳에서 회원이 아닌 사용자의 접근을 막기 위한 용도 등으로 사용됩니다. 내 메일을 다른 누군가가 쉽게 훔쳐볼 수 있다면 메일 서비스는 누구도 사용하려 하지 않을 것입니다. 권한이 있는 사용자만 해당 컨텐츠를 볼 수 있게 하는 것이 사용자 인증의 핵심입니다.

일반적으로 사이트에 접속할 때 한번 사용자 인증을 거친 다음에는 브라우저를 닫거나 로그아웃을 하기 전까지는 사이트를 여기저기 드나드는 데 전혀 문제가 없습니다. 심지어는 사용자 인증을 받고 다른 사이트를 방문하였다가 되돌아오더라도 다시 사용자 인증을 받으라는 메시지를 볼 수 없는 경우가 대부분입니다. 그것은 바로 앞서 배운 쿠키나 세션을 통해서 사용자의 상태를 지속적으로 유지해주기 때문입니다.

이 장에서는 사용자 인증을 하는 여러 가지 방법과 쿠키나 세션을 이용하여 사용자의 상태를 지속적으로 유지하는 방법을 배워보도록 하겠습니다.

Section 01

HTTP 인증

사용자 인증을 생각하면 먼저 아이디와 비밀번호를 입력하는 폼을 떠올릴 것입니다. 그러나 HTTP 인증을 사용하면 별도의 폼을 작성할 필요 없이 아이디/비밀번호 입력창을 띄울 수 있습니다. HTTP 인증을 이용하여 쉽게 사용자 인증을 구성하는 방법을 알아봅시다.

HTTP 인증은 Basic 인증과 Digest 인증으로 나누어 집니다. Basic 인증은 대부분의 웹 브라우저에서 지원하지만 Digest 인증은 아직 널리 사용되지 않아서 몇몇 브라우저에서는 지원하지 않습니다. 하지만 MS 인터넷 익스플로러나 모질라, 오페라, 사파리 등과 같은 대표적인 웹 브라우저들이 최신 버전에서 Digest 인증을 지원하고 있습니다.

Basic 인증은 아주 쉽게 HTTP 인증을 사용할 수 있다는 장점에 반해 로그인 정보가 BASE64 알고리즘으로 인코딩되어 있어서 간단한 방법으로 로그인 정보가 유출될 수 있습니다. 이러한 점을 보완하고자 Digest 인증이 등장하게 되었고 Digest 인증은 MD5와 같은 해시 함수를 이용하여 로그인 정보를 전송하기 때문에 중간에서 해커에 의해 로그인 정보가 공개되는 것을 방지할 수 있습니다.

PHP에서는 기존에 Basic 인증만을 지원하였으나 PHP 5.1. 버전부터는 Digest 인증을 지원합니다. 그러나 심지어 PHP.NET에서 제공하는 PHP 매뉴얼의 Digest 예제 소스조차 제대로 동작하지 않을 만큼 널리 보급되어 있지는 않습니다. 그래서 우선은 널리 사용되고 있는 Basic 인증을 통해 간단한 HTTP 인증을 구현해보고 뒤이어 보안이 한층 강화된 Digest 인증에 대해 살펴보도록 하겠습니다.

Basic 인증

Basic 인증은 앞서 말한 보안 문제에도 불구하고 매우 널리 사용되고 있는 방식으로서 쉽게 로그인을 구현할 수 있다는 것이 장점입니다. 기본적으로 HTTP 인증을 위해서는 PHP가 CGI 방식이 아닌 아파치 웹 서버의 모듈 형태로 설치되어 있어야 합니다.

Basic 인증의 절차는 다음과 같습니다.

① 사용자가 인증이 필요한 페이지에 접속한다.
② 인증 페이지는 header() 함수를 이용하여 웹 브라우저로 사용자 인증을 요구한다.
③ 웹 브라우저는 아이디와 비밀번호 입력창을 사용자에게 보여준다.
④ 사용자가 입력한 로그인 정보는 다시 PHP 문서로 전송된다.
⑤ PHP 문서에서 로그인 정보를 확인하여 사용자 인증을 처리한다.

각각의 절차에 대해 자세히 살펴보면 다음과 같습니다.

① 사용자가 인증이 필요한 페이지에 접속한다.

GET http://ezphp.net/login/http.php HTTP/1.0

② 인증 페이지는 header() 함수를 이용하여 웹 브라우저로 사용자 인증을 요구한다.
인증 페이지는 다음과 같은 헤더 정보를 웹 브라우저로 전송합니다.

```
HTTP/1.1. 401 Unauthorized
Date: Fri, 02 Nov 2007 18:05:44 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
WWW-Authenticate: Basic realm="데이터베이스 관리 시스템"
Content-Length: 32
Connection: close
Content-Type: text/html
```

헤더에 401 unauthorized 메시지를 전송하면 웹 브라우저는 사용자 인증이 안 되어 있으니 사용자 인증이 필요하다고 느끼고 로그인창을 보여줍니다. 현재 Basic 인증을 하고 있으므로 WWW-Authenticate 값은 Basic이 됩니다. realm은 로그인 시에 출력되는 메시지로 프로그래머가 지정한 값이며 주로 사용자가 접속하고자 하는 영역의 이름을 표시합니다. 이 헤더 정보와 함께 사용자 인증 실패 시 출력할 HTML 문서를 같이 전송합니다.

[그림 8-1] HTTP Basic 인증의 로그인 윈도우

③ 웹 브라우저는 아이디와 비밀번호 입력창을 사용자에게 보여준다.

②번으로부터 헤더 정보를 넘겨받으면 웹 브라우저는 위와 같은 모양의 아이디와 비밀번호 입력창을 사용자에게 보여줍니다. 여기서 매우 편리한 점은 비밀번호를 자동 저장하는 기능이 있어서 만약 〈암호 저장〉을 클릭한 후 로그인에 성공하면 다음 접속에서는 사용자 이름과 비밀번호를 기입할 필요 없이 로그인할 수 있습니다.

주) 일반적으로 '암호'라는 용어보다 '비밀번호'라는 용어를 많이 사용합니다. 그래서 이 책에서는 본문을 비롯하여 소스 코드에서도 비밀번호로 표기했습니다. 그런데 [그림 8-1]처럼 윈도우에서 제공하는 대화상자에서는 암호로 표기되어 있어서 이 책에서 사용한 비밀번호와 다른 것처럼 비춰질 수 있습니다. 그림에서 보이는 암호는 비밀번호와 같은 것임을 밝힙니다.

[그림 8-2] 비밀번호 저장을 한 후 다음번에 접속한 경우

그러나 컴퓨터를 여러 명이 같이 사용하는 공공장소와 같은 곳에서 비밀번호 저장을 하면 명의 도용과 같은 일이 벌어질 수 있으므로 반드시 개인 PC에서만 비밀번호 저장을 하기 바랍니다.

④ 사용자가 입력한 로그인 정보는 다시 PHP 문서로 전송된다.

③번에서 사용자가 입력한 로그인 정보는 다시 원래의 PHP 문서로 전달됩니다. 즉, 일반적인 경우처럼 한 번의 접속으로 이루어지는 것이 아니라 HTTP 인증은 두 번에 나뉘어 진행됩니다.

```
GET http://ezphp.net/login/http.php HTTP/1.0
Accept: image/gif, ...
Accept-Language: ko
Authorization: Basic YnJvd246MTIzNA==
User-Agent: Mozilla/4.0 ...
Host: ezphp.net
```

다시 PHP 문서로 요청을 보내면서 로그인 정보를 base64 알고리즘으로 인코딩하여 전송합니다. 이처럼 단순히 base64로 인코딩된 문자열을 전송하므로 누군가 엿보고자 한다면 base64 알고리즘으로 쉽게 디코딩할 수 있습니다. "YnJvd246MTIzNA=="를 디코딩하면 "brown:1234"라는 문자열을 쉽게 알 수 있습니다.

⑤ PHP 문서에서 로그인 정보를 확인하여 사용자 인증을 처리한다.

사용자가 입력한 아이디와 비밀번호 정보는 각각 $_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'] 변수에 저장됩니다. 우리는 이 두 변수 값과 실제 로그인이 가능한 권한을 가진 아이디와 비밀번호를 비교하여 사용자 인증을 처리하면 됩니다.

실제 예제를 통해서 알아봅시다.

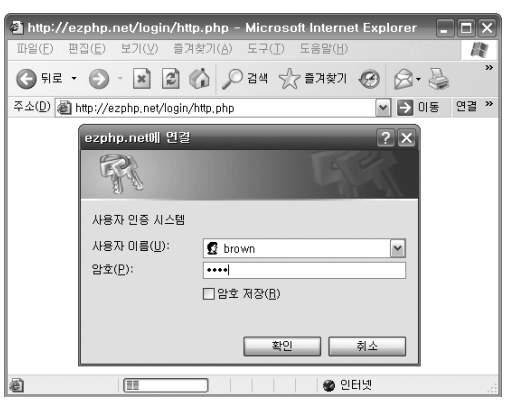
[예제 8-1] HTTP Basic 인증을 이용한 사용자 이름과 비밀번호 전달

```
1  <?
2     //인증이 되어 있지 않은 경우
3     if (!$_SERVER['PHP_AUTH_USER'])
4     {
```

```
5       //인증이 되어 있지 않으므로 인증을 시도한다.
6       header('WWW-Authenticate: Basic realm="사용자 인증 시스템"');
7       header('HTTP/1.0 401 Unauthorized');
8
9       //인증에 실패하였을 경우 에러 메시지를 출력한다.
10      echo '사용자 인증에 실패하였습니다.';
11      exit;
12    }
13    else
14    {
15      echo "<p>사용자 이름 : {$_SERVER['PHP_AUTH_USER']}</p>";
16      echo "<p>비밀번호 : {$_SERVER['PHP_AUTH_PW']}</p>";
17    }
18 ?>
```

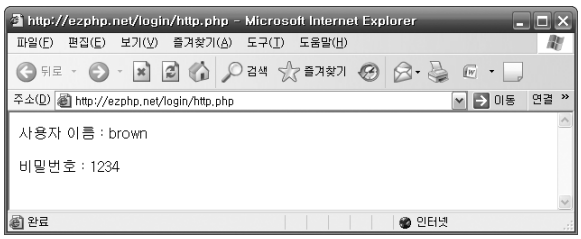
$_SERVER['PHP_AUTH_USER'] 변수에는 HTTP Basic 인증을 통해 전달된 사용자 이름 정보가 기록되어 있습니다. 만약 이 값이 비어 있다면 인증되지 않은 상태이므로 인증을 요구하는 헤더를 웹 브라우저로 전송합니다. 이때 인증에 실패할 경우에 보여주는 에러 메시지를 함께 전송하는데 사용자가 취소 버튼을 클릭하거나 인터넷 익스플로러의 경우 인증을 세 번 시도해서 모두 실패하면 이 메시지를 출력합니다.

[예제 8-1]을 실행하면 다음과 같이 사용자 인증을 위한 대화상자를 띄워 줍니다.(인터넷 익스플로러 버전에 따라 모양은 차이가 있습니다)



[그림 8-3] 예제 8-1의 실행결과

사용자의 이름과 비밀번호를 모두 기입하고 확인을 누르면 다음과 같은 결과를 볼 수 있습니다.



[그림 8-4] 예제 8-1에서 입력한 값 확인

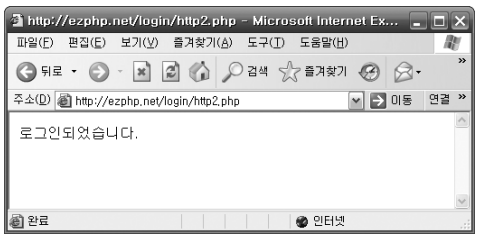
사용자가 입력한 값이 제대로 전송되고 있음을 알 수 있으며 만약 이와 같은 결과를 보고 난 후 페이지를 새로 고침 하거나 다른 페이지로 이동하였다가 다시 돌아와도 여전히 위와 같은 결과를 볼 수 있습니다.

HTTP Basic 인증을 통해서 입력받은 값을 확인하였으니 이를 이용하여 아이디와 비밀번호를 검증해 봅시다.

[예제 8-2] HTTP Basic 인증을 이용한 아이디 및 비밀번호 검증

```
<?
  $user = "brown";
  $pass = "1234";

  if ( $_SERVER['PHP_AUTH_USER']==$user &&
  $_SERVER['PHP_AUTH_PW']==$pass )
  {
    echo "로그인되었습니다.";
  }
  else {
    header('WWW-Authenticate: Basic realm="사용자 인증 시스템"');
    header('HTTP/1.0 401 Unauthorized');
    echo '사용자 인증에 실패하였습니다.';
    exit;
  }
?>
```

위와 같이 코드를 작성하면 인터넷 익스플로러 기준으로 사용자 인증이 되기 전까지 총 3번의 기회가 주어집니다. 만약 3회 안에 성공적으로 인증되면 브라우저를 닫기 전까지는 해당 페이지에서 다시 사용자 인증을 할 필요가 없습니다.



[그림 8-5] 인증이 성공한 경우

하지만 다른 사용자 아이디로 로그인하고자 할 때처럼 때로는 웹 브라우저를 닫지 않고 로그아웃을 해야 할 때가 있습니다. HTTP 인증에서는 기본적으로 로그아웃이 지원되지 않습니다. 하지만 HTTP 인증의 특성을 이용하면 비록 매끄럽지는 못하지만 로그아웃 기능을 구현할 수 있습니다.

[예제 8-3] HTTP Basic 인증 후 로그아웃 구현

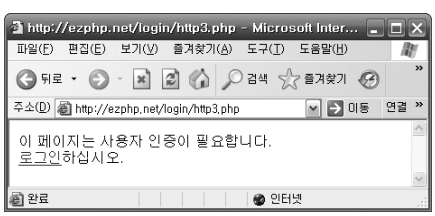
```
<?
  if (!isset($_SERVER['PHP_AUTH_USER']) || $_COOKIE['login'] != "1") {
    //웹 브라우저의 헤더로 전송
    header('WWW-Authenticate: Basic realm="사용자 인증 시스템"');
    header('HTTP/1.0 401 Unauthorized');
    setcookie ("login", "1");
    //취소 버튼을 눌렀을 경우
    echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
    echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
    exit;
  }
  else {
    //로그아웃
    if($_GET['logout'] == "1") {
      setcookie ("login", "", 0);
      header("location: {$_SERVER['PHP_SELF']}");
      exit;
    }
    //사용자 인증 처리
    if($_SERVER['PHP_AUTH_USER'] == "brown" &&
    $_SERVER['PHP_AUTH_PW']== "1234") {
      //로그인된 후 메시지 출력
      echo "안녕하세요. {$_SERVER['PHP_AUTH_USER']}님<BR>";
      echo "<a href='".$_SEVER['PHP_SELF']."?logout=1'>로그아웃</a>";
    }
    else {
      //사용자 인증 실패 후
      setcookie ("login", "0");
      echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
      echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
```

```
31      exit;
32      }
33    }
34 ?>
```

HTTP 인증에서 로그아웃을 구현하는 방법은 새로 사용자 인증 대화상자를 띄우는 것입니다. 새롭게 사용자 인증 대화상자를 띄우고 값을 입력하지 않은 상태에서 확인 버튼을 누르면 기존에 로그인한 정보가 빈 값으로 변경됩니다. 결국 이 빈 값은 사용자 인증에서 실패하므로 로그아웃과 같은 역할을 합니다. 그러나 로그아웃을 위해서 사용자 인증 대화상자를 띄웠을 때 사용자가 취소 버튼을 누르면 취소 버튼을 눌렀을 때의 메시지만 나타날 뿐 로그아웃이 되지 않습니다. 그 이유는 취소 버튼을 누른 경우 인증 정보가 초기화되지 않기 때문입니다. 그래서 사용자는 로그아웃된 줄 믿고 웹 브라우저를 그대로 두었다가 큰일을 당하는 수가 있습니다.

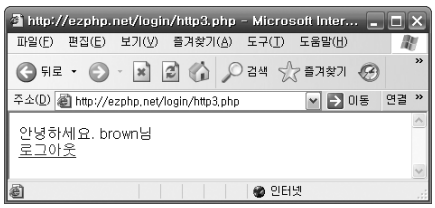
[그림 8-6] 로그아웃을 클릭한 결과

위의 그림처럼 [예제 8-3]을 실행한 후 로그아웃을 하면 취소 버튼을 클릭하였을 때 다음과 같은 메시지를 볼 수 있습니다.



[그림 8-7] 취소 버튼을 클릭한 결과

그러나 로그인 링크를 클릭해보면 다음과 같이 전혀 로그아웃이 되어있지 않음을 알 수 있습니다.



[그림 8-8] 로그아웃이 제대로 되지 않음

현재 HTTP 인증을 사용하는 많은 프로그램이 이와 같은 문제를 갖고 있습니다. 대표적으로 PHP를 통해 MySQL 데이터베이스를 관리하는 phpmyadmin 프로그램만 하더라도 거짓 로그아웃 메시지가 나타납니다.

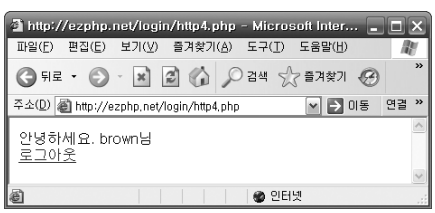
그래서 [예제 8-3]의 소스를 수정하여 거짓 로그아웃 메시지를 출력하지 못하도록 바꾸어 보겠습니다.

[예제 8-4] 거짓 로그아웃 방지

```
<?
  if (!isset($_SERVER['PHP_AUTH_USER']) || $_COOKIE['login'] != "1") {
    //웹 브라우저의 헤더로 전송
    header('WWW-Authenticate: Basic realm="사용자 인증 시스템"');
    header('HTTP/1.0 401 Unauthorized');
    setcookie ("login", "1");

    //취소 버튼을 눌렀을 경우
    echo "<meta http-equiv=\"refresh\" content=\"0\">";
    exit;
  }
  else {
    //로그아웃
    if($_GET['logout'] == "1") {
      setcookie ("login", "", 0);
      header("location: {$_SERVER['PHP_SELF']}");
      exit;
    }

    //사용자 인증 처리
    if($_SERVER['PHP_AUTH_USER'] == "brown" &&
    $_SERVER['PHP_AUTH_PW']== "1234") {
      //로그인이 된 후 메시지 출력
      echo "안녕하세요. {$_SERVER['PHP_AUTH_USER']}님<BR>";
      echo "<a href='".$_SEVER['PHP_SELF']."?logout=1'>로그아웃</a>";
    }
      else {
```

```
28          //사용자 인증 실패 후
29          setcookie ("login", "0");
30          echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
31          echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
32          exit;
33      }
34    }
35 ?>
```

지금까지 인증 헤더 밑에는 실패할 때 보여줄 메시지를 추가했으나 실패 메시지가 웹 브라우저에서 처리되어 거짓 로그아웃이 되는 것을 방지하기 위해 다시 페이지를 불러들이도록 변경했습니다. 이렇게 하면 모든 실패 메시지는 한 곳에서 처리가 되고 새로 고침으로 인해서 거짓 로그아웃 정보가 갱신됩니다.



[그림 8-9] 새로 고침을 통한 거짓 로그아웃 방지

위의 그림은 사용자 인증 대화상자에서 취소 버튼을 눌렀을 때 출력되는 메시지입니다. 즉, 로그아웃이 취소되어 로그아웃이 제대로 되지 않았음을 알 수 있습니다. 그래서 사용자는 다시금 로그아웃 버튼을 누릅니다.

그런데 여기서 "사용자는 개발자가 아니다."라는 말을 되새길 필요가 있습니다. 개발자는 자신이 어떤 구조로 프로그램을 만들었는지 알고 있기 때문에 너무나 당연시하며 프로그램을 사용하지만 사용자는 프로그램이 어떤 구조로 개발되었는지에 대해 전혀 관심이 없습니다. 즉, 사용자 인터페이스가 누가 보아도 직관적이지 못할 경우 사용법에 대한 지시사항이 없다면 사용자는 제대로 프로그램을 사용하지 못합니다.

일반적으로 로그아웃은 로그아웃 버튼을 클릭하는 것만으로 로그아웃이 처리됩니다. 이러한 방식에 익숙해져 있는 사용자는 HTTP 인증과 같이 다시 사용자 인증 대화상자가 띄워지는 경우 당황할 수 있습니다. 그래서 대부분은 다시 취소 버튼을 누르게 됩니다. 취소 버튼을 누를 경우 로그아웃이 되지 않기 때문에 사용자는 무한히 반복해서 로그아웃-취소-로그아웃-취소를 하고 있을지도 모르는 일입니다.

그래서 사용자 인증에서 사용되는 대화상자와 로그아웃에서 사용되는 대화상자를 구분할 필요가

있습니다. 대화상자 자체는 프로그래머가 수정할 수 없는 부분이고 고칠 수 있는 부분은 realm밖에 없습니다. 그러나 realm이 해당 영역을 나타내는 이름이기 때문에 로그인할 때 realm 이름과 로그아웃할 때 realm 이름이 다르면 로그인이 제대로 되지 않는 문제가 생길 수 있습니다.

따라서 다음과 같이 realm에 로그인과 로그아웃 모두에 관한 지시사항을 기입하는 방법밖에 없습니다.

```
$realm = "로그인 하고자 하는 아이디와 패스워드를 입력하거나 로그아웃을 하려면 값을 입력하지 말고
          확인을 클릭하세요.";
header("WWW-Authenticate: Basic realm=\"$realm\"");
```

적절한 공백을 사용하여 보기 좋게 출력하도록 합니다. [예제 8-4]의 소스를 위와 같이 수정하면 다음과 같은 대화상자를 볼 수 있습니다.

[그림 8-10] 로그아웃을 위한 지시사항을 표시

그러나 만약 인터넷 익스플로러에 대해서만 고려한다면 다음과 같은 간단한 자바스크립트를 이용하여 로그아웃을 구현할 수 있습니다.

```
<script>
document.execCommand('ClearAuthenticationCache');
</script>
```

이 스크립트는 인증 정보를 모두 삭제하는 것으로 인터넷 익스플로러에서 지원하는 명령입니다. 따라서 파이어폭스나 크롬과 같은 다른 브라우저에서는 제대로 동작하지 않습니다. 이를 적용한 소스는 다음과 같습니다.

[예제 8-5] 인터넷 익스플로러만의 HTTP 인증 로그아웃

```
1  <?
2    if (!isset($_SERVER['PHP_AUTH_USER'])) {
3      //웹 브라우저의 헤더로 전송
4      header('WWW-Authenticate: Basic realm="login"');
5      header('HTTP/1.0 401 Unauthorized');
```

```
6
7       //취소 버튼을 눌렀을 경우
8       echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
9       echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
10      exit;
11    }
12    else {
13      //로그아웃
14      if($_GET['logout'] == "1") {
15        echo "<script>\n";
16        echo " document.execCommand('ClearAuthenticationCache') \n";
17        echo "</script>\n";
18        echo "로그아웃 되었습니다.<BR>";
19        echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
20        exit;
21      }
22
23      //사용자 인증 처리
24      if($_SERVER['PHP_AUTH_USER'] == "brown"
25      && $_SERVER['PHP_AUTH_PW']== "1234") {
26        //로그인이 된 후 메시지 출력
27        echo "안녕하세요. {$_SERVER['PHP_AUTH_USER']}님<BR>";
28        echo "<a href='".$_SEVER['PHP_SELF']."?logout=1'>로그아웃</a>";
29      }
30      else {
31        //사용자 인증 실패 후
32        echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
33        echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
34        exit;
35      }
36    }
37  ?>
```

Basic 인증으로 로그인 상태 유지하기

예제를 통해서 HTTP 인증을 어떻게 사용하는지 배웠습니다. 그러나 시작과는 달리 여러 가지 보완 사항을 덧붙여보면 생각보다는 간단하지 않음을 알 수 있습니다. 하지만 일반적인 사용자 인증도 모든 문제를 해결하고자 하면 간단하지 않기 때문에 특별히 어려운 것은 아니라고 생각됩니다.

사용자 인증에서 아이디와 비밀번호를 검증하는 것이 물론 제일 중요하지만 그 상태를 유지하는 것도 중요하지 않을 수 없습니다. 아무리 로그인을 잘 하였다고 하더라도 상태 유지가 안 되면 무용지

물이 되어버릴 것입니다.

HTTP 인증 상태를 유지하는 방법은 매우 간단합니다. 로그인이 요구되는 모든 페이지에 앞서 작성한 사용자 인증 코드를 추가해주기만 하면 됩니다. 너무나 당연하게도 매번 로그인을 할 필요가 전혀 없습니다. 어느 페이지에서건 한 번 제대로 사용자 인증이 되면 웹 브라우저에 인증 정보가 남아 있기 때문에 웹 브라우저를 닫지않는 한 지속적으로 인증 상태가 유지됩니다.

로그인이 필요한 페이지마다 소스 코드를 추가해야 하므로 앞의 소스 코드를 다음과 같이 변경하고 http_auth.php라는 이름으로 파일을 생성합니다.

[예제 8-6] 인클루드를 위해서 수정된 HTTP 인증 코드

```
<?
  if (!isset($_SERVER['PHP_AUTH_USER']) || $_COOKIE['login'] != "1") {

    //웹 브라우저의 헤더로 전송
    $realm = "로그인하고자 하는 아이디와 패스워드를 입력하거나 로그아웃을
    하려면 값을 입력하지 말고 확인을 클릭하세요.";
    header("WWW-Authenticate: Basic realm=\"$realm\"");
    header('HTTP/1.0 401 Unauthorized');
    setcookie ("login", "1");

    //취소 버튼을 눌렀을 경우
    echo "<meta http-equiv=\"refresh\" content=\"0\">";
    exit;
  }
  else {
    //로그아웃
    if($_GET['logout'] == "1") {
      setcookie("login", "", 0);
      header("location: {$_SERVER['PHP_SELF']}");
      exit;
    }

    //사용자 인증 처리
    if($_SERVER['PHP_AUTH_USER'] != "brown" ||
    $_SERVER['PHP_AUTH_PW'] != "1234") {
      //사용자 인증 실패 후
      setcookie ("login", "0");
      echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
      echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
      exit;
    }
  }
?>
```

이렇게 소스 코드를 변경하는 이유는 이 파일을 다른 페이지에서 쉽게 인클루드할 수 있게 하기 위해서입니다. 바뀐 부분은 아이디와 비밀번호 검증 부분으로 아이디와 비밀번호 둘 중 하나라도 맞는 것이 없다면 인증에 실패하는 코드로 변경하였습니다. 이 파일을 인클루드하고자 사용자 인증이 필요한 페이지의 상단에 추가하고 만약 인증에 성공하면 계속 코드를 실행하고, 인증에 실패한 경우 코드 실행을 중지하기 위해서입니다.

이 파일을 다음과 같이 인클루드하여 사용자 인증 상태를 유지할 수 있습니다.

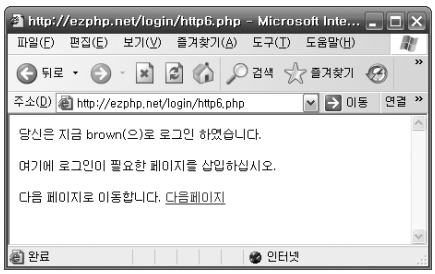
[예제 8-7] http6.php – 사용자 인증 상태 유지하기

```
1  <? require_once("http_auth.php"); ?>
2  <HTML>
3  <BODY>
4    <font size=2>
5    당신은 지금 <?=$_SERVER['PHP_AUTH_USER']?>(으)로 로그인 하였습니다.<P>
6    여기에 로그인이 필요한 페이지를 삽입하십시오.<P>
7    다음 페이지로 이동합니다. <a href="http7.php">다음페이지</a>
8    </font>
9  </BODY>
10 </HTML>
```

[예제 8-8] http7.php – 사용자 인증 상태 유지하기

```
1  <? require_once("http_auth.php"); ?>
2  <HTML>
3  <BODY>
4    <font size=2>
5    당신은 지금 <?=$_SERVER['PHP_AUTH_USER']?>(으)로 로그인 하였습니다.<P>
6    이전 페이지에 이어 지속적으로 인증상태가 유지되고 있습니다.<P>
7    로그아웃을 원하면 <a href="http7.php?logout=1">로그아웃</a>을 클릭하십시오.
8    </font>
9  </BODY>
10 </HTML>
```

[예제 8-8]의 소스를 실행하여 로그인을 해보면 다음과 같은 HTML 문서를 볼 수 있습니다.



[그림 8-11] 로그인에 성공한 경우

이어서 다음 페이지를 클릭하면 사용자 인증 대화상자 없이 다음과 같은 로그인된 상태의 페이지를 볼 수 있습니다.

[그림 8-12] 사용자 인증이 유지된 페이지

Digest 인증

앞서 HTTP 인증 중에서 Basic 인증에 대해 알아보았습니다. 그러나 앞서 언급하였듯이 HTTP Basic 인증은 악의적인 해커에게 아이디와 비밀번호가 노출될 가능성이 있습니다. 사실 SSL(Secure Socketl Layer)을 사용하여 보안이 적용된 페이지를 구현하지 않는 이상 패킷 스니핑에 의한 아이디와 패스워드 노출을 막기는 힘듭니다. 이것이 바로 보안 허점에도 불구하고 HTTP Basic 인증이 사용되는 이유입니다. "노출이 꺼려지면 SSL을 이용하라"는 것입니다.

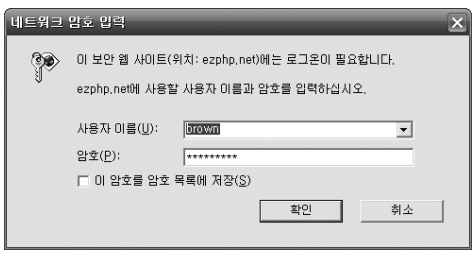
HTTP Basic 인증의 보안적 문제를 해결하고자 Digest 인증이 등장하였습니다. 아직 여러 웹 서버와 웹 브라우저가 Digest 인증을 지원하지 않지만 아파치와 PHP 그리고 인터넷 익스플로러 최신 버전의 조합에서는 Digest 인증을 구현할 수 있습니다. 그래서 짧게나마 Digest 인증을 어떻게 구현할 수 있는지 살펴보도록 하겠습니다.

우선 HTTP Digest 인증을 위한 헤더 설정을 살펴봅시다.

```
header('WWW-Authenticate: Digest realm="' . $realm . '",qop="auth",
        nonce="'. uniqid() .'", opaque="'. md5($realm).'"');
```

기존 Basic 인증에서는 단순히 realm까지만 입력하면 되었는데 조금 복잡해졌습니다. 우선 qop는 quality of protection의 약자로 auth, auth-int, token과 같은 세 종류의 값을 가질 수 있습니다. 여기서는 auth를 사용합니다. nonce는 서버 쪽에서 생성한 유일한 값을 기록하는 것이어서 uniqid() 함수를 이용하여 유일한 값을 생성합니다. 마지막으로 opaque는 $realm 값을 이용하여 해시값을 생성합니다. Digest 인증에 대해 보다 자세히 알고 싶다면 http://www.ietf.org/rfc/rfc2617에서 Digest에 대한 모든 것을 얻을 수 있습니다.

위의 헤더를 기존 [예제 8-5]에 적절히 수정 반영한다면 다음과 같이 변화된 모습의 대화상자를 볼 수 있습니다.



[그림 8-13] Digest 인증창

그림에서 보는 바와 같이 기존의 대화상자와 약간 모양이 바뀌었으며 무엇보다도 열쇠 모양의 그림이 보안에 더 역점을 주었다는 느낌을 강하게 어필하는 것 같습니다.

보안이 강화되면서 사용자 인증을 처리하는 과정이 조금 복잡해졌습니다. 기존에는 $_SERVER['HTTP_AUTH_USER']와 같은 방법으로 간단히 사용자 이름과 비밀번호에 접근할 수 있었던데 반해서 Digest 인증에서는 $_SERVER['HTTP_AUTH_DㄴIGEST'] 변수 하나만을 제공하며 여기에 각종 정보가 평문이나 해시값으로 기록되어 있습니다. 그래서 사용자 이름과 같은 정보를 얻고자 할 때는 $_SERVER['HTTP_AUTH_DIGEST'] 변수에서 PHP의 도움 없이 직접 가져와야만 합니다.

일단 $_SERVER['HTTP_AUTH_DIGEST'] 변수에 저장된 값이 어떤 모양인지 살펴보겠습니다.

```
username="brown",
realm="ezphp.net",
qop="auth",
algorithm="MD5",
uri="/login/digest.php",
nonce="472d4699aba99",
nc=00000001,
cnonce="d9736db68233f72060e43e416d29ebe9",
opaque="2376e4fe270a0eaee637100877a779f3",
response="3d65f9e6943c17ba9ff28108c1351bba"
```

Digest 인증에서 사용되는 값은 총 10가지입니다. username은 사용자 인증 대화상자에 입력한 사용자 이름이 기록된 값이고 realm, qop, nonce, opaque는 헤더 정보를 통해 웹 브라우저로 전송했던 값입니다. 또한 algorithm은 암호화에 사용되는 알고리즘을 말하며 앞서 잠시 언급하였듯이 MD5 해시 함수를 이용하고 있습니다. uri는 사용자 인증을 요구하는 현재 페이지의 주소를 기록하고 nc는 nonce-count의 약자로 nonce 값을 몇 번이나 호출하였는지에 대한 정보를 기록합니다. 그리고 cnonce는 악의적인 해커에 의한 평문 공격을 막으려고 웹 브라우저에서 생성한 값입니다. 마지막으로 response는 이 모든 항목을 통해서 최종적으로 생성한 값으로 최종 사용자 인증 검증을 위해서 생성된 값입니다.

이제 우리는 위와 같은 형식으로 제공되는 $_SERVER['HTTP_AUTH_DIGEST'] 변수를 분석하여 각 항목을 연관 배열에 저장해야 합니다. 예를 들면 $data['username']의 값이 "brown"과 같은 값을 갖도록 해야 합니다. 이러한 작업을 위해서 여러 가지 방법을 생각할 수 있으나 $_SERVER['HTTP_AUTH_DIGEST'] 변수에 기록된 형식이 마치 연관 배열을 생성하는 것과 흡사한 모양이기에 변수 값을 약간 조정하여 배열 생성문을 만드는 방법을 선택했습니다.

지난 기억을 되살려보면 배열은 다음과 같은 방법으로 생성할 수 있습니다.

```
$data = array(username=>"brown", realm=>"ezphp.net");
```

뭔가 감이 오십니까? $_SERVER['HTTP_AUTH_DIGEST'] 변수 값과 배열 생성문의 차이는 '='를 '=>'으로 변경하면 된다는 것을 알 수 있습니다. 그러기 위해서 앞서 배운 str_replace 함수를 이용해 보면 다음과 같은 방법으로 변경할 수 있습니다.

[예제 8-9] $_SERVER['HTTP_AUTH_DIGEST'] 변수 값을 배열에 저장하기

```
<?
   $txt = '
   username="brown",
   realm="ezphp.net",
   qop="auth",
   algorithm="MD5",
   uri="/login/digest.php",
   nonce="472d4699aba99",
   nc=00000001,
   cnonce="d9736db68233f72060e43e416d29ebe9",
   opaque="2376e4fe270a0eaee637100877a779f3",
   response="3d65f9e6943c17ba9ff28108c1351bba"
   ';

   $txt = str_replace("=", "=>", $txt);
   echo $txt;
?>
```

[예제 8-9]의 결과는 다음과 같습니다.

```
username=>"brown",
realm=>"ezphp.net",
qop=>"auth",
algorithm=>"MD5",
uri=>"/login/digest.php",
nonce=>"472d4699aba99",
nc=>00000001,
cnonce=>"d9736db68233f72060e43e416d29ebe9",
opaque=>"2376e4fe270a0eaee637100877a779f3",
response=>"3d65f9e6943c17ba9ff28108c1351bba"
```

이 결과의 앞뒤에 배열을 생성하기 위한 구문만 넣어주면 됩니다.

[예제 8-10] 문자열을 이용하여 배열을 생성하기

```
<?
   $txt = '
   username="brown",
   realm="ezphp.net",
   qop="auth",
   algorithm="MD5",
   uri="/login/digest.php",
   nonce="472d4699aba99",
   nc=00000001,
   cnonce="d9736db68233f72060e43e416d29ebe9",
   opaque="2376e4fe270a0eaee637100877a779f3",
   response="3d65f9e6943c17ba9ff28108c1351bba"
   ';

   $txt = str_replace("=", "=>", $txt);
   $data = array($txt);
   print_r($data);
?>
```

[예제 8-10]의 결과는 다음과 같습니다.

```
Array
(
    [0] =>
    username=>"brown",
```

```
    realm=>"ezphp.net",
    qop=>"auth",
    algorithm=>"MD5",
    uri=>"/login/digest.php",
    nonce=>"472d4699aba99",
    nc=>00000001,
    cnonce=>"d9736db68233f72060e43e416d29ebe9",
    opaque=>"2376e4fe270a0eaee637100877a779f3",
    response=>"3d65f9e6943c17ba9ff28108c1351bba"

)
```

그런데 결과를 잘 살펴보면 의도한 바와 다르게 저장된 것을 알 수 있습니다. 분명히 $data['username']과 같은 방법으로 "brown"이란 값을 얻고자 하였는데 항목별로 각각 연관 배열을 생성하지 않고 전체 문장이 배열의 값이 되어버린 것입니다.

이것은 array($txt)와 같은 방법으로 배열을 생성했을 때 $txt가 실제로 배열 생성문 형식으로 구성되어 있다 하더라도 하나의 문자열로 판단하여 배열을 생성하기 때문입니다. 이것을 해결하기 위하여 eval() 함수를 사용합니다.

[예제 8-11] eval() 함수를 이용하여 배열 생성하기

```
1  <?
2    $txt = '
3    username="brown",
4    realm="ezphp.net",
5    qop="auth",
6    algorithm="MD5",
7    uri="/login/digest.php",
8    nonce="472d4699aba99",
9    nc=00000001,
10   cnonce="d9736db68233f72060e43e416d29ebe9",
11   opaque="2376e4fe270a0eaee637100877a779f3",
12   response="3d65f9e6943c17ba9ff28108c1351bba"
13   ';
14
15   //배열 생성문 형식으로 변경한다.
16   $txt = str_replace("=", "=>", $txt);
17
18   //$data 배열을 생성한다.
19   $txt = "\$data = array($txt);";
20   eval($txt);
21
22   echo "<PRE>";
```

```
23    print_r($data);
24    echo "</PRE>";
25 ?>
```

[예제 8-11]에서 변경된 부분은 다음과 같습니다.

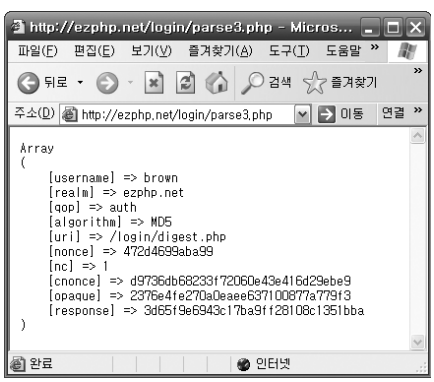
```
19    $txt = "\$data = array($txt);";
20    eval($txt);
```

위에서 $data 앞에 역슬래시(\)를 추가한 것은 따옴표 안의 문자열에서 $data라는 문자열이 변수로 대체되는 것을 막기 위함입니다. $txt는 변수로 우리가 지정한 값을 문자열에 추가해야 하지만 $data는 글자 그대로 남아 있어야 하는 값이기 때문입니다.

```
$data = array(username="brown", ...);
```

즉, eval 함수는 위와 같은 문장을 PHP로 실행합니다.

[예제 8-11]의 결과는 다음과 같습니다.



[그림 8-14] Digest 인증 정보를 배열로 저장

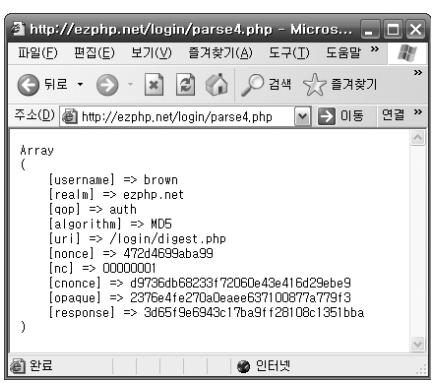
결과를 보면 각 항목별로 배열이 저장된 것을 알 수 있습니다. 그런데 문제는 nc 값이 숫자로 판단되어 1로 변경되어 버렸습니다. 원래는 "00000001"이란 값인데 말입니다. 그래서 1을 "00000001"로 변경해주는 작업이 필요합니다. 방법을 생각해보면 자리수가 8자리이니 부족한 자릿수만큼 0을 채워주면 될 듯합니다.

[예제 8-12] 자릿수만큼 숫자 채우기

```
1  <?
2     $txt = '
3     username="brown",
```

```
    realm="ezphp.net",
    qop="auth",
    algorithm="MD5",
    uri="/login/digest.php",
    nonce="472d4699aba99",
    nc=00000001,
    cnonce="d9736db68233f72060e43e416d29ebe9",
    opaque="2376e4fe270a0eaee637100877a779f3",
    response="3d65f9e6943c17ba9ff28108c1351bba"
    ';

    //배열 생성문 형식으로 변경한다.
    $txt = str_replace("=", "=>", $txt);

    //$data 배열을 생성한다.
    $txt = "\$data = array($txt);";
    eval($txt);

    $data[nc] = str_pad($data[nc], 8, "0", STR_PAD_LEFT);

    echo "<PRE>";
    print_r($data);
    echo "</PRE>";
?>
```

[예제 8-12]의 결과는 다음과 같습니다.



[그림 8-15] 자릿수만큼 0을 채운 결과

값이 모두 항목별로 잘 정리되어 있는 것을 확인할 수 있습니다. 그런데 아직 문제가 남아 있습니다. 총 10가지 항목 중에서 변경될만한 항목은 username과 realm 정도입니다. 그런데 여기에 만약 "=" 기호가 들어간다면 str_replace 함수를 통해서 "=>" 기호로 변경될 것입니다. 이 때문에 검

증이 올발라지지 않습니다.

이러한 현상을 막고자 배열에 저장된 값 중에서 "=〉" 기호를 다시금 "=" 기호로 변경해주는 것을 생각해 볼 수 있습니다.

```
$txt = "\$data = array($txt);";
eval($txt);
$data[nc] = str_pad($data[nc], 8, "0", STR_PAD_LEFT);
$data = str_replace("=>","=",$data);
```

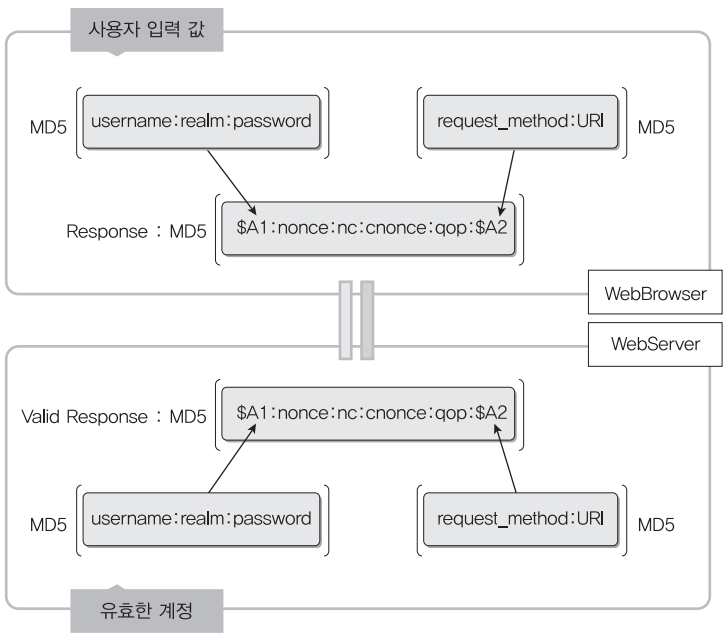
위와 같이 적용하면 모든 배열의 요소에 "=〉" 기호가 있다면 모두 "=" 기호로 변경되어 원래의 값으로 돌아오게 됩니다. 만약 원문에 "=〉" 기호가 있었다 하더라도 첫 번째 reaplce에 의해 "=〉〉"로 변경되고 이것이 두 번째 replace에 의해서 "=〉"로 변경되기 때문에 문제가 되지 않습니다.

완성된 패턴 분석 함수는 다음과 같습니다.

```
//패턴 분석 함수
function http_digest_parse()
{
  $txt = str_replace("=","=>",$_SERVER['PHP_AUTH_DIGEST']);
  $txt = "\$data = array($txt);";
  eval($txt);
  $data[nc] = str_pad($data[nc], 8, "0", STR_PAD_LEFT);
  $data = str_replace("=>","=",$data);
  return $data;
}
```

이제 패턴 분석 함수를 통해서 구해진 각 항목의 값을 가지고 사용자 인증을 검증해 보도록 합시다. HTTP Basic 인증에서는 사용자의 이름과 비밀번호가 평문으로 직접 전송되었기 때문에 쉽게 사용자 인증을 확인할 수 있었습니다. 그러나 Digest 인증에서는 사용자의 이름은 평문으로 전송되지만 어디에서도 비밀번호가 전달되지 않습니다. 단지 비밀번호가 다른 항목과 더불어 MD5 해시 함수를 통해서 해시값으로 전해질 뿐입니다. 그래서 비밀번호만 알아내는 방법이 없습니다.

이러한 구조 때문에 Digest 인증에서 사용자가 입력한 아이디와 비밀번호를 검증하는 방법은 $_SERVER['PHP_DIGEST_AUTH'] 변수에 기록된 response 값과 서버에서 동일한 방법으로 생성한 유효한 response 값을 비교하는 것입니다.



[그림 8-16] Digest 인증 검증 구조

위의 검증 구조를 함수로 표현하면 다음과 같습니다.

```
function check_response($data)
{
    global $realm, $user, $pass;

    $A1 = md5($user . ':' . $realm . ':' . $pass);
    $A2 = md5($_SERVER['REQUEST_METHOD'].':'.$data['uri']);
    $valid_response = md5("$A1:$data['nonce']:$data['nc']";
    $valid_response .= ":$data['cnonce']:$data['qop']:$A2");

    return $data['response']==$valid_response?true:false;
}
```

그림에서 자세히 표시하였듯이 3단계로 나뉘어서 response 값이 생성됩니다. 1차적으로 유효한 계정의 아이디와 비밀번호 그리고 realm 값을 가지고 해시값을 생성하고 2차적으로 요청 방식과 URI를 통해서 두 번째 해시값을 생성합니다. 마지막으로 앞서 생성한 두 값과 나머지 항목을 조합하여 마지막 해시값을 생성합니다. 마지막으로 생성된 이 해시값이 Response 값이 되고 웹 브라우저를 통해서 넘어온 $data['response'] 값과 비교하여 같은 경우 로그인에 성공하고 다른 경우에는 로그인에 실패합니다.

여기서 global 키워드를 사용한 이유는 $user와 $pass 그리고 $realm 값이 함수 밖에 존재하기 때문입니다. 그래서 글로벌 변수로 접근하여 사용하도록 했습니다. 이 방법이 싫다면 함수의 인수로

세 값을 입력하게끔 해도 됩니다.

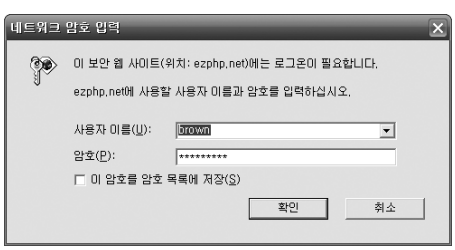
앞서 제작한 패턴 분석 함수와 response 검증 함수를 [예제 8-5]에 적용시켜보면 다음과 같은 소스를 얻을 수 있습니다.

[예제 8-13] Digest 인증 소스 코드

```
<?
  //기본값 설정
  $realm = "ezphp.net";
  $user = "brown";
  $pass = "brown1234";

  if (empty($_SERVER['PHP_AUTH_DIGEST']) || $_COOKIE['login'] != "1") {
    header('HTTP/1.1. 401 Unauthorized');
    header('WWW-Authenticate: Digest realm="' . $realm . '",
    qop="auth",nonce="'. uniqid() .'",opaque="'. md5($realm).'"');

    setcookie("login","1");

    //취소 버튼을 눌렀을 경우
    echo "<meta http-equiv=\"refresh\" content=\"0\">";
    exit;
  }
  else {
    //로그아웃
    if($_GET['logout'] == "1") {
      setcookie("login", "", 0);
      header("location: {$_SERVER['PHP_SELF']}");
      exit;
    }
  }

  //사용자 인증 처리
  if (!check_response( $data = http_digest_parse() ))
  {
    setcookie("login","",0);
    echo "이 페이지는 사용자 인증이 필요합니다.<BR>";
    echo "<a href='{$PHP_SELF}'>로그인</a>하십시오.";
    exit;
  }

  //패턴 분석 함수
  function http_digest_parse()
  {
    $txt = str_replace("=","=>",$_SERVER['PHP_AUTH_DIGEST']);
```

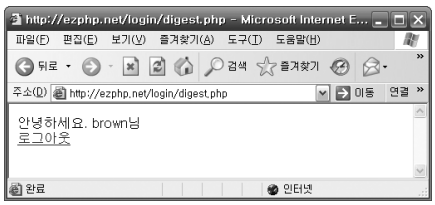
```
40      $txt = "\$data = array($txt);";
41      eval($txt);
42
43      $data[nc] = str_pad($data[nc], 8, "0", STR_PAD_LEFT);
44      $data = str_replace("=>","=",$data);
45
46      return $data;
47    }
48
49    //MD5 비교 검증 함수
50    function check_response($data)
51    {
52      global $realm, $user, $pass;
53
54      $A1 = md5($user . ':' . $realm . ':' . $pass);
55      $A2 = md5($_SERVER['REQUEST_METHOD'].':'.$data['uri']);
56      $valid_response = md5("$A1:$data['nonce']:$data['nc']";
57      $valid_response .= ":$data['cnonce']:$data['qop']:$A2");
58
59      return $data['response']==$valid_response?true:false;
60    }
61  ?>
62
63  안녕하세요. <?=$data['username']?>님<BR>
64  <a href='<?=$_SEVER['PHP_SELF']?>?logout=1'>로그아웃</a>
```

[예제 8-13]를 실행시키면 다음과 같은 새로운 대화상자를 볼 수 있습니다.



[그림 8-17] Digest 인증 로그인 대화상자

이 대화상자에 올바른 아이디와 비밀번호를 입력하고 확인 버튼을 눌러보면 다음과 같은 로그인된 페이지를 볼 수 있습니다.



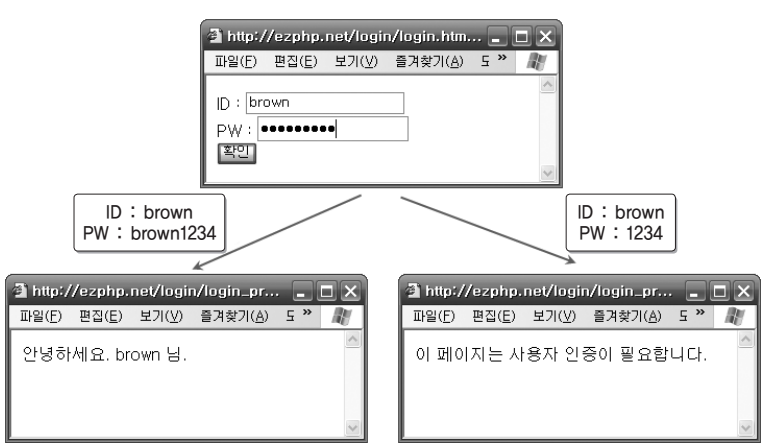
[그림 8-18] Digest 인증에 성공한 경우

[예제 8-13]는 [예제 8-6]의 경우와 마찬가지로 digest_auth.php 파일로 만든 다음에 사용자 인증이 필요한 페이지마다 인클루드하는 방법으로 사용자 인증 상태를 유지할 수 있습니다.

Section 02

Form을 이용한 인증

앞 절에서는 HTTP 인증을 통해서 간단(?)하게 사용자 인증을 처리하는 방법에 대해 배웠습니다. 이 장에서는 가장 일반적인 폼을 이용한 사용자 인증을 다루어 보겠습니다. 폼을 이용한 인증은 다음과 같은 방식으로 이루어집니다.



[그림 8-19] 폼을 이용한 인증 방식

로그인 Form 페이지

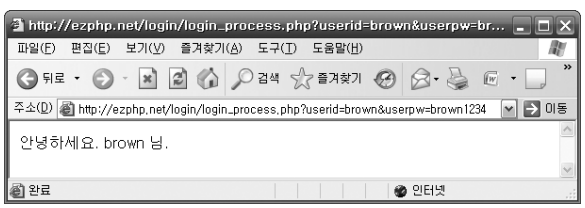
로그인 페이지를 만들고자 간단히 아이디와 비밀번호를 묻는 Form 페이지를 만들어보면 다음과 같

습니다.

[예제 8-14] login.html – 로그인 Form 페이지

```
1  <HTML>
2    <BODY>
3      <FORM METHOD="POST" ACTION="login_process.php">
4      ID : <INPUT TYPE="TEXT" NAME="userid"><BR>
5      PW : <INPUT TYPE="PASSWORD" NAME="userpw"><BR>
6      <INPUT TYPE="SUBMIT" VALUE="확인">
7    </BODY>
8  </HTML>
```

[예제 8-14]에서 중요한 것은 Form 전송방식을 GET이 아닌 POST로 사용하는 것입니다. GET으로 사용하면 로그인을 구현할 수 없다는 뜻이 아니라 다음 그림에서 보는 바와 같이 주소창에 아이디와 비밀번호 정보가 그대로 드러나는 결과를 초래하기 때문입니다.



[그림 8-20] GET을 이용하여 로그인 정보를 전달하는 경우

혼자 컴퓨터를 사용하는 경우라면 특별한 경우를 제외하고 문제가 되지 않을 수 있으나 여럿이 같이 사용하는 컴퓨터라면 주소가 그대로 기록에 남게 되어 다른 누군가가 아이디와 비밀번호를 엿볼 수 있게 됩니다. 따라서 중요한 정보를 전송하고자 할 때는 반드시 POST 방식을 사용합니다.

또한 잊어서는 안 될 것이 비밀번호 입력 상자의 형식을 반드시 PASSWORD로 설정해야 한다는 것입니다. 만약 형식을 PASSWORD가 아닌 TEXT로 해두었다면 다음과 같은 상황이 일어날 수 있습니다.

[그림 8-21] 비밀번호 항목을 일반 글 상자로 설정한 경우

웹 브라우저의 기능 중 하나인 자동 완성 기능 때문에 비밀번호를 TEXT 형식으로 사용하면 위의 그림과 같이 기존에 입력한 비밀번호가 보이는 경우가 생깁니다. 그러나 PASSWORD 형식으로 설정해둔다면 자동 완성 기능이 켜져 있다고 하더라도 해당 값은 기록되지 않습니다. 따라서 비밀번호는 반드시 PASSWORD 형식으로 사용합니다.

로그인 검증 페이지

이제 사용자 인증을 검증하는 페이지를 만들어 봅시다. 앞서 login.html 파일에서 폼 정보를 POST 형식으로 전송했습니다. 이것을 제대로 넘겨받아 아이디와 비밀번호를 비교하면 로그인 검증이 끝납니다.

[예제 8-15] login_process.php – 로그인 검증

```
<?
  if ($_POST['userid']=="brown" && $_POST['userpw']=="brown1234")
  {
    echo "안녕하세요. {$_POST['userid']} 님.<BR>";
  }
  else
  {
    echo "이 페이지는 사용자 인증이 필요합니다.";
    exit;
  }
?>
```

위와 같이 특별히 설명드릴 것도 없을 만큼 아주 간단하게 로그인 검증이 가능합니다.

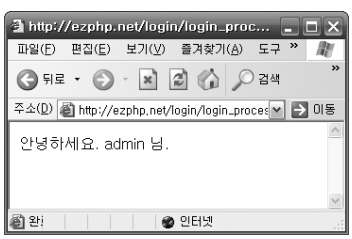
만약 접속 권한을 갖는 아이디가 하나가 아니라 서너 개쯤 되고 비밀번호가 변경될 일이 거의 없다면 데이터베이스를 이용하는 것보다 배열을 이용하는 것이 나을 수도 있습니다.

[예제 8-16] login_process.php – 다중 사용자 로그인 검증

```
<?
  $user = array(
    "brown"=>"brown1234",
    "admin"=>"admin1234",
    "guest"=>"guest1234"
  );

```

```
8     if ($_POST['userid'] && $_POST['userpw']==$user[$_POST['userid']])
9     {
10      echo "안녕하세요. {$_POST['userid']} 님.<BR>";
11    }
12    else
13    {
14      echo "이 페이지는 사용자 인증이 필요합니다.";
15      exit;
16    }
17  ?>
```

[예제 8-16]의 소스를 통해서 admin 아이디로 로그인을 해보면 다음과 같은 결과를 얻을 수 있습니다.



[그림 8-22] 로그인에 성공한 경우

로그인 검증 부분을 살펴보면 사용자 아이디가 입력되어 있고 그 아이디에 대한 비밀번호가 같으면 로그인이 되게 되어 있습니다. 그러나 여기에는 문제가 하나 있는데 사용자의 아이디를 입력하지 않고 공백을 넣게 되면 다음과 같이 로그인이 되어버리는 것입니다.

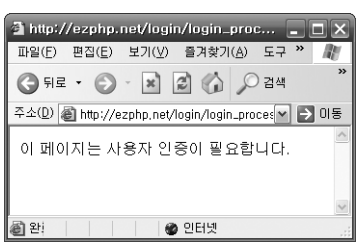
[그림 8-23] 아이디를 입력하지 않고 로그인되는 경우

왜냐하면 입력한 아이디에 대한 비밀번호를 배열에서 검색하였지만 해당 아이디에 대한 비밀번호가 없으므로 빈 값을 반환하게 됩니다. 그런데 사용자는 비밀번호를 입력하지 않아서 비밀번호가 빈 값이기 때문에 결과가 모두 빈 값이어서 로그인이 되는 것입니다. 이러한 현상이 방지하기 위하여 [예제 8-15]의 소스는 다음과 같이 수정되어야 합니다.

[예제 8-17] login_process.php – 수정된 다중 사용자 로그인 검증

```
<?
  $user = array(
    "brown"=>"brown1234",
    "admin"=>"admin1234",
    "guest"=>"guest1234"
  );

  if (array_key_exists($_POST['userid'], $user) &&
  $_POST['userpw']==$user[$_POST['userid']])
  {
    echo "안녕하세요. {$_POST['userid']} 님.<BR>";
  }
  else
  {
    echo "이 페이지는 사용자 인증이 필요합니다.";
    exit;
  }
?>
```

수정된 부분은 아이디 값이 배열에 존재하는지 검사하는 것으로 array_key_exists() 함수는 배열에 해당 키가 존재하는지 검사하는 기능을 합니다. 앞서 시도한 것처럼 빈 공백을 입력하여 로그인을 해보면 로그인이 성공하지 못하는 것을 알 수 있습니다.



[그림 8-24] 아이디와 비밀번호 모두를 검증한 결과

DB를 이용한 로그인 검증 페이지

이제 본격적으로 많은 사용자를 위한 로그인 검증 페이지를 구현해 보겠습니다. 배열을 이용하여 10명, 20명까지는 어떻게 감당할 수도 있겠지만 사용자가 100명, 1000명 이상이면 더 이상 데이터베이스를 사용하지 않고서는 힘들어집니다.

우선 사용자 테이블을 생성해봅시다.

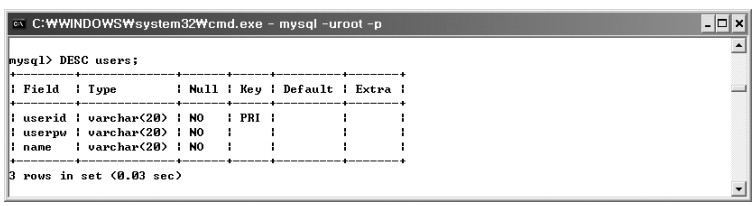
필드	형식	제약	내용
userid	VARCHAR(20)	Primary Key	사용자 아이디
userpw	VARCHAR(20)		사용자 비밀번호
name	VARCHAR(20)		사용자 이름

[표 8-1] 사용자 테이블

로그인 검증을 위해서 간략하게 스키마를 작성했습니다. 작성된 스키마를 통해서 테이블 생성문을 만들어보면 다음과 같습니다.

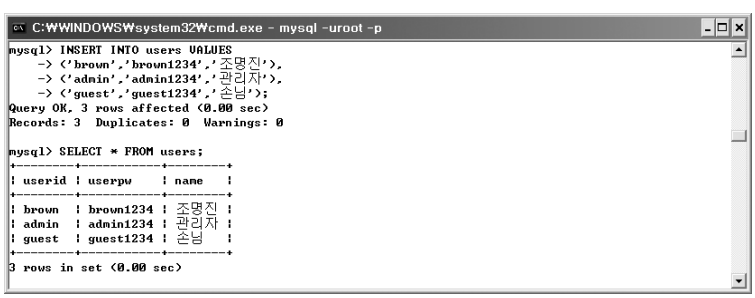
```
CREATE TABLE users (
  userid varchar(20) NOT NULL,
  userpw varchar(20) NOT NULL,
  name varchar(20) NOT NULL,
  PRIMARY KEY (userid)
) ENGINE=MyISAM DEFAULT CHARSET=euckr;
```

테이블 생성문을 통해 데이터베이스에 users 테이블을 생성합니다.



[그림 8-25] 생성된 사용자 테이블 정보

로그인 검증을 위한 테스트 계정을 몇 개 등록합니다.



[그림 8-26] 사용자의 추가

이제 로그인에 대해서 생각해 봅시다. 일반적으로 로그인하는 방법은 두 가지가 있습니다. 하나는 아이디와 비밀번호를 통해서 일치하는 값이 데이터베이스에 있는지 확인하는 것이고 또 하나는 아이디를 통해서 비밀번호를 알아낸 후 사용자가 입력한 값과 비교하는 방법입니다.

첫 번째 방법은 다음과 같은 쿼리로 표현할 수 있습니다.

```
SELECT COUNT(*) FROM users
WHERE userid='$_POST[userid]' && userpw='$_POST[userpw]'
```

두 번째 방법은 다음과 같은 쿼리로 표현할 수 있습니다.

```
SELECT userpw FROM users
WHERE userid='$_POST[userid]'
```

우선 첫 번째 방식을 구현해 보겠습니다.

[예제 8-18] 첫 번째 방식을 이용한 로그인 검증

```
<?
   include "db_info.php";

   //사용자가 입력한 아이디와 비밀번호를 갖는 레코드의 수를 검색한다.
   $query = "SELECT count(*) FROM users WHERE userid = '{$_POST[userid]}'
   and userpw = '{$_POST[userpw]}'";

   $result = mysql_query($query, $conn);
   $row = mysql_fetch_array($result);

   //검색 결과가 1이면 성공적으로 로그인된다.
   if ($row[0] == 1)
   {
     echo "안녕하세요. {$_POST['userid']} 님.<BR>";
   }
   else
   {
     echo "이 페이지는 사용자 인증이 필요합니다.";
     exit;
   }
?>
```

[예제 8-18]을 통해서 brown 아이디로 로그인을 해보면 다음과 같은 결과를 얻습니다.

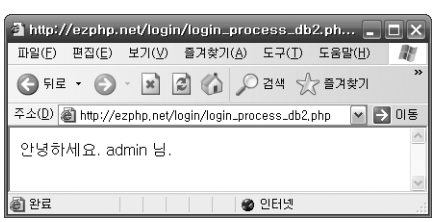
[그림 8-27] 예제 8-18의 로그인 결과

이제 두 번째 방법을 통해서 사용자 인증을 해보도록 하겠습니다.

[예제 8-19] 두 번째 방법을 이용한 로그인 검증

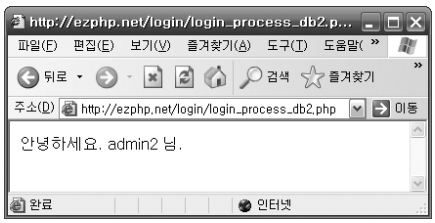
```
<?
  include "db_info.php";

  //사용자가 입력한 아이디의 비밀번호를 검색한다.
  $query = "SELECT userpw FROM users WHERE userid = '{$_POST[userid]}'";
  $result = mysql_query($query, $conn);
  $row = mysql_fetch_array($result);

  //사용자가 입력한 비밀번호와 비교한다.
  if ($row[userpw] == $_POST[userpw])
  {
    echo "안녕하세요. {$_POST['userid']} 님.<BR>";
  }
  else
  {
    echo "이 페이지는 사용자 인증이 필요합니다.";
    exit;
  }
?>
```

[예제 8-19]의 결과는 다음과 같습니다.



[그림 8-28] 예제 8-19의 로그인 결과

그러나 첫 번째 방법과 달리 두 번째 방법에서는 로그인되지 않아야 할 계정으로 로그인될 수 있습니다. 예를 들면 admin2와 같이 데이터베이스에 계정 정보가 없는 경우 배열을 이용한 로그인 검증에서와 마찬가지로 입력한 결과와 데이터베이스를 통한 결과가 모두 빈 값이 되어 로그인되는 경우가 일어날 수 있습니다.



[그림 8-29] 존재하지 않는 아이디를 통한 로그인 회피

따라서 해당 아이디에 대한 데이터베이스 검색 결과가 제대로 있는지를 검증하는 부분을 추가해야 합니다. 다음과 같이 두 가지 방법으로 검증할 수 있습니다.

① mysql_num_rows() 함수를 이용하여 레코드의 수를 센다.

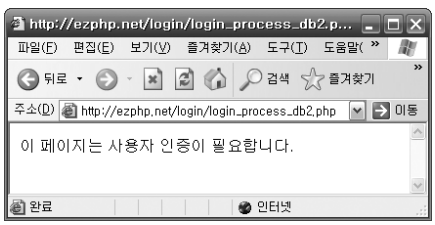
② $row 변수가 빈 값인지 검사한다.

그런데 두 가지 방법 중에서 두 번째 방법이 더 간단하므로 두 번째 방법을 사용하기로 합니다.

[예제 8-20] 수정된 사용자 로그인 검증

```
<?
   include "db_info.php";

   //사용자가 입력한 아이디의 비밀번호를 검색한다.
   $query = "SELECT userpw FROM users WHERE userid = '{$_POST[userid]}'";
   $result = mysql_query($query, $conn);
   $row = mysql_fetch_array($result);

   //사용자가 입력한 비밀번호와 비교한다.
   if (!empty($row) && ($row[userpw] == $_POST[userpw]))
   {
     echo "안녕하세요. {$_POST['userid']} 님.<BR>";
   }
   else
   {
     echo "이 페이지는 사용자 인증이 필요합니다.";
     exit;
   }
?>
```

수정된 소스 코드를 이용하여 거짓으로 로그인을 해보면 다음과 같이 로그인에 실패하는 것을 확인할 수 있습니다.



[그림 8-30] 예제 8-20의 실행결과

쿠키를 이용한 로그인 상태 유지하기

변수, 배열, 데이터베이스를 이용한 로그인 검증에 대해 배웠습니다. 이제 사용자가 로그인을 통해서 인증받은 상태를 지속적으로 유지하는 방법에 대해 알아봅시다. HTTP 프로토콜은 사용자의 상태를 유지하는 기능이 없다고 말씀드렸습니다. 그래서 우리는 앞서 쿠키와 세션을 이용한 사용자의 상태를 유지하는 방법에 대해 배웠습니다.

우선 쿠키를 이용한 로그인 상태 유지하기에 대해 알아보고 다음에 세션을 이용한 방법을 다루도록 하겠습니다.

쿠키를 이용해서 사용자의 로그인 상태를 유지하는 방법의 원리는 매우 간단합니다. 로그인이 성공한 경우 사용자에게 쿠키를 발급해주면 됩니다. 그다음에 로그인이 필요한 모든 페이지에 해당 쿠키를 가졌는지 검증하는 코드를 추가해주면 됩니다.

[예제 8-21] 사용자 인증 후 상태 유지를 위한 쿠키 발급

```
1  <?
2    include "db_info.php";
3
4    //사용자가 입력한 아이디와 비밀번호를 갖는 레코드의 수를 검색한다.
5    $query = "SELECT count(*) FROM users WHERE userid = '{$_POST[userid]}'
6    and userpw = '{$_POST[userpw]}'";
7    $result = mysql_query($query, $conn);
8    $row = mysql_fetch_row($result);
9
10   //검색 결과가 1이면 성공적으로 로그인된다.
11   if ($row[0] == 1)
12   {
13     //로그인 상태를 유지하기 위해 쿠키를 발급한다.
14     setcookie("userid",$_POST['userid'],0,"/");
15     echo "안녕하세요. {$_POST['userid']} 님.<BR>";
16   }
17   else
18   {
```

```
19      echo "이 페이지는 사용자 인증이 필요합니다.";
20      exit;
21    }
22 ?>
```

위의 예제 소스를 통해 정상적으로 로그인을 해보면 헤더 정보를 통해 제대로 쿠키가 설정되고 있음을 알 수 있습니다.

```
HTTP/1.1. 200 OK
Date: Sun, 04 Nov 2007 17:21:06 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: userid=brown; path=/
Content-Length: 25
Connection: close
Content-Type: text/html
```

이제 로그인이 필요한 페이지에서는 이 쿠키를 이용하여 로그인 상태를 유지하면 됩니다.

[예제 8-22] 쿠키를 통해 로그인 상태를 확인

```
1  <?
2    if (empty($_COOKIE[userid]))
3    {
4      die("이 페이지는 사용자 인증이 필요합니다.");
5    }
6  ?>
7
8  성공적으로 로그인 하였습니다.<BR>
9  접속된 아이디는 <?=$_COOKIE[userid]?>입니다.
```

[예제 8-22]을 조금 전 [예제 8-21]에서 로그인한 후 닫지 않은 웹 브라우저를 사용하여 실행해보면 다음과 같은 결과를 얻을 수 있습니다.

[그림 8-31] 로그인을 유지한 경우

위의 결과를 통해 별도의 로그인 없이 로그인 상태를 유지하고 있는 것을 확인할 수 있습니다.

이제 [예제 8-22]의 소스를 간추려서 로그인이 필요한 모든 페이지에 삽입할 수 있게 login_check.php라는 파일을 생성합니다.

[예제 8-23] login_check.php – 로그인 상태 검증

```
1  <?
2    if (empty($_COOKIE[userid])) die("이 페이지는 사용자 인증이 필요합니다.");
3  ?>
```

위의 파일을 인클루드하여 로그인이 필요한 모든 페이지의 최상단에 추가합니다.

[예제 8-24] login_check.php 파일을 include 하여 로그인 상태 검증

```
1  <?
2    include "login_check.php";
3  ?>
4
5  로그인이 필요한 페이지2
```

위의 예제를 실행해보면 다음과 같습니다.

[그림 8-32] 로그인 상태를 유지한 경우

이처럼 쿠키를 이용하여 쉽게 로그인 상태를 유지할 수 있습니다.

여기서 잠깐

쿠키는 사용자의 웹 브라우저에 저장되는 값으로 마음만 먹으면 쉽게 변조를 할 수 있습니다. 그래서 쿠키를 이용한 로그인 상태 검증 방법은 매우 위험합니다. 쿠키를 이용해서 로그인 상태를 유지하려면 반드시 쿠키가 있다 없다만 확인할 것이 아니라 쿠키 값을 암호화하여 발급한 후 검증할 때 쿠키 값이 유효한 값인지 확인하는 방식을 사용해야 합니다. 이 부분에 대해서는 웹 해킹 단원에서 다루도록 하겠습니다.

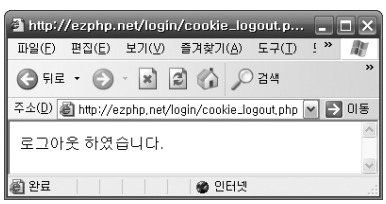
쿠키를 이용한 로그인 상태 해제하기

HTTP 인증에서는 로그아웃하기 위해 여러 가지 번거로운 절차를 사용해야 했습니다. 그러나 쿠키를 이용한 경우에는 쿠키를 삭제하는 방법으로 간단하게 로그아웃을 구현할 수 있습니다.

[예제 8-25] 로그아웃

```
1  <?
2    setcookie("userid","",0,"/");
3  ?>
4
5  로그아웃 하였습니다.<BR>
```

예제를 통해서 로그아웃을 실행해보면 다음과 같은 결과를 얻을 수 있습니다.



[그림 8-33] 로그아웃 결과

실제로 로그아웃이 되었는지 확인하기 위해서 [예제 8-23]인 login_check.php 파일을 실행해 봅시다.

[그림 8-34] 로그아웃 확인 결과

세션을 이용한 로그인 상태 유지하기

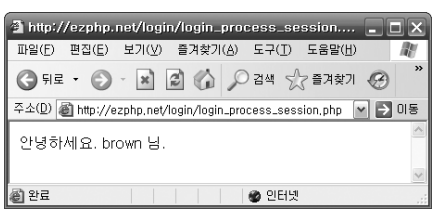
앞서 쿠키는 사용자의 웹 브라우저에 통해 저장되기 때문에 변조의 가능성이 있어서 보안상 위험하다고 했습니다. 그래서 보다 안전한 세션을 이용해서 로그인 상태를 유지해 보겠습니다.

세션을 통해 로그인 상태를 유지하는 방법은 쿠키의 경우와 마찬가지로 로그인에 성공했을 때 세션을 등록해주면 됩니다. 쿠키와 다른 점은 단지 세션 정보가 서버 측에 저장된다는 것밖에 없습니다.

[예제 8-26] 사용자 인증 후 상태 유지를 위한 세션 등록

```
<?
  include "db_info.php";

  //사용자가 입력한 아이디와 비밀번호를 갖는 레코드의 수를 검색한다.
  $query = "SELECT count(*) FROM users WHERE userid = '{$_POST[userid]}'
  and userpw = '{$_POST[userpw]}'";
  $result = mysql_query($query, $conn);
  $row = mysql_fetch_row($result);

  //검색 결과가 1이면 성공적으로 로그인된다.
  if ($row[0] == 1)
  {
    //로그인 상태를 유지하기 위해 세션을 등록한다.
    session_start();
    $_SESSION['userid'] = $_POST['userid'];
    echo "안녕하세요. {$_SESSION['userid']} 님.<BR>";
  }
  else
  {
    echo "이 페이지는 사용자 인증이 필요합니다.";
    exit;
  }
?>
```

[예제 8-26]을 통해 로그인을 시도해보면 다음과 같은 결과를 얻을 수 있습니다.



[그림 8-35] 세션을 이용한 로그인 상태 유지

세션이 제대로 등록되었는지 확인하기 위해서 헤더를 확인해보면 다음과 같이 세션 ID가 발급된 것을 확인할 수 있습니다.

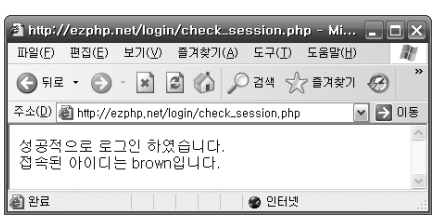
```
HTTP/1.1. 200 OK
Date: Sun, 04 Nov 2007 18:04:32 GMT
Server: Apache
X-Powered-By: PHP/5.1.6
Set-Cookie: PHPSESSID=ce51b5f53343567506b73db7d86e95b2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Content-Length: 25
Connection: close
Content-Type: text/html
```

쿠키에서와 마찬가지로 세션을 통해서 로그인 상태를 검증하는 코드를 작성해 보겠습니다.

[예제 8-27] 세션을 통한 로그인 상태 검증

```
<?
   session_start();
   if (empty($_SESSION[userid]))
   {
      die("이 페이지는 사용자 인증이 필요합니다.");
   }
?>
성공적으로 로그인 하였습니다.<BR>
접속된 아이디는 <?=$_SESSION[userid]?>입니다.
```

[예제 8-26]에서 로그인한 웹 브라우저를 통해서 [예제 8-27]을 실행해보면 결과는 다음과 같습니다.



[그림 8-36] 로그인 상태 유지 검증 결과

쿠키에서와 마찬가지로 [예제 8-27]의 소스를 적당히 간추려서 로그인이 필요한 페이지에서 인클루드해서 쓸 수 있게 만듭니다.

[예제 8-28] check_session.php – 세션을 통한 로그인 상태 검증

```
<?
  session_start();
  if (empty($_SESSION[userid])) die("이 페이지는 사용자 인증이 필요합니다.");
?>
```

이제 이 파일을 쿠키에서와 마찬가지로 로그인이 필요한 모든 페이지 최상단에 삽입하면 로그인 상태를 지속적으로 관리할 수 있습니다.

세션을 이용한 로그인 상태 해제하기

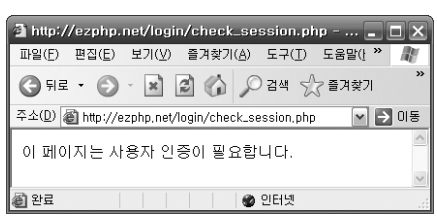
쿠키에서 쿠키를 삭제하면 되었듯이 세션을 삭제하면 바로 로그아웃이 됩니다.

[예제 8-29] 로그아웃

```
<?
  session_start();
  session_unset();
  session_destroy();
?>
로그아웃 하였습니다.
```

위의 예제를 실행해 보면 [그림 8-37]의 결과를 얻을 수 있습니다.

[그림 8-37] 로그아웃의 결과



[그림 8-38] 로그아웃 확인

로그아웃이 제대로 되었는지 확인하기 위해 다시 check_session.php 파일로 접속해 봅시다. 결과는 [그림 8-38]입니다.

Chapter

09

카운터

이 장에서는 첫 번째 실용 예제인 카운터를 만들어 봅니다. 카운터는 홈페이지의 방문자 수를 세기 위한 프로그램으로 쿠키와 세션 그리고 파일과 데이터베이스를 기본 개념으로 하는 매우 간단한 프로그램입니다. 이 4가지 개념에 대한 조합으로 다양한 카운터를 만들 수 있으며 여기서는 가장 기본적인 카운터를 구현해 봅니다.

Section 01

카운터

홈페이지를 운영하다 보면 누구나 자신의 홈페이지에 얼마나 많은 사람이 관심을 가지는지 또한 얼마나 많은 사람이 방문하는지 궁금해지게 마련입니다. 그래서 사람들은 홈페이지에 접속하는 사용자들을 세어보는 프로그램을 생각하게 되었습니다. 바로 이것이 카운터입니다.

자, 그럼 카운터가 어떻게 돌아가는지 알아봅시다.

① 홈페이지에 손님이 방문한다.
② 저장된 값(현재까지의 방문자 수)을 읽어온다.
③ 그 수에 1을 더한다.
④ 더해진 값을 저장한다.
⑤ 그 값을 보여준다.

카운터의 구조는 위와 같습니다. 너무 간단하기에 자신감이 쑥쑥 생기지 않나요?

그렇다면 카운터에 대한 문제점을 파헤쳐 보겠습니다.

Q 홈페이지에 손님이 방문한 것은 어떤 식으로 알 수 있을까요?

어느 장소에 누가 들어오는지를 확인하고 싶다면 그 장소로 들어올 수 있는 모든 곳에 감시카메라를 설치해야 합니다. 홈페이지도 마찬가지로 홈페이지 내의 모든 웹 페이지에 카운터를 넣어두면 될 것입니다. 그런데 모든 웹 페이지에 카운터를 넣어두려면 여간 힘든 일이 아닐 것입니다. 그래서 일반적으로 카운터는 손님이 가장 많이 드나드는 곳(메인 페이지)에 넣어둡니다.

Q 메인 페이지를 여러 번 드나들 경우에는 매번 카운터가 증가되나요?

HOME 버튼을 통해 메인 페이지로 자주 드나들거나 새로 고침을 하는 경우 방문자 수가 계속해서 증가하므로 방문자 수의 의미가 사라지게 됩니다. 물론 홈페이지 운영자를 흥분시키기에는 충분하겠지요. 그래서 쿠키와 세션을 이용하여 여러 번 메인 페이지를 드나들어도 한 번만 카운터를 증가시키게 합니다.

Q 방문자 수는 어디에다 저장하나요?

카운터가 사용하는 방문자 수의 값은 일반적으로 파일로 저장하거나 데이터베이스에 기록합니다.

카운터는 [표 9-1], [표 9-2]와 같이 두 가지 방식으로 분류할 수 있습니다.

	접속자 처리 방식	특징
1	일반 카운터	카운터가 있는 페이지를 다시 읽을 때마다 카운터가 올라감
2	쿠키 카운터	쿠키를 이용해서 브라우저를 닫기 전까지 단 1회만 올라감
3	세션 카운터	세션을 이용해서 세션이 살아 있을 때까지 단 1회만 올라감

[표 9-1] 접속자 처리 방식에 따른 분류

	저장 매체	특징
1	파일 카운터	방문자 수를 파일에 저장
2	DB 카운터	방문자 수를 데이터베이스에 저장

[표 9-2] 저장 매체에 따른 분류

그래서 우리는 먼저 '접속자를 어떻게 처리할 것인가?'와 '방문자 수는 어디에 저장할 것인가?'에 대한 결정을 해야 합니다.

Section 02 접속자를 어떻게 처리할 것인가?

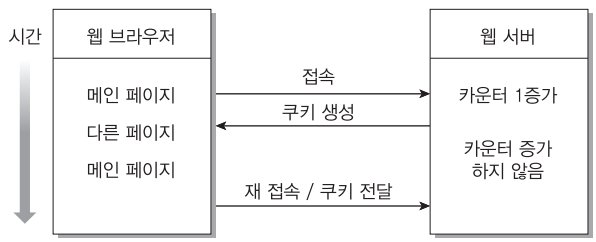
앞서 [표 9-1]에서 정리하였듯이 총 3가지의 접속자 처리 방식이 있습니다. 그중에서 모든 방문에 대해 카운터를 증가시키는 일반 카운터 방식은 새로 고침이나 Home 버튼 등으로 자주 이용되는 페이지는 계속해서 증가하므로 순수 방문자 수를 측정하고자 하는 카운터에는 큰 의미가 없습니다. 이 방식이 전혀 사용되지 않을 것으로 보이지만 실제로는 많이 쓰이고 있습니다. 가장 대표적인 예가 게시판의 게시물에 대한 조회 수입니다. 대부분의 게시판에서 반복된 접속은 조회 수를 상승시킵니다. 이러한 방식은 한 홈페이지내에서 특정 페이지에 대한 접속 수를 파악하기 위해서 유용하게 사용될 수 있습니다. 그러나 홈페이지 전체에 대한 방문자 수를 파악하고자 하는 용도의 카운터에는 사용하기 힘든 방식입니다.

그래서 우리는 쿠키를 사용할 것인지, 세션을 사용할 것인지의 사이에서 고민에 빠집니다. 쿠키와 세션에 대해서 배웠듯이 쿠키와 세션은 정보를 서버에 저장하는지 혹은 클라이언트에 저장하는지에 따라 차이를 나타냅니다. 쿠키는 부담도 없고 편리하지만 사용자에 의해서 수정하거나 삭제하는 것이 가능하다는 단점이 있습니다. 반면에 쿠키보다는 안전하지만 서버 측에 기록되기 때문에 서버에서 세션 관리에 대한 부담을 안아야 한다는 단점이 있습니다.

그런데 카운터는 단순히 접속자의 방문을 확인하는 수준이므로 쿠키를 수정하여도 아무런 영향을

받지 않습니다. 또한 이러한 단순한 작업을 위해서 굳이 세션을 사용하여 서버에 부담을 줄 필요도 없습니다. 그리하여 우리는 카운터를 위해서 쿠키를 사용합니다. 실제로도 대부분의 카운터가 바로 쿠키를 사용하여 만들어져 있습니다.

그렇다면 쿠키는 카운터에서 무슨 일을 하는 것일까요?



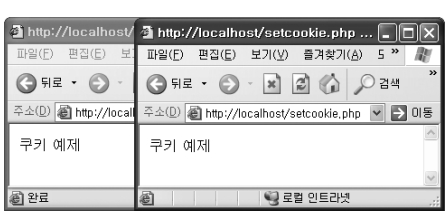
[그림 9-1] 카운터에서 쿠키의 역할

사용자가 홈페이지에 처음 접속하면 쿠키를 손에다 쥐어주고 카운터를 하나 올립니다. 사용자가 다시 메인 페이지에 접속하면 쿠키를 가졌는지 확인을 해서 쿠키를 가지고 있으면 카운터를 무시하고 쿠키가 없으면 새로 접속한 사람으로 인식하고 새로 카운터를 올립니다. 쿠키의 만료 시간을 설정하지 않으면 브라우저가 닫힐 때까지 쿠키가 살아있으므로 브라우저를 닫고 다시 접속하지 않는 한 카운터는 증가하지 않게 됩니다. 이미 접속하고 나서 새로운 브라우저창을 띄워서 접속하는 경우는 어떻게 될까요?

[예제 6-1]의 소스를 수정하여 한번 테스트해 보겠습니다.

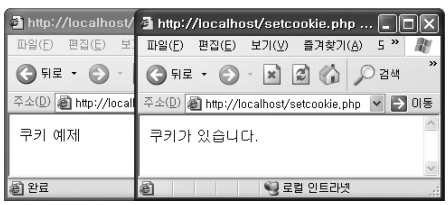
[예제 9-1] 쿠키의 생성 여부 확인

```
1  <?
2     if (isset($_COOKIE['php'])) die("쿠키가 있습니다."); //쿠키가 이미 존재하는 경우
3     setcookie("php");
4     echo "쿠키 예제";
5  ?>
```



[그림 9-2] 만료 시간이 지정되지 않은 경우

새롭게 창을 띄워서 접속하면 쿠키가 적용되지 않는 것을 알 수 있습니다. 그 이유는 만료 시간이 없으므로 쿠키가 파일로 저장되지 않아서입니다. 새로운 웹 브라우저를 통해서 접속하면서 해당 사이트의 쿠키를 찾았으나 쿠키 파일이 없는 관계로 쿠키가 없는 것으로 생각하기 때문입니다. 만약 새로운 접속을 할 때까지 유효한 만료 시간을 지정해 주었다면 쿠키 파일이 생성되면서 다음과 같이 출력됩니다.



[그림 9-3] 만료 시간이 지정되어 쿠키 파일이 생성된 경우

Section 03 어디에 저장할 것인가?

사용자의 접속 처리 방식은 쿠키를 선택하였습니다. 이제 남은 것은 저장 매체를 선택하는 것입니다. 앞서 파일과 데이터베이스 두 가지 종류가 있음을 말씀드렸습니다. 그렇다면 이 둘의 차이는 무엇일까요?

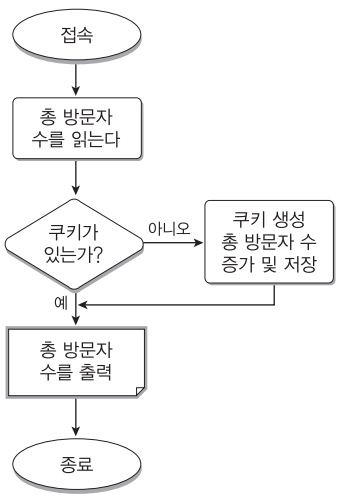
파일을 이용한 카운터는 소규모의 사이트에서만 사용할 수 있습니다. 만약 사이트의 동시 접속자가 그리 많지 않다면 충분히 만족할 만한 성능을 기대할 수 있습니다. 그러나 중, 대규모의 사이트여서 동시에 많은 수의 접속이 빈번히 발생하는 경우라면 파일을 이용한 카운터는 무용지물이 됩니다. 왜냐하면 파일은 여러 사람이 동시에 접근하면 그중에 한 명만 실제로 파일에 접근할 수 있기 때문에 다른 사람은 파일 접근 실패 탓에 에러가 발생합니다. 그래서 비록 여러 명이 접속을 했으나 한 명에 대해서만 카운터가 증가하는 현상이 발생합니다. 그러나 데이터베이스를 이용하면 이러한 동시 접속을 데이터베이스가 관리하여 모든 사람의 접속을 유효하게 처리해 줍니다.

따라서 소규모 사이트의 카운터를 원하거나 간단하게 구현하고자 한다면 파일을 이용하고 중, 대규모 사이트의 카운터를 원한다면 데이터베이스를 선택하면 됩니다. 또한 오늘의 접속자 수, 총 접속자 수 등 통계를 원하거나 좀 더 복잡한 형태의 카운터를 원한다면 소규모의 사이트라 하더라도 데이터베이스를 사용하는 것이 훨씬 좋습니다.

Section 04

파일을 이용한 쿠키 카운터

우선 소규모 사이트에서 사용할 수 있는 "파일을 이용한 쿠키 카운터"를 만들어 봅시다. 기본적인 카운터의 구조에 쿠키를 확인하는 구문만 추가하면 쉽게 카운터를 구현할 수 있습니다.



[그림 9-4] 쿠키 카운터의 순서도

파일 카운터를 만들기 전 선행 작업

파일 카운터를 만들려면 먼저 방문자 수를 저장할 파일을 생성해야 합니다. 파일은 카운터가 들어갈 페이지가 있는 디렉토리에 count.txt라는 이름으로 내용 없는 파일을 생성하면 됩니다. 파일을 생성했다면 웹 서버에서 파일을 읽고 쓸 수 있게끔 권한을 설정해 주어야 합니다.

파일의 권한 설정은 웹 서버의 운영체제에 따라서 다르므로 운영체제에 맞는 방법을 사용합니다.

리눅스의 경우 권한 설정

웹 서버로 유닉스나 리눅스 계열의 운영체제를 사용한다면 다음과 같은 두 방법을 통해서 권한 설정을 할 수 있습니다.

① 텔넷이나 SSH를 이용한 방법

```
chmod 777 count.txt
```

chmod 명령은 함수 레퍼런스에서 이미 다루었던 chmod() 함수와 동일하게 파일의 권한을 설

정하는 명령입니다. 777은 모든 사용자에게 읽고 쓰고 실행할 수 있는 권한을 부여하겠다는 뜻으로 즉, count.txt 파일을 누구나 마음대로 사용할 수 있게 해주는 명령입니다. 이렇게 설정을 하면 웹 서버가 언제든지 count.txt 파일에 접근하여 파일의 내용을 읽거나 수정하는 등의 작업을 할 수 있게 됩니다.

여기서 잠깐

파일에 대한 권한은 소유자, 그룹, 그 외 사용자로 구분됩니다. 770 또는 775는 파일의 소유자와 소유자의 그룹에게는 모든 권한을 부여하였으나 그 외의 사용자에게는 권한을 전혀 부여하지 않거나(770) 읽기와 실행하기에 대한 권한(775)만을 부여합니다. 그런데 웹 서버는 일반적으로 그 외의 사용자에 해당하므로 웹 서버에서는 파일에 전혀 접근할 수 없거나 그 값을 읽어 올 수만 있고 수정할 수 없는 상태가 될 수 있습니다. 따라서 반드시 777로 설정을 해주어야 합니다.

② FTP를 이용한 방법

웹 서버에서 FTP 서버를 같이 운영하고 있다면 FTP 프로그램을 통해서 파일의 권한을 설정할 수 있습니다. 모든 FTP 프로그램이 이 기능을 지원하지는 않으니 여러분이 사용하는 FTP가 파일 권한을 설정할 수 있는 기능을 제공하는지 꼭 확인해 보기 바랍니다. 파일 권한 설정 기능을 제공하는 FTP 프로그램 중에서 알 FTP의 예를 들어서 사용 방법을 알려 드리겠습니다.

알 FTP를 통해서 FTP 서버에 접속해보면 다음과 같은 파일 목록을 볼 수 있습니다. count.txt 파일을 찾아서 마우스 오른쪽 버튼을 클릭하면 다음과 같이 권한 설정 메뉴를 찾을 수 있습니다.

[그림 9-5] 알 FTP를 이용한 파일 권한 설정

권한 설정을 클릭하면 다음과 같이 파일 권한 변경창이 뜹니다.

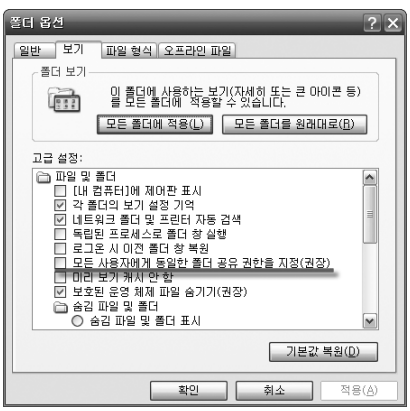
[그림 9-6] 파일의 권한 변경

현재 Mode 값을 777로 수정하거나 모든 체크 상자를 클릭하면 모든 사용자에 대해서 count.txt 파일을 읽고 쓰게 할 수 있습니다.

윈도우의 경우 권한 설정

웹 서버의 운영체제가 윈도우를 사용하고 있고 파일 시스템으로 NTFS를 사용하는 경우에는 리눅스와 마찬가지로 권한 설정이 필요합니다. 반면 FAT32인 경우에는 권한의 의미가 없으므로 설정할 필요가 없습니다.

윈도우 탐색기 → 도구 → 폴더옵션 → 보기 → "모든 사용자에게 동일한 폴더 공유 권한을 지정"을 체크 해제합니다.



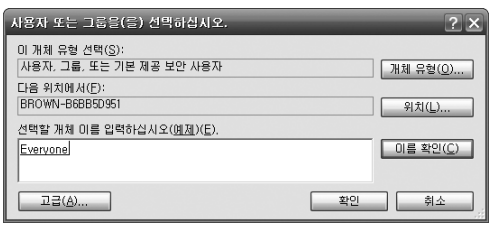
[그림 9-7] 윈도우에서 파일 권한 설정하기

기본적으로 "모든 사용자에게 동일한 폴더 공유 권한을 지정" 항목은 체크되어 있으므로 그림에서 보이는 바와 같이 체크 상태를 해제해 줍니다. 이 작업을 하지 않으면 다음 과정에서 필요한 보안 탭이 보이지 않으므로 반드시 이 과정을 실행해야 합니다.

그런 다음 count.txt 파일을 선택하고 마우스 오른쪽 버튼의 속성 → 보안 탭을 선택합니다.

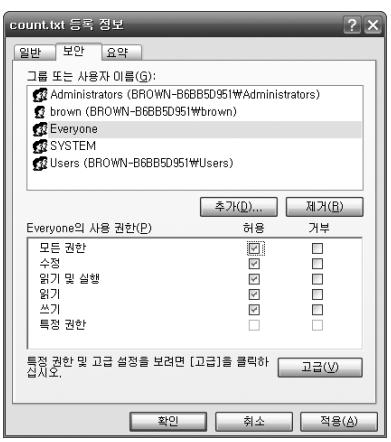
[그림 9-8] count.txt 파일의 보안 설정 창

그다음 추가 버튼을 클릭하여 다음과 같이 Everyone 사용자를 추가합니다.



[그림 9-9] Everyone 사용자를 추가

Everyone을 입력하고 이름 확인 버튼을 클릭하면 밑줄이 그어지는 것을 확인할 수 있습니다. 밑줄이 그어지면 확인 버튼을 클릭하여 Everyone 사용자를 추가합니다.



[그림 9-10] Everyone 사용자에게 모든 권한 부여

그룹 또는 사용자 이름에 Everyone 사용자가 추가되었으면 Everyone 사용자를 선택하고 사용 권한으로 "모든 권한" 항목의 허용 체크 상자를 클릭합니다. 마지막으로 확인 버튼을 클릭하면 파일 권한 설정이 마무리됩니다.

소스 코드

선행 작업이 완료되었으니 본격적으로 카운터 프로그램을 구현해 보겠습니다. 카운터 소스는 몇 줄 안 되는 작은 프로그램이므로 쉽게 이해할 수 있을 것입니다.

[예제 9-2] 파일을 이용한 쿠키 카운터

```
1  <?
2     $count = file("count.txt"); // 파일에서 총 방문자 수를 읽어옴
3     $count = trim($count[0]); // 가져온 값에서 빈 공간을 없앰
4
5     if (!isset($_COOKIE['visit'])) { // 쿠키가 설정되지 않았으면
6        $count++;
7        $fp = fopen("count.txt", "w"); // 파일을 쓰기 모드로 열고
8        fwrite($fp, $count); // 파일에다 $count 값을 저장
9        fclose($fp); // 파일을 닫음
10
11       setcookie("visit"); // 쿠키를 생성
12    }
13    echo $count; // 총 방문자 수를 출력
14 ?>
```

2행 `$count = file("count.txt");`

file() 함수는 인자에 지정된 파일을 배열로 읽어들입니다. 파일에는 한 줄로 방문자 수에 대한 정보가 저장되므로 첫 번째 줄만 의미 있는 값이 됩니다. 그래서 $count[0]을 통해서 총 방문자 수 정보를 읽어올 수 있습니다.

3행 `$count = trim ($count[0]);`

trim() 함수는 변수의 처음과 끝에 있는 빈 공백을 제거해주는 함수입니다. 파일을 통해 읽어들인 $count[0] 변수에 혹시나 빈 공백이 포함될 경우를 대비하여 앞뒤의 공백을 제거해 줍니다. trim() 함수를 처리한 결과를 다시 $count 변수에 대입하면 배열이었던 $count 변수는 배열이 아닌 문자열 변수로 변합니다.

5행 `if (!isset($_COOKIE['visit']))`

isset() 함수는 변수가 존재하는지를 확인하는 함수입니다. 만약 쿠키가 생성되어 있지 않다면

$_COOKIE['visit'] 변수가 자동으로 설정되지 않기 때문에 FALSE를 반환하고, 쿠키가 있다면 TRUE를 반환합니다. 그러나 느낌표(!)가 있기 때문에 쿠키가 없는 경우에 if 구문이 실행됩니다.

6행 `$count++;`

새로운 방문이면 방문자 수를 증가시킵니다.

7행 `$fp=fopen("count.txt","w");`

변경된 총 방문자 수 값을 저장하기 위해 다시 count.txt 파일을 엽니다. 파일을 열 때 쓰기 모드로 열어서 파일에 쓰기가 가능하게끔 합니다. 파일이 열리면 파일의 핸들을 변수에 저장하고 이를 통해 파일을 쓰거나 읽거나 합니다.

8행 `fwrite( $fp , $count );`

fopen() 함수를 통해 얻은 파일 핸들을 이용하여 파일에 증가시킨 총 방문자 수 정보를 기록합니다.

9행 `fclose( $fp );`

방문자 수를 수정하기 위해서 열어둔 파일을 닫습니다. 파일을 fopen() 함수를 이용해 열었으면 반드시 fclose() 함수를 이용하여 닫아 주도록 합니다. 실제로 프로그램이 종료되면서 자동으로 열린 파일이 닫히지만 프로그램이 종료되기 전까지는 파일이 닫히지 않으므로 그동안 다른 사용자가 파일을 사용할 수 없게 됩니다.

11행 `setcookie("visit");`

이제 visit이라는 이름의 쿠키를 생성합니다. 이 쿠키를 이용하여 다시 접속하였을 때 카운터를 증가시키지 않도록 합니다. 쿠키의 만료 시간을 지정하지 않았으므로 이 쿠키는 해당 브라우저가 종료되면 같이 소멸하게 됩니다. 따라서 웹 브라우저를 닫은 후에 다시 접속하면 새로운 접속으로 인정하여 방문자 수가 증가합니다.

13행 `echo $count;`

이제 마지막으로 총 방문자 수를 출력해 줍니다. 만약 새로운 접속이라면 증가된 방문자 수를 볼 수 있습니다.

카운터의 구조와 실제 어떻게 구현할 수 있는지를 알아보았습니다. 머릿속에 쏙쏙 이해가 되시나요?

Section 05

파일 핸들

카운터를 만들어 보면서 파일을 다루는 간단한 방법을 알게 되었습니다. 그런데 여기서 한 가지 의문이 생깁니다. 그냥 fwrite(파일이름, 저장값)와 같은 형식으로 만들었으면 간단할 거라 생각되는데 왜 핸들이 어쩌고 파일을 열고 닫고 이 난리 법석을 피워야 하는지 통 알 수가 없습니다. 자 그럼 상상의 날개를 펴 볼까요?

파일 핸들링에 대한 개념적 이해

책상에 일기장 한 권이 놓여 있습니다. 일기장에다가 오늘의 일기를 쓰려고 합니다. "오늘 순이를 만났다." 식으로 말입니다. (제가 좀 단순합니다.) 일기를 쓸 때 우선 이 일기장이 나의 일기장인지 남의 일기장인지 확인해야 합니다. 대개 별도로 이리저리 뒤적거리지 않아도 자기 일기장이라면 바로 알 수 있을 것입니다. (앞서 chmod 명령어를 통해 내가 읽고 쓸 수 있는 일기장을 만든 것과 같습니다.)

자 그럼 이게 내 일기장이라는 건 확실하네요. 그럼 일기를 써봅시다.

- 우리의 소망대로 일기장을 딱 잡고 펜으로 씁니다. (fwrite를 위처럼 하고자 할 경우)
- 이런! 일기장 표지에다 일기를 써버렸군요. (실제로 안 된다는 말입니다.)
- 에고! 그럼 표지를 넘겨야겠군요. (fopen을 합니다.)
- 일기를 쓸 것이기 때문에 손에는 펜을 들고 있어야겠지요? (파일 모드는 w)
- 자 원하는 페이지를 열어 일기를 씁니다.
- "오늘 순이를"까지 썼는데 갑자기 순이 사진이 보고 싶어졌습니다. 일기 쓰던 걸 멈추고 서랍에서 옛날 일기장을 꺼냅니다. 옛날 일기장에 순이 사진을 꽂아 두었거든요. 일기장을 펴고 순이 사진을 보면서 느끼한 미소를 짓습니다. 아~ 감정이 북받치는군요. 일기가 왠지 잘 써질 것 같습니다. 다시 일기를 씁니다.
- 근데 일기장이 두 개가 있네요. 옛날 일기장과 요즘 쓰는 일기장. 우리가 이 일기장을 구분할 수 있는 것은 각각의 일기장에 대한 기억을 가지고 있기 때문입니다. 그렇다고 일기장의 모든 내용을 기억해서 둘을 구분하는 건 아닙니다. 단지 일기장의 모양으로도 일기장 표지의 그림으로도 쉽게 구분할 수 있는 것입니다. 그리고 내가 어디쯤에 일기를 쓰고 있었다 하는 기억

도 있을 것입니다. 이 기억이란 것이 핸들입니다. 여러 파일이 있을 때 하나의 파일을 구분 지을 수 있는 값입니다. 핸들을 이용해서 현재 사용하는 일기장을 찾았습니다. 그럼 마저 일기를 씁시다.

"만났다."

이제 일기를 다 썼습니다. 그리고 일기장을 덮습니다 (fclose). 나중에 일기를 더 쓰고 싶을지도 모르기 때문에 일기장을 안 덮을 수도 있습니다. 하지만 나중에 바로 또 쓸 것이 아니라면 닫는 것이 바람직합니다.

카운터 소스를 보면 파일 하나를 사용하여 방문자 수를 체크합니다. 그런데 파일은 하나뿐인데 사용자는 여럿입니다. 파일을 노트라고 생각한다면 먼저 접속한 사람이 노트를 집어서 거기에 적힌 숫자에 1을 더하고 나서 노트를 다시 사람들이 쓸 수 있게 있던 자리에 내려놓아야 할 것입니다. 그래야 다음번에 접속한 사람이 다시 노트에 자신의 흔적을 남길 수 있을 테니까요. 그런데 이때 먼저 방문한 사람이 노트를 내려놓지 않고 계속 들고 있다면 뒤에 접속한 사람은 당장 노트를 사용할 수 없으니 자신의 흔적을 남기지 못할 것입니다. 그래서 꼭 파일을 쓴 후에는 반드시 다른 사람이 사용 가능하게 닫아주는 버릇을 들여야 합니다.

파일 핸들은 파일 포인터를 말합니다. 그래서 파일 포인터의 이니셜을 따서 $fp라는 이름의 변수로 많이 사용합니다. 파일 포인터는 예를 들면 메모장에서 깜박거리고 있는 커서 같은 것입니다.

파일이 열리고 나서부터 열린 파일의 어디서부터 글을 읽을 것인지 혹은 어디서부터 글을 쓸 것인지 그 위치를 항상 가리키는 것이 파일 포인터라 할 수 있습니다. 그래서 파일을 쓰거나 읽거나 할 때 파일 포인터를 사용하고, 파일을 닫을 때도 파일 포인터를 제거함으로써 마감합니다.

Chapter

10

방명록

개인용 미니홈피를 제공하는 싸이월드가 인기를 끌면서 방명록에 새 글이 올라왔는지 확인하는 버릇을 가진 사람들이 많아졌습니다. 심지어 네이트온 메신저에서는 방명록에 새로운 글이 등록되면 실시간으로 알려주는 기능이 있을 정도입니다. 최근에 미니홈피와 같이 개인용 홈페이지를 쉽게 제작해주고 또한 쉽게 관리하게 도와주는 사이트가 늘어남에 따라 방문해놓고 방명록에 글을 남기지 않으면 예의 없는 사람이란 말이 생길 정도로 방명록은 우리의 가까이에 다가와 있습니다.

방명록은 행사장이나 식장과 같은 곳에서 참석한 사람의 이름을 기록하는 공책을 말합니다. 그런데 인터넷 문화가 발전하면서 이 방명록이 개인 홈페이지를 방문하고 그 감상이나 홈페이지 운영자에게 남기고 싶은 말을 기록해두는 용도로 사용되기 시작했습니다. 홈페이지 운영자는 많은 사람이 방문하고 또한 자신의 홈페이지에 관심을 표하는 것을 굉장한 기쁨으로 느낍니다. 그래서 방명록을 만들어 두어 방문자로 하여금 방문의 흔적을 남기기를 바라는 것입니다. 대부분의 운영자는 공손하게 "한 말씀 남겨주고 가세요."라고 적어두지만 가끔 "안 적고 가면 주거~"라며 조금 과격한 협박을 하는 홈페이지도 종종 눈에 띄곤 합니다.

개인 홈페이지를 구성하면서 없어서는 안 될 방명록을 이번 기회에 만들어 보도록 합시다.

Section 01

방명록의 구조

방명록은 수많은 사람이 글을 남기게 될 노트입니다. 앞에서는 정보를 저장하려고 파일을 이용했지만 파일을 이용하는 방법에는 한계가 있다고 했습니다. 그래서 파일의 한계를 극복하기 위해 데이터베이스라는 것을 배웠습니다. 방명록은 바로 데이터베이스를 본격적으로 활용하는 첫 번째 실용 예제가 될 것입니다.

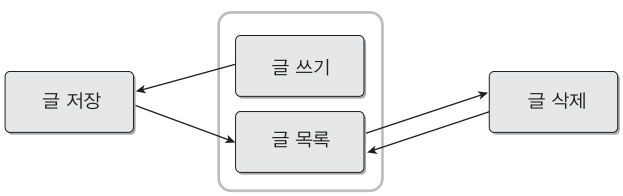
방명록을 만들기에 앞서 방명록이 어떤 식으로 구성되어 있는지 먼저 알아보도록 합시다. 방명록은 다양하게 구성할 수 있으나 여기에서는 다음과 같이 세 가지 요소로 구성하고자 합니다.

① 글 쓰기와 글 목록

② 글 저장

③ 글 삭제

각각은 하나의 파일로 구성되며 세 가지 요소 간의 관계는 다음과 같습니다.



[그림 10-1] 방명록의 구조

글 쓰기와 글 목록은 하나의 파일에 공존합니다. 즉, 글 쓰기 폼과 글 목록이 같은 페이지에 출력되며 글을 입력하고 확인 버튼을 클릭하면 글 저장 페이지로 이동하여 데이터베이스에 글이 저장됩니다. 글이 저장된 후에는 다시 글 목록 페이지로 돌아오게 하여 새롭게 쓰인 글이 목록에 제대로 출력되는지 확인합니다.

글 목록에서는 일반적인 게시판과 달리 글의 내용을 모두 출력합니다. 일반적인 게시판은 글의 제목을 먼저 목록으로 보여주고 제목을 클릭하면 글의 내용을 자세히 볼 수 있는 페이지로 이동하지만 방명록은 본문의 내용이 매우 짧아서 제목을 따로 입력하지 않고 본문의 내용만 입력합니다. 따라서 방명록의 목록에서는 다음과 같이 글의 정보가 모두 출력됩니다.

[그림 10-2] 방명록의 목록

글 목록에서는 del 링크를 이용하여 해당 글을 삭제할 수 있도록 합니다. 방명록은 대체로 짧은 글을 등록하기 때문에 글을 수정하는 기능이 없는 경우가 많습니다. 글을 수정하지 말고 해당 글을 지우고 다시 글을 등록하라는 의미입니다. 만약 방명록에 글을 수정하는 기능을 추가하고 싶다면 다음에 배우게 될 "11장. 게시판 만들기"를 참고하면 쉽게 만들 수 있습니다.

글 삭제를 할 때에는 글쓴이와 홈페이지 운영자만이 글을 지울 수 있습니다. 다른 누군가가 내 글을 지우는 것을 방지하기 위해서 글을 지울 때 반드시 비밀번호를 검증하여 삭제하도록 해야 합니다. 올바른 비밀번호가 입력되면 글을 삭제하고 다시 글 목록으로 되돌아와서 해당 글이 실제로 삭제된 것을 확인합니다.

Section 02 방명록 테이블 설계

방명록을 만들기 위해서 파일이 아닌 데이터베이스를 사용하기로 했습니다. 방명록 정보를 저장하기 위한 테이블을 설계해 봅시다.

우선 방명록에 저장되어야 할 정보를 정리해야 합니다. 방명록에는 어떠한 정보를 저장해야 할까요? 우선 필수적인 정보를 생각해보면 글쓴이의 이름, 글쓴이를 구별할 수 있는 비밀번호, 글의 내용 그리고 글이 등록된 날짜와 시간 정보가 있습니다. 이외에 글쓴이의 정보를 보다 많이 남기려고 이메일이나 홈페이지 주소 등을 남기도록 할 수도 있습니다. 그러나 너무 많은 정보를 입력하게 하면 방문자가 글을 남기는 것을 귀찮아 할 경우가 생길 수 있으므로 방명록은 최대한 간략한 정보를 남길 수 있도록 합니다. 그래서 필수적인 부분만 고려하여 표로 정리해보면 다음과 같습니다.

항목	내용
name	글쓴이의 이름
pass	글쓴이의 비밀번호
content	글의 내용
wdate	글이 등록된 날짜와 시간

[표 10-1] 방명록의 필요 항목

그런데 모든 테이블에는 한 튜플을 구분할 수 있는 키(Key)가 있어야 합니다. 그러나 위와 같이 작성된 테이블에서 키가 존재하지 않기 때문에 키가 되는 필드를 새로 추가해야 합니다. 이에 글을 구분 지을 수 있는 글 번호를 추가하도록 합니다.

항목	내용
id	글 번호, 키
name	글쓴이의 이름
pass	글쓴이의 비밀번호
content	글의 내용
wdate	글이 등록된 날짜와 시간

[표 10-2] 글 번호(키)를 추가한 방명록의 필요 항목

위와 같이 글 번호를 추가하여 글을 구분 지을 수 있게 되었습니다. 그러나 글 번호가 중복될 수 있다면 id는 결코 키가 될 수 없습니다. 따라서 id 값은 중복된 값을 입력할 수 없게끔 처리해야 합니다. 키에 대한 처리는 조금 미루도록 하고 우선 각 필드의 형식을 지정해 보도록 하겠습니다.

항목	형식	NULL	내용
id	int(11)	NOT NULL	글 번호, 키
name	varchar(20)	NOT NULL	글쓴이의 이름
pass	varchar(20)	NOT NULL	글쓴이의 비밀번호
content	text	NOT NULL	글의 내용
wdate	timestamp	NOT NULL	글이 등록된 날짜와 시간

[표 10-3] 방명록 필요 항목의 상세 정보

id는 글을 구분 짓는 번호이니 정수형으로 1씩 증가하는 값을 갖도록 합니다. name과 pass는 이름과 비밀번호이므로 문자열로 설정하고 또한 20자면 충분히 표현할 수 있을 것으로 생각합니다. 글의 내용인 content의 경우 방명록의 특성에 따라서 매우 짧은 내용이 등록될 것으로 예상되므로 text 정도로 설정합니다. tinytext는 255자밖에 저장이 되지 않아서 간혹 글이 잘리게 될 경우가 발생할 수 있습니다. mediumtext는 너무나 크고 65535자인 text를 사용하면 충분할 것으로 생각합니다. (실제로는 대부분의 글들이 2000자도 안 될 가능성이 많습니다.) 마지막으로 wdate

는 날짜와 시간을 저장하는 값으로 datetime, timestamp, int형 등을 사용할 수 있으나 여기서는 timestamp 값을 사용합니다.

키와 기타 제약 사항을 모두 정리하면 다음과 같습니다.

항목	형식	NULL	Key	추가 속성
id	int(11)	NOT NULL	PRIMARY KEY	auto_increment
name	varchar(20)	NOT NULL		
pass	varchar(20)	NOT NULL		
content	text	NOT NULL		
wdate	timestamp	NOT NULL		CURRENT_TIMESTAMP

[표 10-4] 키와 제약 사항을 표기한 방명록 필요 항목

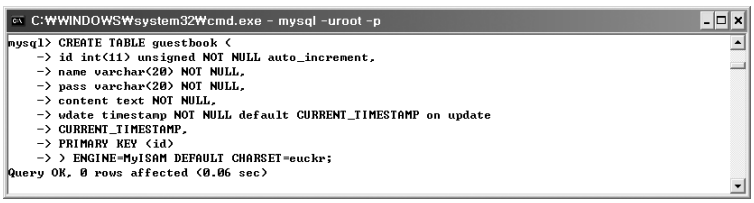
id 값은 auto_increment 속성을 추가하여 자동으로 1씩 증가하는 값을 부여받게 설정합니다. 그리고 wdate는 기본값으로 현재의 timestamp 값을 자동으로 입력하게끔 설정합니다. 이를 위해서는 default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP라는 문구를 추가해야 합니다.

Section 03 방명록 테이블 생성하기

앞서 설계한 테이블에 따라서 테이블 생성문을 작성해보면 다음과 같습니다.

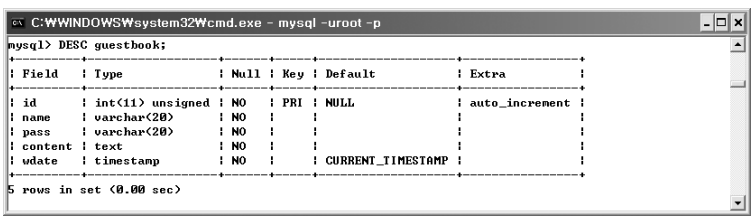
```
CREATE TABLE guestbook (
  id int(11) unsigned NOT NULL auto_increment,
  name varchar(20) NOT NULL,
  pass varchar(20) NOT NULL,
  content text NOT NULL,
  wdate timestamp NOT NULL default CURRENT_TIMESTAMP on update
        CURRENT_TIMESTAMP,
  PRIMARY KEY (id)
) ENGINE=MyISAM DEFAULT CHARSET=euckr;
```

작성된 테이블 생성문을 데이터베이스에 접속하여 입력하면 guestbook이라는 테이블이 생성됩니다.



[그림 10-3] 테이블 생성

desc 명령을 통해서 생성된 테이블을 확인해보면 다음과 같습니다.



[그림 10-4] 생성된 테이블 정보

Section 04

방명록 글 쓰기

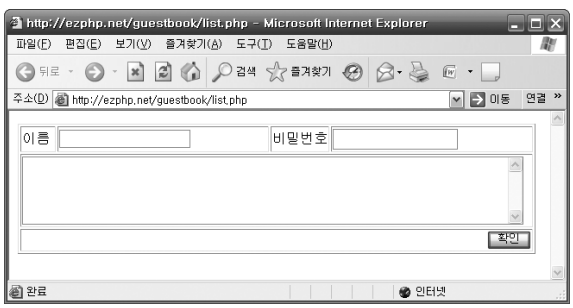
이제 테이블이 생성되었으니 본격적으로 방명록을 작성합시다. 우선 방명록의 글을 남기기 위한 글 쓰기 폼을 작성합니다. 글 쓰기 폼은 이미 폼 다루기에서 만들어 보았으므로 쉽게 만들 수 있을 것으로 생각합니다. 디자인은 일단 신경 쓰지 말고 필수 항목을 입력할 수 있는 폼을 만들어 봅시다.

[예제 10-1] list.php – 글 쓰기 페이지

```
1  <FORM ACTION="insert.php" METHOD="POST">
2  <TABLE BORDER=1 WIDTH=600>
3    <TR>
4      <TD>이름</TD><TD><INPUT TYPE="TEXT" NAME="name"></TD>
5      <TD>비밀번호</TD><TD><INPUT TYPE="PASSWORD" NAME="pass"></TD>
6    </TR>
7    <TR>
8      <TD COLSPAN=4>
9        <TEXTAREA NAME="content" COLS=80 ROWS=5></TEXTAREA>
```

```
10      </TD>
11    </TR>
12    <TR>
13      <TD COLSPAN=4 align=right><INPUT TYPE="SUBMIT" VALUE=" 확인 "></TD>
14    </TR>
15 </TABLE>
```

이름과 비밀번호 그리고 글의 내용을 입력하게 합니다. 각 폼의 이름은 편의를 위해서 데이터베이스 필드 이름과 동일하게 사용하겠습니다. 여기서 암호는 반드시 비밀번호 타입으로 설정하고 또한 SUBMIT 버튼을 잊지 않고 만들어 줍니다.



[그림 10-5] 방명록 글쓰기 폼

Section 05

방명록 글 저장

글 쓰기 폼이 완성되었으니 입력한 글을 저장할 수 있는 글 저장 페이지를 만들어 봅시다. 입력된 글을 데이터베이스에 저장하려면 우선 데이터베이스에 연결하고 자신이 사용하는 데이터베이스를 선택해야 합니다.

```
$conn = mysql_connect("localhost", "사용자아이디", "패스워드");
mysql_select_db("데이터베이스이름", $conn);
```

또한 MySQL4.1 버전부터 발생하는 문자셋 문제를 해결하기 위해서 문자셋을 euckr로 설정하는 쿼리를 먼저 실행합니다.

```
mysql_query("set names euckr");
```

이제 입력된 값을 데이터베이스에 저장하기 위해서 INSERT 쿼리를 작성하면 됩니다. id와 wdate 값은 자동으로 데이터베이스가 알아서 넣어주므로 우리가 입력해야 하는 값은 name, pass, content 이 세 가지 항목입니다. 따라서 INSERT 문을 작성해보면 다음과 같습니다.

```
INSERT INTO guestbook (name, pass, content)
VALUES ('$_POST[name]', '$_POST[pass]', '$_POST[content]');
```

이를 쿼리 변수에 저장합니다.

```
$query = "INSERT INTO guestbook (name, pass, content) ";
$query .= " VALUES ('$_POST[name]', '$_POST[pass]',
          '$_POST[content]')";
```

데이터베이스에 쿼리를 전송합니다.

```
$result = mysql_query($query, $conn);
```

데이터베이스에 올바르게 쿼리가 전송되고 글이 제대로 등록되면 $result 값으로 0이 아닌 값이 리턴됩니다. 만약 0이 리턴되면 쿼리 전송에 실패한 것입니다.

글이 등록되었으니 다시 글 목록(글 쓰기) 페이지로 돌아갑니다.

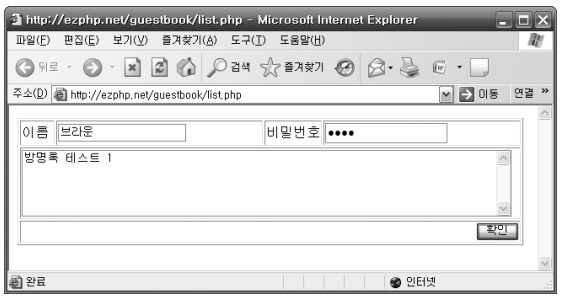
```
<script> alert("글이 등록되었습니다."); location.href="list.php"; </script>
```

모든 소스를 정리해보면 다음과 같습니다.

[예제 10-2] insert.php – 글 저장하기

```
1  <?
2    $conn = mysql_connect("localhost", "사용자아이디", "패스워드");
3    mysql_select_db("데이터베이스이름", $conn);
4    mysql_query("set names euckr");
5
6    $query = "INSERT INTO guestbook (name, pass, content) ";
7    $query .= " VALUES ('$_POST[name]', '$_POST[pass]', '$_POST[content]')";
8    $result = mysql_query($query, $conn);
9  ?>
10
11 <script>
12 alert("글이 등록되었습니다.");
13 location.href="list.php";
14 </script>
```

글 저장하기가 완성되었으니 실제로 글 쓰기 폼과 글 저장하기를 통해서 글을 작성합니다.



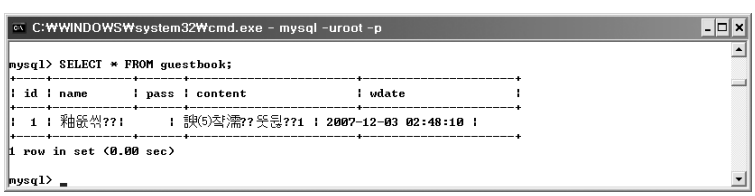
[그림 10-6] 글쓰기 폼에 테스트 글 입력

위의 그림과 같이 간단히 글을 입력하고 확인 버튼을 클릭해보면 다음과 같은 메시지를 볼 수 있습니다.

[그림 10-7] 데이터베이스에 입력된 글을 등록

실제로 제대로 등록되었는지와 상관없이 글이 등록되었다는 메시지가 출력되므로 정말로 글이 등록되었는지 확인을 합시다.

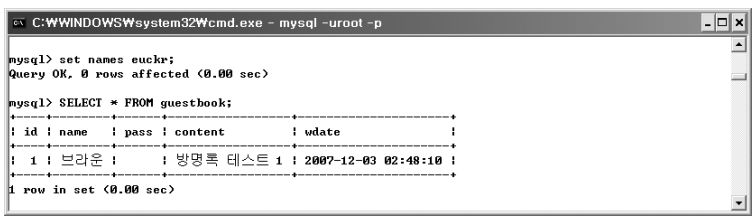
MySQL 콘솔에 로그인한 후 SELECT 문을 이용하여 글이 등록되어 있는지 확인합니다.



[그림 10-8] 데이터베이스에 글이 등록되어 있는지 확인

그런데 글자가 깨져서 보이는 것을 확인할 수 있습니다. 앞에서 문자셋 문제를 해결하기 위해서 set names euckr; 명령을 입력해주는 이유가 바로 여기에 있습니다. 문자열 기본 세팅이 한국어가 아

니므로 위와 같이 글자가 깨져나오는 것입니다. 그래서 MySQL 콘솔에 set names euckr;을 입력하고 다시 SELECT를 해보면 제대로 보이는 것을 확인할 수 있습니다.



[그림 10-9] 문자셋 문제를 해결한 결과

Section 06

방명록 글 목록

방명록에 새로운 글을 등록하였으니 등록된 글의 목록을 출력해 봅시다. 글 목록 기능에서는 페이징 기법이 가장 중요합니다. 글이 100개가 등록된 경우 페이징을 하지 않으면 한 페이지에 100개의 글이 모두 출력되어 스크롤의 압박을 받게 됩니다. 그래서 많은 글을 여러 페이지로 나누는 것을 페이징이라고 합니다.

데이터베이스에 저장된 글을 가져와야 하므로 글 저장하기에서와 마찬가지로 데이터베이스에 연결합니다.

```
$conn = mysql_connect("localhost", "사용자아이디", "패스워드");
mysql_select_db("데이터베이스이름", $conn);
mysql_query("set names euckr");
```

그리고 우선 한 페이지에 보이게 될 글의 수를 정의합니다. 일단 한 페이지에 다섯 개의 글이 출력되도록 합니다.

```
$pagesize = 5;
```

다음으로 데이터베이스에서 글을 가져오는 쿼리를 작성합니다.

```
SELECT * FROM guestbook ORDER BY id DESC
```

guestbook 테이블에서 글을 가져오되 id 값을 기준으로 내림차순 정렬을 합니다. 또한 모든 필드의 값을 가져옵니다. 작성된 쿼리를 변수에 저장하고 쿼리를 전송합니다.

```
$query = "SELECT * FROM guestbook ORDER BY id DESC";
$result = mysql_query($query, $conn);
```

위의 쿼리의 결과가 몇 개나 있는지 확인하기 위해서 mysql_affected_rows() 함수를 사용할 수 있습니다. 이 함수를 이용하여 총 몇 개의 튜플이 검색되었는지를 확인하여 페이징할 때 도움을 주도록 합니다.

```
$total = mysql_affected_rows();
```

그런데 위의 쿼리는 guestbook 테이블에 저장된 모든 글을 가져옵니다. 그래서 페이징을 위해서 이중 $pagesize만큼만 글을 가져오는 방법을 고안해야 합니다.

mysql_data_seek() 함수는 mysql_query() 함수를 통해서 얻어진 튜플 중에서 포인터를 이동하여 해당 튜플을 선택할 수 있는 기능을 하는 함수입니다. 이 함수를 이용하면 전체의 글 중에서 몇 번째 글을 선택할 수 있으므로 페이징을 구현할 수 있습니다.

```
mysql_data_seek($result, 0);
```

위와 같이 사용하면 0번째 튜플 앞에 포인터가 설정되고, mysql_fetch_array() 함수를 이용해서 0번째 튜플을 가져옵니다. 따라서 for 문을 이용하여 현재 페이지에 출력될 글의 번호를 찾아서 출력하면 글 목록을 구현할 수 있습니다.

```
for($i=$_GET[no] ; $i < $_GET[no]+$pagesize ; $i++) {
  if ($i >= $total) break;
  mysql_data_seek($result,$i);
  $row = mysql_fetch_array($result);
  //목록을 위한 HTML 출력 부분을 삽입한다.
}
```

여기서 $_GET[no]는 0, 5, 10과 같이 $pagesize에 따라 증가하는 값으로 해당 페이지에서 첫 번째 글을 구분 짓기 위한 값입니다. 만약 $no 값이 0이라면 전체 글 중에서 0번째 글부터 5개를 가져오며 $no 값이 10이라면 10번째 글부터 5개의 글을 가져옵니다. 그런데 이때 글이 총 11개가 있다면 10번째 글부터 5개를 가져오는데 4개의 글을 가져오지 못해서 에러가 출력됩니다. 이를 막기 위해서 총 검색된 글의 수와 현재 글이 몇 번째 글인지 비교하여 출력을 그만두도록 합니다.

```
<?
  for($i=$_GET[no] ; $i < $_GET[no]+$pagesize ; $i++) {
    if ($i < $total)
    {
      mysql_data_seek($result,$i);
      $row = mysql_fetch_array($result);
?>
<TABLE WIDTH=500 BORDER=1>
```

```
<TR>
  <TD>No. <?=$row[id]?></TD>
  <TD><?=$row[name]?></TD>
  <TD>(<?=$row[wdate]?>)</TD>
  <TD><a href="delete.php?id=<?=$row[id]?>">del</a></TD>
</TR>
<TR>
  <TD COLSPAN=4><?=$row[content]?></TD>
</TR>
</TABLE>
<?
    }
  }
?>
```

이제 마지막으로 페이지가 여러 개일 경우에 이전 페이지와 다음 페이지로 이동할 수 있는 링크를 만듭니다.

```
<?
  $prev = $no - $pagesize ; // 이전 페이지는 시작 글에서 $pagesize를 뺀 값부터
  $next = $no + $pagesize ; // 다음 페이지는 시작 글에서 $pagesize를 더한 값부터

  if ($prev >= 0) {
     echo("<a href='$PHP_SELF?no=$prev'>이전</a> ");
  }

  if ($next < $total) {
     echo("<a href='$PHP_SELF?no=$next'>다음</a></center>");
  }
?>
```

이전 페이지는 현재 페이지의 시작 글에서 $pagesize를 뺀 값부터 시작합니다. 그런데 $no에서 $pagesize를 뺐을 때 음수가 나오면 이전 페이지가 존재하지 않음을 의미합니다. 따라서 $prev 값이 0 이상인 경우에만 이전 링크를 보여줍니다.

이와 유사하게 다음 페이지는 현재 페이지의 시작 글에서 $pagesize를 더한 값부터 시작합니다. 그런데 더한 $next 값이 총 글의 수보다 크게 될 수 없으므로 $next 값이 $total보다 작을 때에만 다음 링크를 보여줍니다.

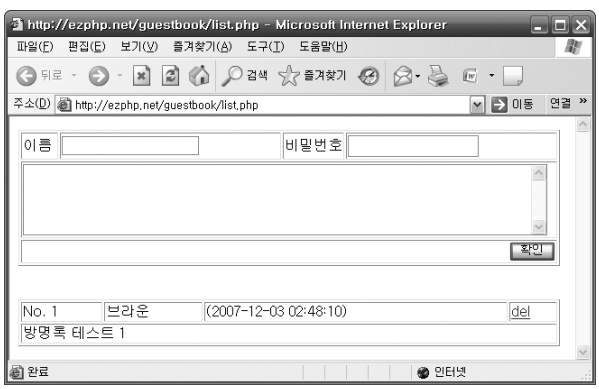
글 쓰기와 글 목록 기능을 하나로 합치면 다음과 같습니다.

[예제 10-3] list.php – 글 목록 및 글 쓰기

```
<?
  $conn = mysql_connect("localhost", "사용자아이디", "패스워드");
  mysql_select_db("데이터베이스이름", $conn);
  mysql_query("set names euckr");

  $query = "SELECT * FROM guestbook ORDER BY id DESC";
  $result = mysql_query($query, $conn);
  $total = mysql_affected_rows();

  $pagesize=5;
?>

<FORM ACTION="insert.php" METHOD="POST">
<TABLE BORDER=1 WIDTH=600>
  <TR>
    <TD>이름</TD><TD><INPUT TYPE="TEXT" NAME="name"></TD>
    <TD>비밀번호</TD><TD><INPUT TYPE="PASSWORD" NAME="pass"></TD>
  </TR>
  <TR>
    <TD COLSPAN=4><TEXTAREA NAME="content" COLS=80 ROWS=5></TEXTAREA></TD>
  </TR>
  <TR>
    <TD COLSPAN=4 align=right><INPUT TYPE="SUBMIT" VALUE=" 확인 "></TD>
  </TR>
</TABLE>
</FORM>
<BR>
<?
  for($i=$_GET[no] ; $i < $_GET[no]+$pagesize ; $i++) {

    if ($i < $total)
    {
      mysql_data_seek($result,$i);
      $row = mysql_fetch_array($result);
?>
<TABLE WIDTH=600 BORDER=1>
  <TR>
    <TD>No. <?=$row[id]?></TD>
    <TD><?=$row[name]?></TD>
    <TD>(<?=$row[wdate]?>)</TD>
    <TD><a href="delete.php?id=<?=$row[id]?>">del</a></TD>
  </TR>
  <TR>
    <TD COLSPAN=4><?=$row[content]?></TD>
  </TR>
```

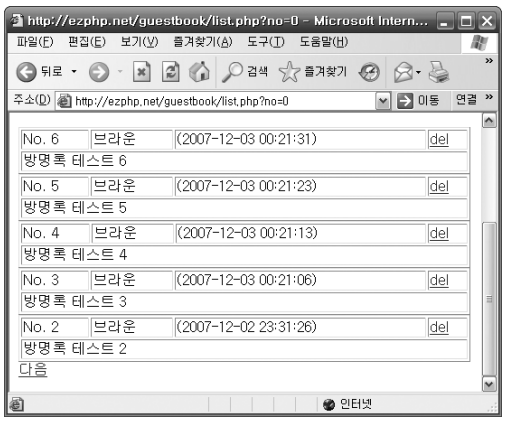
```
46 </TABLE>
47 <?
48      } //end if
49   } //end for
50
51   $prev = $no - $pagesize ; // 이전 페이지는 시작 글에서 $scale을 뺀 값부터
52   $next = $no + $pagesize ; // 다음 페이지는 시작 글에서 $scale을 더한 값부터
53
54   if ($prev >= 0) {
55     echo("<a href='$PHP_SELF?no=$prev'>이전</a> ");
56   }
57   if ($next < $total) {
58     echo("<a href='$PHP_SELF?no=$next'>다음</a></center>");
59   }
60 ?>
```

작성된 list.php 파일을 통해서 좀 전에 등록한 글이 목록에 잘 표시되는지 확인해 봅시다.



[그림 10-10] 등록된 글이 나타난 목록

글을 좀 더 등록하여 이전/다음 링크가 제대로 출력되는지도 확인해 봅시다.



[그림 10-11] 페이징된 결과

총 글을 6개 등록하였더니 첫 번째 페이지에 5개의 글이 출력되고 나머지 하나의 글은 다음 페이지에 있기 때문에 다음 링크가 출력된 것을 확인할 수 있습니다. 만약 다음 링크를 클릭한다면 No. 1 글이 출력되고 이전 링크가 출력된 페이지를 볼 수 있습니다.

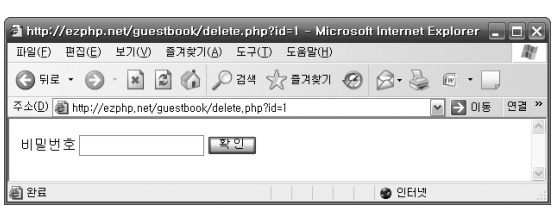
Section 07 방명록 글 삭제

앞서 글 목록에서 글 삭제를 위한 del 링크를 설정해 두었습니다.

```
<a href="delete.php?id=<?=$row[id]?>">del</a>
```

이 링크를 잘 보면 id 값을 delete.php 파일로 전송하고 있음을 알 수 있습니다. 이는 지워야 할 글을 정확히 명시해주기 위해서입니다. 만약 id 값을 전달해주지 않는다면 delete.php 페이지에서는 어떤 글을 지워야 하는지 알 수 없을 것입니다.

글을 삭제하기 이전에 글의 삭제 권한이 있는지 확인하기 위해서 비밀번호를 확인해야 합니다.



[그림 10-12] 패스워드 확인 폼

이렇게 입력된 비밀번호를 다시 검사하여 해당 글의 비밀번호와 같은지 확인을 해야 합니다. 이처럼 비밀번호를 입력받고 또한 비밀번호가 올바른지 검증한 후 데이터베이스에서 해당 글을 삭제하는 기능을 구현하고자 일반적으로 두 개의 파일을 만들지만, 두 파일 모두 너무 짧고 간단하므로 mode 값에 따라서 비밀번호 입력을 할 수도 있고 비밀번호 검증 후 글을 삭제할 수도 있게 만들어 보겠습니다.

```
<?

  if ($mode!="delete")
  {
?>
    <HTML>
    <FORM METHOD="POST"
          ACTION="<?=$PHP_SELF?>?id=<?=$_ GET[id]?>&mode=delete">

    <TABLE>
    <TR>
      <TD>비밀번호</TD>
      <TD><INPUT TYPE="PASSWORD" NAME="pass"></TD>
      <TD><INPUT TYPE="SUBMIT" VALUE=" 확 인 "></TD>
    </TR>
    </TABLE>
<?
  exit;
  }
?>
```

mode 값이 delete가 아니면 비밀번호를 입력하는 폼을 출력하게 합니다. 폼의 action 정보를 잘 보면 자기 자신으로 데이터를 전송하는 것을 확인할 수 있습니다.

```
ACTION="<?=$PHP_SELF?>?id=<?=$_GET[id]?>&mode=delete"
```

만약 비밀번호를 입력하고 확인 버튼을 클릭하면 mode 값으로 delete 값이 전송되므로 데이터베이스에서 글을 삭제하는 코드를 실행할 수 있게 합니다.

데이터베이스에서 글을 삭제하려면 일단 데이터베이스에 연결합니다.

```
$conn = mysql_connect("localhost", "사용자아이디", "패스워드");
mysql_select_db("데이터베이스이름", $conn);
mysql_query("set names euckr");
```

그리고 GET으로 전달된 $id 값을 통해 해당 글의 비밀번호를 가져옵니다.

```
$query = "SELECT pass FROM guestbook WHERE id='$_GET[id]'";
$result = mysql_query($query, $conn);
$row = mysql_fetch_array($result);
```

가져온 비밀번호는 입력한 값과 비교하여 올바른 경우에 데이터베이스에서 삭제합니다.

```
if ($row[pass] == $_POST[pass])
{
  $query = "DELETE FROM guestbook WHERE id='$id'";
  $result = mysql_query($query, $conn);
}
```

그리고 다시 글 목록 페이지로 이동합니다.

```
<script> alert("글이 삭제되었습니다."); location.href="list.php"; </script>
```

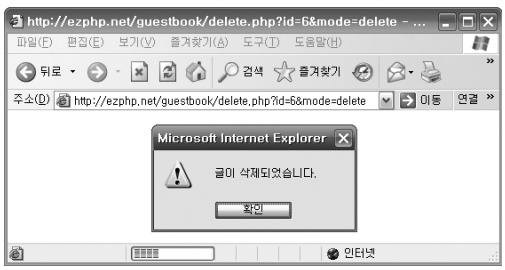
정리된 소스는 다음과 같습니다.

[예제 10-4] delete.php – 글 삭제하기

```
<?
  if ($mode!="delete")
  {
?>
<HTML>
<FORM METHOD="POST"
ACTION="<?=$PHP_SELF?>?id=<?=$_GET[id]?>&mode=delete">
<TABLE>
  <TR>
    <TD>비밀번호</TD>
    <TD><INPUT TYPE="PASSWORD" NAME="pass"></TD>
    <TD><INPUT TYPE="SUBMIT" VALUE=" 확 인 "></TD>
  </TR>
</TABLE>
<?
    exit;
  } //end if

  $conn = mysql_connect("localhost", "사용자아이디", "비밀번호");
  @mysql_select_db("데이터베이스이름", $conn);
  @mysql_query("set names euckr");

  $query = "SELECT pass FROM guestbook WHERE id='$_GET[id]'";
  $result = mysql_query($query, $conn);
  $row = mysql_fetch_array($result);

  if ($row[pass] == $_POST[pass])
  {
    $query = "DELETE FROM guestbook WHERE id='$id'";
    $result = mysql_query($query, $conn);
  }
```

```
32 ?>
33 <script> alert("글이 삭제되었습니다."); location.href="list.php"; </script>
```

완성된 소스를 통해서 글을 삭제해보면 다음과 같습니다.



[그림 10-13] 글이 삭제된 경우

Section 08

이 방명록의 문제점

이 방명록은 99년 PHPSCHOOL.COM에서 사용하던 방명록을 본떠서 만든 것입니다. 이 당시에는 PHP 프로그래밍이 지금과 같이 널리 퍼져 있을 때가 아니었고 PHP 버전도 PHP 3일 때였습니다. 이때는 PHP나 MySQL이나 모두 초기 버전을 갓 넘기고 안정화 버전으로 변모하던 시기였기 때문에 기능적으로나 프로그래머의 경험으로나 모두 부족한 점이 많았습니다.

이 방명록은 사실 많은 문제점을 가지고 있는데 보안 이슈는 모두 무시하고 보았을 때 가장 큰 문제는 글 목록에서 전체의 글을 가져오는 쿼리문입니다.

```
SELECT * FROM guestbook ORDER BY id DESC
```

위의 쿼리를 사용하여 글이 1000개이건 10000개이건 상관없이 항상 모든 글을 검색합니다. 실제로 한 페이지에 보여주어야 할 글은 5개 남짓인데 5개의 글만 검색하면 될 것을 1000개, 10000개 글을 검색하는 것입니다. 실제로 이 당시에 많은 프로그램이 위와 같이 전체 글을 가져와서 필요한 부분만 출력하는 코딩을 많이 했습니다. 그래서 이런 프로그램들은 하나같이 글의 수가 늘어가면 늘어갈수록 속도가 현저하게 느려지곤 하였습니다.

이러한 문제를 해결하려면 MySQL의 LIMIT 키워드를 사용하면 됩니다. LIMIT 키워드는 전체 레

코드 중에서 몇 번째 레코드부터 몇 개의 글을 가져올지를 지정할 수 있습니다. 따라서 위의 방법보다는 훨씬 효율적으로 목록을 구성하게 됩니다. 자세한 방법은 "11장. 게시판 만들기"에서 배울 수 있습니다.

Chapter

11

게시판 만들기

이 장에서는 게시판을 만들어 보겠습니다. 게시판은 웹 프로그래머에게는 정말 특별한 존재라고 생각되는데요, 웹 프로그램을 개발하면서 가장 많이 사용하는 소스가 바로 게시판 소스입니다. 실제 게시판으로서의 기능뿐만이 아니라 현존하는 대부분의 웹 프로그램은 게시판을 기반으로 한다고 할 수 있습니다. 어떤 정보를 목록이나 특정 형태의 폼으로 출력하고 해당 정보를 수정하고 삭제하는 것은 모두 게시판의 기능을 응용해서 만드는 것입니다.

필자도 그러했듯이 웹 프로그래머가 되면 누구나 자신만의 게시판을 만들고 싶어 합니다. 실제로 괜찮은 공개 게시판이 많이 존재하지만 자신이 개발한 그리고 자신이 이름 붙인 좋은 게시판을 만드는 것은 웹 프로그래머의 로망이라고 할만 합니다. 내가 만든 게시판을 많은 사람이 사용한다면 뿌듯하지 않을까요?

하지만 지금부터 만들고자 하는 게시판은 제로보드 같은 거창한 게시판이 아닙니다. 그야말로 게시판 계의 햇병아리 수준의 게시판입니다. 앞서 말씀드렸듯이 대부분의 응용 프로그램들은 제로보드와 같은 거대한 게시판이 아니라 이와 같은 간단한 게시판을 기반으로 합니다.

Section 01

게시판 둘러보기

우선 게시판이 어떤 식으로 구성되어 있는지 알아보도록 합시다. 사실 PHP를 공부하는 분 중에 게시판이 어떤 것인지 모르는 분은 없겠지만 우리가 만들어 볼 게시판의 모양이 어떤 것인지 어떠한 기능들이 있는 것인지를 먼저 알아보는 취지입니다. 여기서 우리가 만들어 보고자 하는 게시판은 가장 기본적인 기능만을 다루고 있으며 다음과 같이 구성되어 있습니다.

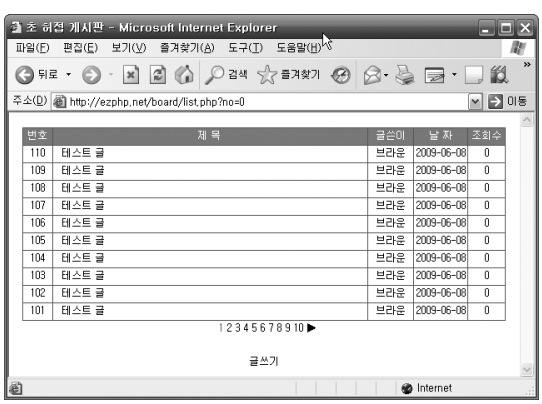
구성	설명
글 목록 보기	글을 10개 혹은 20개 단위로 목록을 보여주는 페이지
글 내용 보기	글의 자세한 내용을 보는 페이지
글 입력	글을 새로 입력하는 페이지
글 수정	등록된 글을 수정하는 페이지
글 삭제	등록된 글을 삭제하는 페이지

[표 11-1] 게시판의 구성

크게 위와 같이 5가지로 구분할 수 있으며 글 입력과 수정은 글을 입력/수정하는 폼으로 구성된 페이지와 입력/수정된 값을 데이터베이스에 반영하는 페이지로 구성되며, 글 삭제는 글쓴이를 검증하기 위한 비밀번호를 묻는 페이지와 비밀번호를 검사하여 올바른 경우 데이터베이스에서 삭제하는 페이지로 나누어 집니다.

그렇다면 실제 우리가 만들어볼 게시판의 총체적인 모습을 살펴보겠습니다.

글 목록 보기

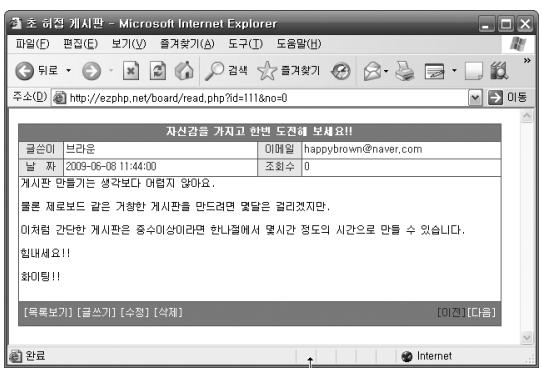


[그림 11-1] 글 목록

게시판의 기본 페이지라고 할 수 있는 글 목록 보기 페이지입니다. 방명록은 바로 방명록의 내용을 보여주는 것과 달리 게시판은 글의 제목 등을 간단하게 보여주는 글 목록 보기 기능을 가지고 있습니다. 대체로 위의 그림과 같이 글의 번호, 제목, 글쓴이, 날짜, 조회 수와 같은 값을 기본적으로 보여주어 글의 내용을 선택해서 볼 수 있게끔 하는 기능을 합니다.

글 목록 보기에서는 페이지를 구분하는 것이 핵심입니다. 많은 글을 10개 혹은 20개 등의 단위로 하나의 페이지에 목록화해서 보여주는 것으로 일반적으로 페이징 기법이라고 합니다. 게시판의 정복은 이 페이징 기법에 있다고 해도 과언이 아닙니다.

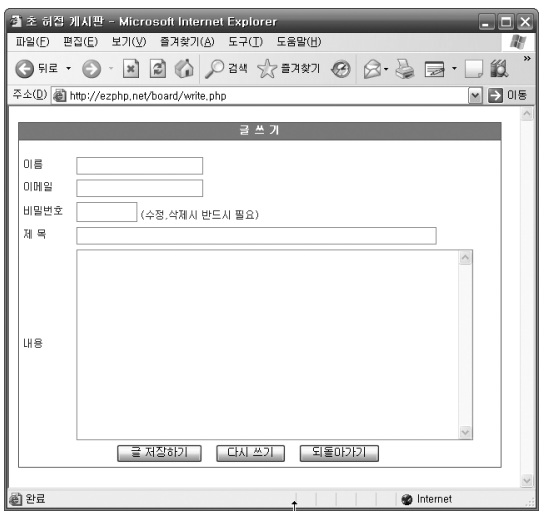
글 내용 보기



[그림 11-2] 글 내용 보기

글의 모든 정보를 보여주는 페이지로 제목과 글쓴이, 글의 내용을 비롯하여 해당 글에 대한 수정과 삭제를 할 수 있는 링크를 제공합니다. 또한 이전/다음 링크를 이용하여 글 목록 페이지로 가지 않고 바로 이전 글이나 다음 글로 바로 갈 수 있게끔 하는 기능을 제공합니다.

글 쓰기

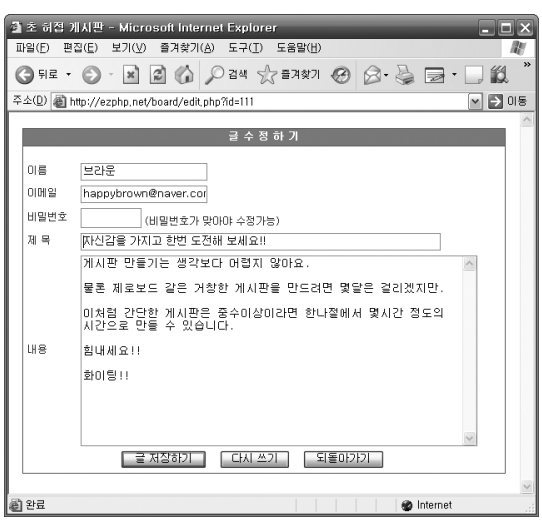


[그림 11-3] 글 쓰기

글의 내용을 입력하는 페이지로 필수적으로 이름, 비밀번호, 제목, 내용을 입력합니다. 이외에 일반적으로 이메일 주소나 홈페이지 등을 입력하기도 합니다. 또한 그림이나 문서 파일을 업로드하거나 최근에는 게시물의 키워드(태그)를 입력하여 글을 키워드로 분류하거나 하는 등에 사용하기도 합니다.

이 페이지는 단순히 HTML로 구성된 페이지로 글 저장하기 버튼을 누른다고 하여 모든 작업이 끝나는 것이 아닙니다. 이 페이지는 단지 HTML의 폼 페이지이며 입력된 값을 데이터베이스에 저장하는 페이지가 별도로 존재해야 합니다.

글 수정하기



[그림 11-4] 글 수정하기

글 수정하기는 글 쓰기와 유사한 모습을 하고 있습니다. 실제로 글 쓰기 페이지를 그대로 가져다가 약간 수정해서 사용할 수 있으며 글 쓰기 페이지와 다른 점은 이미 등록된 글을 수정하는 기능이기 때문에 기존의 입력된 글을 다시 불러와서 보여주어야 한다는 점입니다. 따라서 이 페이지는 단순 HTML 폼 페이지가 아니라 PHP를 이용하여 데이터베이스에서 해당 글의 내용을 가져오는 기능이 포함되어 있습니다.

글을 수정하려면 반드시 본인 인증을 해야 합니다. 글쓴이가 아닌데 글을 수정할 수 있다면 큰 문제가 생길 수 있기 때문입니다. 명의를 도용한다거나 남을 비방하는데 악용할 가능성이 있겠죠. 그렇기 때문에 반드시 비밀번호를 입력받아 비밀번호가 맞는 경우 글을 수정하게끔 해야 합니다.

이 페이지 또한 글 쓰기 페이지와 마찬가지로 글 저장하기 버튼을 누르는 것만으로 모든 작업이 끝나는 것이 아니라 데이터베이스에 수정된 내용을 반영하는 페이지가 별도로 필요합니다.

글 삭제하기

[그림 11-5] 글 삭제를 위한 비밀번호 입력

글 삭제는 글 수정과 마찬가지로 반드시 본인 인증이 필요합니다. 따라서 글을 쓸 때 입력한 비밀번호를 확인하는 작업을 합니다.

이 페이지는 단순하게 비밀번호를 입력받는 HTML 폼 페이지로서 비밀번호를 검증하여 올바른 경우 글을 데이터베이스에서 삭제하는 페이지가 별도로 요구됩니다.

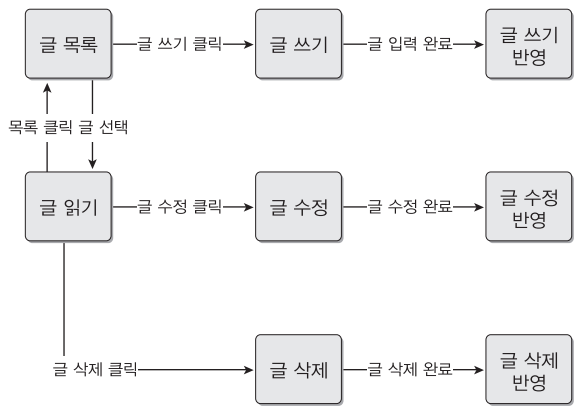
기타 구성

실제로 사용되는 게시판에는 게시물 검색과 답글 기능 그리고 짧은 덧글(코멘트) 기능 등 이 게시판과 다른 여러 가지 기능을 더 포함하고 있습니다. 이러한 진정한 게시판으로의 모양새는 점차 다듬어 나가는 방식으로 게시판을 완성해 보겠습니다.

Section 02

게시판 설계

게시판을 만들기에 앞서 설명한 구성요소들의 흐름을 살펴보도록 합시다. 모든 경우를 설명한 것은 아니지만 게시판의 구동 방식에 대한 개략적인 개념을 이해할 수 있을 것입니다.



[그림 11-6] 게시판의 구조

게시판은 위와 같은 개략적인 흐름을 갖습니다. 앞서도 살짝 언급한 내용이니 그다지 이해하기 힘들지는 않을 것이라 생각합니다. 모든 흐름에 대해서 자세히 표현하지 않은 것은 여러분이 어지러이 연결된 선들을 보고 어렵게 생각할 것 같아 간략히 표현했습니다.

흐름도에 표현되지 않은 정책은 다음과 같은 것들이 있습니다.

① 글 쓰기 도중 취소 시 이전 페이지로 돌아감
② 글 쓰기 반영 후 글 목록으로 돌아감
③ 글 수정 도중 취소 시 이전 페이지(글 읽기)로 돌아감
④ 글 수정 반영 후 글 읽기로 돌아감
⑤ 글 삭제 도중 취소 시 이전 페이지(글 읽기)로 돌아감
⑥ 글 삭제 반영 후 글 목록으로 돌아감

이와 같은 흐름을 정확하게 프로그램에 적용한다면 빈틈없는 게시판을 만들 수 있을 것입니다. 만약 위와 같이 흐름을 정확하게 알아두지 않고 작업을 한다면 글 삭제 완료 후 사라진 글을 읽게 하려고 시도한다거나 글 쓰기 반영 후 안드로메다로 가버리는 경우가 생길지도 모릅니다.

위와 같은 흐름도를 바탕으로 각 기능을 하는 파일을 나눠 보도록 하겠습니다. 위 흐름도에서 사각형으로 표시된 박스는 파일 하나로 표현됩니다. 그렇다면 총 몇 개의 파일로 게시판을 만들 수 있을까요? 물어보나 마나 파일 8개로 게시판을 만들어 볼 것입니다.

파일 이름	파일의 역할
db_info.php	데이터베이스 접속 정보
list.php	글 목록
read.php	글 읽기
write.php	글 쓰기
insert.php	글 쓰기 반영
edit.php	글 수정
update.php	글 수정 반영
predel.php	글 삭제를 위한 비밀번호 입력
del.php	글 삭제 반영

[표 11-2] 게시판 파일의 구성

이미 독자 분 중 많은 분이 눈치채셨겠지만 앞서 말한 것과 다르게 게시판이 총 9개의 파일로 구성되어 있습니다. db_info.php 파일은 원래 굳이 파일로 만들지 않아도 되지만 편의를 위해 따로 떼어내어 만든 파일입니다. 그 이유는 나중에 자세히 설명합니다.

위 표를 보면 대충 어떤 파일이 무슨 역할을 하는지 알 수 있을 겁니다. 파일 이름은 필자가 임의로 정한 것으로 필자의 어줍짢은 영어 실력에 딴죽을 걸지는 말기 바랍니다.

테이블 설계

글을 저장할 테이블을 설계해 보겠습니다. 테이블을 설계할 때 가장 먼저 해야 할 것이 무엇일까요? 바로 이름이죠. 우선 테이블의 이름을 정해보도록 합시다. 우리가 보통 이름을 지을 때 남자는 '철수' 여자는 '영희'라고 짓듯이 가장 일반적인 이름인 'board'로 지어 봅시다. (물론 이 작명법은 꽤 오래 전 것이고 최근 뉴스 기사를 보면 남자는 '민준' 여자는 '서연'을 가장 많이 쓴다고 합니다.)

테이블 이름은 지었고 이제 테이블에 무엇을 저장할 것인지를 생각해 봅시다. 만약 주소록을 만든다고 한다면 이름, 나이, 전화번호, 주소 등의 정보를 저장하면 될 것 같은데 게시판에는 어떤 정보들이 저장되어야 할까요?

하나하나씩 게시판에 필요한 것들을 나열해 봅시다.

① 글쓴이
② 이메일 주소
③ 글의 비밀번호 (수정과 삭제를 위해서)
④ 글의 제목
⑤ 글의 내용
⑥ 글쓴 날짜

정도를 떠올릴 수 있겠네요. 뭐 당연히 더 있을 수 있습니다.

글을 사람들이 몇 번 봤는가? 하는 조회 수도 있으면 좋겠고 혹시 모르니깐 글 쓴 사람의 IP 주소를 남겨 두는 것도 좋습니다. 뭐 글쓴이의 홈페이지 주소도 저장할 수 있음 좋겠네요. 방금 나열한 이런 것들이 바로 테이블의 저장 항목이 됩니다.

이러한 항목들을 영어로 바꾸어 봅시다. 테이블에 사용되는 항목은 한글로 사용할 수 없고 사용할 수 있다고 하더라도 표준 언어인 영어를 사용하는 것이 바람직합니다. 웹 프로그램은 개발한 시스템에서 다른 시스템으로 이전하는 경우가 많이 있습니다. 이런 경우 다른 시스템이 이전 개발 환경과 같은 환경을 유지하고 있다고 장담할 수 없습니다. 그래서 항상 개발할 때는 여러 시스템에서 호환할 수 있게 작업하는 것이 추후 많은 시간을 줄이게 합니다.

항 목	간단하게 영어로 변환한 항목
글쓴이	name
이메일 주소	email
글의 비밀번호	pass
글의 제목	title
글의 내용	content
글쓴 날짜	wdate
IP 주소	ip
조회 수	view

[표 11-3] 영어로 표현한 게시판 테이블 저장 목록

대부분 한글에 해당하는 영어 단어를 사용했습니다. 비밀번호는 편의를 위해 긴 password 대신에 pass를 사용했고 조회 수는 view_count 정도로 이름을 지을 수 있겠으나 이것 또한 편의상 view라고 짧게 사용했습니다. 그런데 여기서 글쓴 날짜를 date라고 해도 될 텐데 wdate라고 이름을 지은 것은 date가 MySQL에서 키워드로 사용되고 있으므로 이를 피하고자 write_date를 줄여서 wdate라고 명명한 것입니다.

그런데 여기서 매우 중요한 하나가 빠졌습니다. 바로 많은 글 중에 해당 글을 선택할 수 있게 글을 구분 지어주는 구분자입니다. 글을 읽거나 지우거나 수정하고자 할 때 어떤 글을 가르키는지 알아야만 다른 글을 잘못 지우거나 하는 실수를 하지 않을 테니까요. 그래서 글의 번호가 있어야 합니다. 그리고 그 글 번호는 반드시 유일해야 합니다. 너무나 당연하지만 글의 번호가 중복된다면 중복된 글을 어떻게 구분 지을 수 있을까요? 제목으로 판단하면 된다고요? 제목마저 같다면, 글쓴이마저 같다면 글을 쓴 시간으로 판단하면 된다고요? 이처럼 글을 구분 짓다가는 느려터져서 속이 터질 겁니다.

이런 이유로 위의 테이블에 id라는 글 번호를 하나 더 추가합니다.

항 목	간단하게 영어로 변환한 항목
글의 번호	id
글쓴이	name
이메일 주소	email
글의 비밀번호	pass
글의 제목	title
글의 내용	content
글쓴 날짜	wdate
IP 주소	ip
조회 수	view

[표 11-4] 완성된 테이블 저장 정보

테이블에 저장할 정보들과 이름이 정해졌으니 저장 항목의 형식과 크기를 결정해 봅시다.

항 목	영어 항목	변수형	크기	비 고
글의 번호	id	int	11	글의 번호는 숫자
글쓴이	name	varchar	20	255자 이하의 길지 않은 문자열
이메일 주소	email	varchar	30	
글의 비밀번호	pass	varchar	12	
글의 제목	title	varchar	70	
글의 내용	content	text		충분히 긴 문자열
글쓴 날짜	wdate	datetime		날짜 및 시간
IP 주소	ip	varchar	15	
조회 수	view	int	11	

[표 11-5] 필드 변수형 정의

글의 번호

글의 번호는 숫자이며 11자리 정도의 수로 충분히 표현할 수 있습니다. 따라서 int형에 크기는 11로 설정합니다. 11자리 이상의 글의 번호가 생성되면 어떻게 하느냐고 걱정하시는 분들이 계실지 모르겠습니다. 이런 분들은 원래 세상 걱정이 정말 많으시거나 숫자의 자릿수에 대한 감각이 더디신 분이라고 생각됩니다. 11자리 숫자는 수백억까지 나타낼 수 있는 아주 아주 큰 숫자입니다. 아무리 인기가 좋다 하더라도 웬만한 사이트에서 수백억 개의 글이 올라올 수는 없을 겁니다. 가령 글이 올라온다고 하더라도 수백억 개의 글이 등록되면 데이터베이스가 감당하기도 힘들고 그 정도 글이 저장되려면 설계를 수정해야 합니다.

글쓴이

글쓴이는 사람 이름이니까 길어야 한글로 4자를 넘지 않는 것이 대부분이고 영어 이름이더라도 20자 정도면 충분히 표현되지 않을까 생각합니다. 또한 이름이 너무 길다고 하더라도 이름을 전부 보여줄 필요가 없습니다. 따라서 문자이니 varchar형, 크기는 20 정도로 잡습니다.

이메일은 매우 긴 경우가 있을 수 있으나 테이블을 설계할 때 모든 가능성을 대비하여 테이블을 작성하지 않아도 됩니다. 100자 길이의 이메일을 쓰는 사람은 10만 명 중에 1명 있을까 말까 합니다. 그런 사람을 위해 모든 테이블을 낭비하는 것은 좋은 방법이 아닙니다. 그래서 적당히 조절해야 합니다. 이메일은 30자 정도면 99% 이상 충분히 커버할 수 있을 것입니다. 따라서 문자형으로 30자를 설정합니다.

비밀번호

비밀번호는 길면 길수록 좋습니다. 그러나 이메일과 같은 이유로 무한정 길게 허용할 수는 없습니다. 보통 대략 4자에서 12자 사이가 대부분일 것으로 생각합니다. 따라서 문자형으로 12자로 설정합니다.

글의 제목

글의 제목은 충분히 길게 70자로 하고 글의 내용은 길이 제한이 없으므로 TEXT 타입으로 설정합니다.

글쓴 날짜

날짜와 시간을 표현하는 방법은 datetime과 방명록에서 사용한 timestamp 그리고 microtime()을 이용한 int형 표현이 있습니다. 편의에 따라 어떤 것을 사용해도 좋으며 여기서는 datetime형을 사용하겠습니다.

IP

IP는 IPv4에서 255.255.255.255와 같이 최대 15자리로 모두 표현이 가능하므로 문자열 15자로 설정합니다.

조회 수

조회 수는 일반적으로 수십에서 수천 정도면 충분하지만 통 크게 몇만 몇십만도 가능하도록 int형

으로 설정합니다.

각 항목별 타입과 크기를 정했으니 마지막으로 각 항목별 NULL 허용 여부와 키(Primary Key) 등을 설정해 보겠습니다.

항 목	영어 항목	키	필수 항목	기타
글의 번호	id	Primary Key	필수	1씩 증가하는 0 이상의 수
글쓴이	name		필수	
이메일 주소	email		옵션	
글의 비밀번호	pass		필수	
글의 제목	title		필수	
글의 내용	content		필수	
글쓴 날짜	wdate		필수	
IP 주소	ip		필수	
조회 수	view		필수	기본값 0

[표 11-6] NULL과 키 제약 등의 정보

글의 번호는 board 테이블을 구분 짓는 구분자로써 키(Primary Key)가 됩니다. 또한 앞서 언급하였듯이 1씩 증가하는 0 이상의 정수입니다. 글쓴이의 이름과 비밀번호 및 제목 그리고 내용은 반드시 필요한 항목이므로 사용자로부터 반드시 입력을 받아야 하고 글의 번호 글쓴 날짜 그리고 IP 주소와 조회 수는 프로그램에서 자동으로 입력해야 하는 항목입니다. 단, 이메일 주소는 반드시 필요한 항목이 아니므로 기입하지 않아도 되게 합니다.

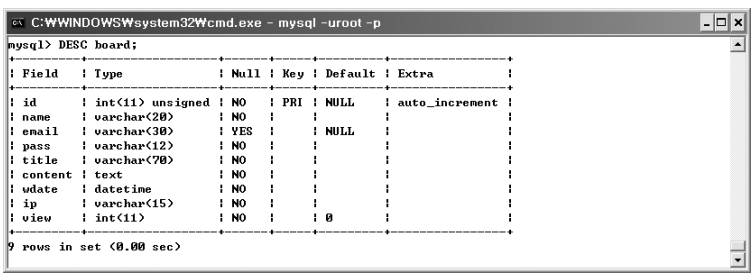
이를 바탕으로 테이블 생성문을 만들어 봅시다.

```
CREATE TABLE board (
id int(11) unsigned NOT NULL auto_increment,
name varchar(20) NOT NULL,
email varchar(30) NULL,
pass varchar(12) NOT NULL,
title varchar(70) NOT NULL,
content text NOT NULL,
wdate datetime NOT NULL,
ip varchar(15) NOT NULL,
view int(11) NOT NULL DEFAULT 0,
PRIMARY KEY (id)
);
```

위와 같이 테이블 생성문이 완성되었습니다. 테이블 생성에 사용된 쿼리가 이해가 되지 않는다면 데이터베이스 장을 참조하기 바랍니다.

테이블 생성

앞서 테이블을 설계했습니다. 테이블을 순서대로 차근차근 설계해 가면서 테이블을 어떻게 설계하는지에 대한 감각을 조금 체득했을 것으로 생각합니다. 이제 앞서 작성한 테이블 생성문을 이용하여 데이터베이스에 테이블을 생성해 봅시다. 테이블의 생성을 성공적으로 완료했으면 다음과 같은 방법으로 테이블이 제대로 생성되었는지 구조를 확인해 봅니다.



[그림 11-7] 생성된 게시판 테이블의 정보

드디어 테이블이 만들어졌습니다. 테이블 설계부터 생성까지 짧고도 생각보다 긴 여정이었던 것 같습니다. 게시판을 만들기 위한 기반 환경이 갖춰졌으니 이제 본격적으로 게시판 프로그램을 작성해 보겠습니다.

Section 03

데이터베이스 연결 파일 – db_info.php

데이터베이스에서 정보를 검색하거나 값을 입력하려면 우선 데이터베이스에 연결하는 코드를 작성해야 합니다. 데이터베이스에 연결하는 코드는 다음과 같이 매우 간단합니다.

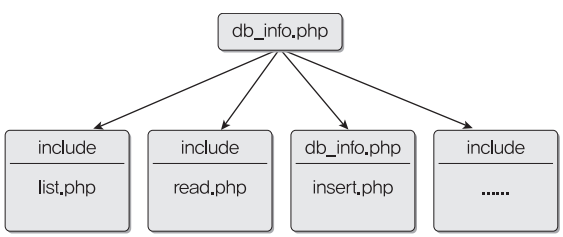
```
<?
  $conn = mysql_connect("localhost", "아이디", "패스워드");
  mysql_select_db("데이터베이스명",$conn);
?>
```

작성한 간단한 코드는 데이터베이스를 사용하는 모든 페이지에 추가되어야 합니다. 하지만 매 페이지마다 위와 같은 코드를 넣는 것은 매우 귀찮은 일이 아닐 수 없습니다. 지금 만들어 보고자 하는 게시판은 파일이 몇 개 되지 않기 때문에 조금의 귀찮음을 감수하면 되지만 예를 들어 파일이 100개 이상이라면 어떨까요? 비밀번호가 너무 간단하여 안전하지 않으니 비밀번호를 변경하라는 관리

자의 요청이 있었다고 생각해 봅시다. 데이터베이스 관리자를 이용하여 간단히 비밀번호를 변경하긴 했는데 100개 이상이 되는 페이지를 모두 수정해야 한다니 난감하기만 합니다. 물론 펄 스크립트나 괜찮은 에디터 등을 이용하여 문자열을 치환하는 방법으로 해결할 수도 있겠지만 여러 파일에 대한 동시 작업은 언제나 올바르게 수정되었는지 혹시나 수정하면 안 되는 부분을 수정하지는 않았는지 확인을 거쳐야 하기 때문에 역시 번거롭게 됩니다.

이럴 때 어떻게 하는 것이 최선의 방법일까요? 정답은 인클루드를 이용하는 것입니다. 앞서 인클루드를 배우면서 위와 같은 유사한 예를 들어서 설명했던 걸 기억할지 모르겠습니다. 필자가 생각하기에 이런 경우가 가장 바람직하면서 모범적인 인클루드의 실제 사용 예가 아닐까 생각합니다.

데이터베이스에 연결하고 데이터베이스를 선택하는 단 두 줄의 코드를 db_info.php 파일에 따로 저장합니다. 저장된 db_info.php 파일은 아래의 그림에서 보이듯이 데이터베이스를 필요로 하는 모든 페이지에서 파일 상단에 인클루드하기만 하면 자유롭게 데이터베이스에 접근할 수 있게 됩니다.

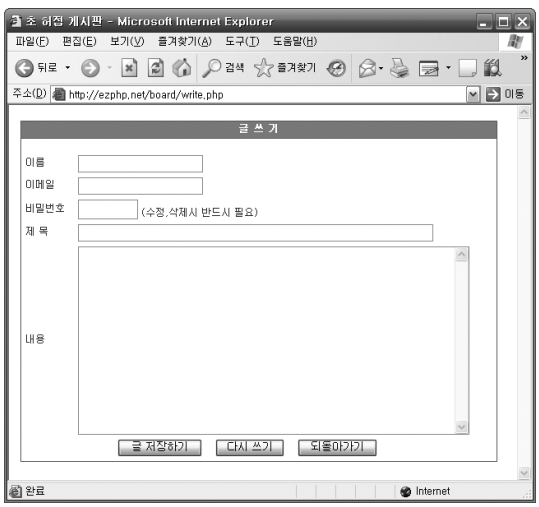


[그림 11-8] db_info.php의 연결 구조

db_info.php 파일은 데이터베이스 접속 정보를 가지고 있는 파일입니다. 이 파일을 데이터베이스를 사용하는 페이지 상단에 인클루드시키면 됩니다. 아이디와 패스워드 및 데이터베이스 이름은 자신의 설정에 맞게 수정하기 바랍니다.

Section 04

글 쓰기 – write.php



[그림 11-9] 글 쓰기 폼

글 쓰기 폼을 작성해 보도록 합시다. 앞서 테이블을 설계하면서 입력 값으로 이름, 비밀번호, 이메일, 제목, 내용을 입력하게끔 했습니다. 그 외의 값은 자동으로 입력되는 값이니 신경 쓸 필요가 없습니다. 대표적으로 글 번호를 글 쓰는 사람이 입력해야 한다고 합시다. 가장 마지막 글이 몇 번인지 확인하고 그것을 잊지 않고 기억해두었다가 글을 입력할 때 번호를 기입한다면 글 쓰기도 전에 다들 지쳐서 글 쓰기를 포기해 버릴지도 모릅니다.

따라서 자동으로 입력해야 하는 글의 번호와 글쓴 날짜, 글쓴이의 IP 마지막으로 해당 글의 조회 수는 따로 입력 폼을 작성할 필요가 없습니다. 또한 여기서 글쓴이의 IP는 나중에 광고나 악성 글로 도배를 하는 사용자를 차단하기 위한 목적이므로 사용자가 입력하게 해서는 절대 안 됩니다. 그러면 의미가 없어져 버리니까요.

write.php

```
1  <html>
2  <head>
3  <title>초 허접 게시판</title>
4  <style>
5  <!--
6  td { font-size : 9pt; }
7  A:link { font : 9pt; color : black; text-decoration : none;
8  font-family  : 굴림; font-size : 9pt; }
```

```
9  A:visited { text-decoration : none; color : black; font-size : 9pt; }
10 A:hover { text-decoration : underline; color : black;
11 font-size : 9pt; }
12 -->
13 </style>
14 </head>
15
16 <body topmargin=0 leftmargin=0 text=#464646>
17 <center>
18 <BR>
19 <!-- 입력된 값을 다음 페이지로 넘기기 위해 FORM을 만든다. -->
20 <form action=insert.php method=post>
21 <table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
22   <tr>
23     <td height=20 align=center bgcolor=#999999>
24     <font color=white><B>글 쓰 기</B></font>
25     </td>
26   </tr>
27   <!-- 입력 부분 -->
28   <tr>
29     <td bgcolor=white> 
30     <table>
31       <tr>
32         <td width=60 align=left >이름</td>
33         <td align=left >
34           <INPUT type=text name=name size=20 maxlength=10>
35         </td>
36       </tr>
37       <tr>
38         <td width=60 align=left >이메일</td>
39         <td align=left >
40           <INPUT type=text name=email size=20 maxlength=25>
41         </td>
42       </tr>
43       <tr>
44         <td width=60 align=left >비밀번호</td>
45         <td align=left >
46           <INPUT type=password name=pass size=8 maxlength=8>
47           (수정,삭제시 반드시 필요)
48         </td>
49       </tr>
50       <tr>
51         <td width=60 align=left >제 목</td>
52         <td align=left >
53           <INPUT type=text name=title size=60 maxlength=35>
54         </td>
55       </tr>
```

```
56        <tr>
57          <td width=60 align=left >내용</td>
58          <td align=left >
59            <TEXTAREA name=comment cols=65 rows=15></TEXTAREA>
60          </td>
61        </tr>
62        <tr>
63          <td colspan=10 align=center>
64            <INPUT type=submit value="글 저장하기">
65              
66            <INPUT type=reset value="다시 쓰기">
67              
68            <INPUT type=button value="되돌아가기"
69            onclick="history.back(-1)">
70          </td>
71        </tr>
72      </TABLE>
73 </td>
74 </tr>
75 <!-- 입력 부분 끝 -->
76 </table>
77 </form>
78 </center>
79 </body>
80 </html>
```

이 페이지는 단순히 HTML로 구성되어 있습니다. PHP 코드가 한 부분도 없는 거 보이시죠? 이 페이지는 그저 글의 내용을 입력받는 것을 목적으로 하기 때문입니다. 앞서 말씀드렸듯이 실제 데이터베이스에 저장하는 페이지가 따로 존재하기 때문에 이 페이지는 단순히 HTML로 구현되어 있습니다.

소스 코드를 살펴보면 폼 정보를 넘기려고 다음과 같은 폼 태그를 사용한 것을 알 수 있습니다.

```
<form action=insert.php method=post>
```

흐름도에서 이미 언급했듯이 입력된 폼 정보를 insert.php라는 파일로 넘겨주어야 합니다. insert.php 파일이 write.php 파일로부터 넘겨받은 정보를 데이터베이스에 저장하는 기능을 하기 때문입니다.

전달 방식은 GET 방식이 아닌 POST 방법을 이용합니다. 게시판을 만들면서 반드시 POST 방식만을 사용해야 하는 것은 아니지만 GET 방식으로 할 경우 URL이 너무 지저분해지고 보기 싫어지니

POST 방식을 이용하도록 합니다. 또한 추후에 추가할 파일 업로드를 위해서도 POST 방식을 사용합니다.

정보를 넘겨받을 페이지를 설정했으니 이제 넘겨줄 항목들을 설정합시다. 우리가 넘겨줄 항목은 아래처럼 다섯 가지입니다.

① 글쓴이
② 이메일 주소
③ 글의 비밀번호
④ 글 제목
⑤ 글 내용

이 항목들을 어떤 이름으로 전달해야 할지를 정해야 합니다. 정보를 넘겨줄 때 받는 쪽에서 구분할 수 있게 이름을 붙여주는 것이지요. 이름은 영문과 지정된 특수문자 몇 개만 가능합니다. 이 규칙을 깨지 않고 기억하기 쉽고 항목을 쉽게 잘 표현하는 이름을 지으면 됩니다. 필자는 데이터베이스의 테이블 정보와 같은 이름을 짓도록 할 예정입니다. 괜히 다른 이름으로 작성하면 서로 헷갈릴 염려가 있으니까요.

항 목	이름	비 고
글쓴이	name	
이메일 주소	email	
글의 비밀번호	pass	입력 시 별표(*)로 표시
글 제목	title	
글 내용	content	

[표 11-7] 외부로부터의 변수 이름 설정

경우에 따라 데이터베이스의 테이블에 지정한 항목 이름과 같게 만들지 못하는 경우가 존재합니다. 예를 들어 하나의 값을 분할하여 입력받거나 한다면 같은 이름을 사용할 수 없을 것입니다.

이름을 정할 때는 알아보기 쉽게 정해주는 것이 무엇보다 중요합니다. 이름을 중복되지 않게 고유하게 짓는다고 asdajsdu23h89w8hefowah라고 정해보죠. 다음 페이지에서 고생할 겁니다. 물론 우리에겐 Copy & Paste 신공이 있기는 하지만요.

이름, 이메일, 비밀번호, 제목은 몇 자만 입력하면 되니까 INPUT 폼을 이용하고 내용은 몇 페이지가 될지 모르니 TEXTAREA 폼을 이용합니다. 특히 비밀번호는 입력 시에 별표로 표시되게끔 type을 password로 지정하는 것을 잊으면 안 됩니다.

```
<INPUT type=text name=name size=20>
<INPUT type=text name=email size=20>
```

```
<INPUT type=password name=pass size=10>
<INPUT type=text name=title size=60>
<TEXTAREA name=comment cols=65 rows=15></TEXTAREA>
```

무슨 말인지 모르겠다 하는 분들은 솔직히 손들고 HTML부터 공부하고 오기 바랍니다.

자 이제 전달될 항목 설정도 다 끝났으니 글 입력 버튼을 추가하면 됩니다.

```
<INPUT type=submit value="글 저장하기">
<INPUT type=reset value="다시 쓰기">
<INPUT type=button value="되돌아가기" onclick="history.back(-1)">
```

버튼이 세 가지가 있는데 첫 번째 버튼은 submit 버튼으로 이 버튼을 누르면 action에 기입된 파일로 폼 정보를 전달합니다. 두 번째 버튼은 reset 버튼으로 폼의 모든 항목을 초기화합니다. 마지막으로 되돌아가기 버튼을 만들고자 자바스크립트를 이용하여 되돌아가기를 구현했습니다.

onclick은 버튼이 클릭되었을 때 발생하는 이벤트로 자바스크립트를 실행하게 되어 있습니다. 따라서 위와 같이 history.back(-1);이라고 하면 바로 이전 페이지로 돌아가게 됩니다.

Section 05 글 쓰기 반영 – insert.php

앞서 write.php에서 글의 내용을 입력받았으니 전달받은 폼 정보를 실제 데이터베이스에 저장하는 부분을 만들어 봅시다. 앞에서 POST 방식으로 데이터를 전송했으니 $_POST[변수명]으로 값을 전달받을 수 있습니다.

우선 insert.php 파일은 데이터베이스에 접근하기 때문에 앞서 작성한 db_info.php가 필요합니다. 따라서 다음과 같이 db_info.php를 인클루드하기로 합니다.

```
include "db_info.php";
```

역시 짧게 한 줄로 끝나버리니 굉장히 편한 것 같습니다. 데이터베이스 연결 파일을 인클루드했으니 본격적으로 데이터를 입력하는 SQL 쿼리문을 작성해 보겠습니다.

새 글을 데이터베이스에 입력하는 것이니 우리가 사용해야 할 쿼리는 바로 INSERT입니다.

INSERT 문을 작성하기 위하여 입력해야 하는 값을 다시 한 번 살펴보도록 합시다.

항 목	이름
글의 번호	id
글쓴이	name
이메일 주소	email
글의 비밀번호	pass
글의 제목	title
글의 내용	content
글쓴 날짜	wdate
IP 주소	ip
조회 수	view

[표 11-8] board 테이블의 필드 정보

입력해야 하는 항목은 총 9개입니다. 이 중에 글쓴이, 이메일, 비밀번호, 제목, 내용은 사용자로부터 입력된 값이고 나머지 4개의 값은 자동으로 채워야 하는 값입니다.

자 그럼 쿼리를 만들어 볼까요?

```
INSERT INTO board (id, name, email, pass, title, content, wdate, ip,
view)
VALUES ('', '$name', '$email', '$pass', '$title', '$content', now(),
'$REMOTE_ADDR', 0);
```

쿼리를 완성했습니다. 쿼리에 대해 자세히 살펴보면 다음과 같습니다.

항 목	이름	저장된 값
글 번호	id	자동으로 1 증가 (입력 값 없음)
글쓴이	name	$name
이메일 주소	email	$email
글의 비밀번호	pass	$pass
글의 제목	title	$title
글의 내용	content	$content
글 쓴 날짜	wdate	now() – 현재 날짜와 시간 함수
IP 주소	ip	$REMOTE_ADDR – IP 정보를 가진 변수
조회 수	view	0 (최초 조회 수를 0으로 설정)

[표 11-9] 필드에 저장되는 값

여기서 now() 함수는 MySQL 내장 함수로서 현재 날짜와 시간 정보를 알려주는 함수입니다. 따라서 now() 라고 쿼리를 입력하면 현재 시간과 날짜를 자동으로 입력합니다. 또한 $REMOTE_ADDR은 PHP의 전역 변수로서 접속자의 IP 정보를 저장하고 있는 변수입니다.

완성된 쿼리는 $query 변수에 저장합니다.

```
$query = "INSERT INTO board (id, name, email, pass, title, content,
wdate, ip, view)
VALUES ('', '$name', '$email', '$pass', '$title', '$comment', now(),
'$REMOTE_ADDR', 0)";
```

변수에 저장된 쿼리는 mysql_query 함수를 이용하여 데이터베이스에 전달됩니다.

```
$result=mysql_query($query, $conn);
```

만약 쿼리에 문제가 없다면 데이터베이스에 INSERT 쿼리를 넘겨줬으니 해당 테이블에 새로운 글을 추가했을 것입니다. 쿼리에 문제가 있다면 에러를 출력할 것이고 쿼리에는 문제가 없는데 데이터베이스 내부에서 문제가 발생하였다면 에러 없이 입력되지 않습니다. 그 성공 여부는 $result 변수에 저장됩니다. 만약 $result 변수의 값이 FALSE라면 쿼리문을 다시 한 번 꼼꼼히 살펴보아야 할 것입니다.

두 줄로 너무나 간단하게 글 입력이 끝나 버렸습니다. 쿼리를 작성하고 mysql_query 함수를 이용하여 데이터베이스에 쿼리를 보내고 그 결과를 받는 것으로 말입니다.

이제 데이터베이스 작업이 끝났으니 데이터베이스와의 연결을 끊도록 합니다.

```
mysql_close($conn);
```

mysql_close 함수는 mysql_connect를 통해서 연결된 접속을 닫아버리는 함수입니다. 앞서 언급했듯이 데이터베이스는 PHP 페이지가 종료됨과 동시에 자동으로 연결이 종료되지만 되도록이면 데이터베이스 작업이 완료된 후 곧바로 닫아주는 버릇을 들이도록 합시다.

데이터베이스에 글이 저장되었으니 사용자에게 글이 제대로 잘 저장되었음을 알려줍니다.

```
echo ("<meta http-equiv='Refresh' content='1; URL=list.php'>");
```

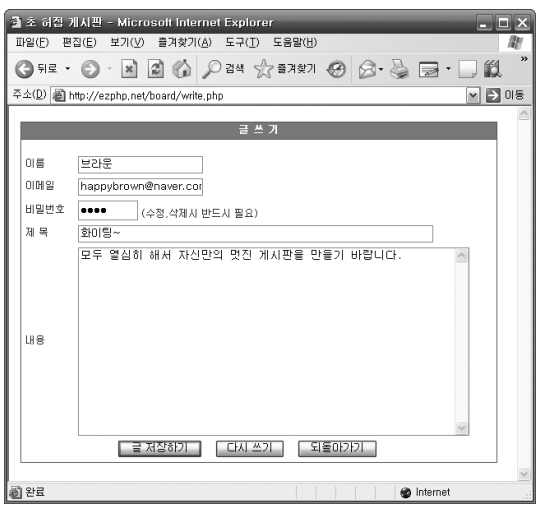
이 부분은 1초 후에 list.php 즉, 글 목록 페이지로 이동하겠다는 HTML 태그입니다. 1초 후에 이동하는 이유는 1초 동안 글이 제대로 저장되었음을 표시하기 위해서입니다.

[예제 11-1] insert.php 파일 소스 코드

```
1  <?
2     //데이터 베이스 연결하기
3     include "db_info.php";
4
5     $query = "INSERT INTO board
6     (id, name, email, pass, title, content,  wdate, ip, view)
7     VALUES ('', '$name', '$email', '$pass', '$title',
8     '$comment',  now(), '$REMOTE_ADDR', 0)";
9     $result=mysql_query($query, $conn);
10
```

```
11    //데이터베이스와의 연결 종료
12    mysql_close($conn);
13
14    // 새 글 쓰기인 경우 리스트로..
15    echo ("<meta http-equiv='Refresh' content='1; URL=list.php'>");
16 ?>
17 <center>
18 <font size=2>정상적으로 저장되었습니다.</font>
```

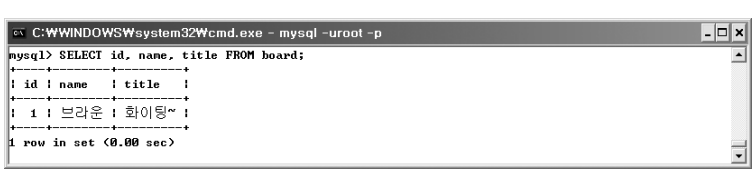
글을 입력하는 부분과 입력한 글을 데이터베이스에 저장하는 부분을 완성하였으니 테스트로 글을 하나 작성하여 입력해 보도록 합시다.



[그림 11-10] 테스트 글 입력

글 저장하기 버튼을 누르면 "정상적으로 저장되었습니다."라는 문구가 출력됩니다. 1초 후면 list.php로 이동하게 되지만 아직 파일을 작성하지 않았으니 파일이 없다는 에러를 보게 됩니다.

정말 제대로 데이터베이스에 입력되었는지 확인하기 위해 MySQL 클라이언트 프로그램을 통해서 확인해 보겠습니다.



[그림 11-11] 글 입력 확인

입력한 값대로 제대로 잘 입력이 되었죠?

사실 글을 저장하는 부분은 현재 문제가 아주 많습니다. 너무 순수하게 입력 부분만 작성했기 때문에 글을 입력하는 것에는 부족함이 없지만 때묻은 현실에서는 살아남기가 힘든 점이 많습니다. 다음 장에서 어떠한 문제가 있는지 하나하나 짚어가면서 수정해 나갈 예정입니다.

Section 06 글 수정 – edit.php

입력된 글은 수정해야 할 때가 많습니다. 특히 필자같이 글을 잘 못 쓰는 사람인 경우엔 더 한데요. 글을 써놓고 다시 한번 읽어보다가 수정하곤 하는 것이 습관이 되어버린 경우가 저 같은 경우입니다.

글을 수정하는 것은 글을 새로 쓰는 것과 매우 흡사합니다. 둘의 차이를 생각해보면 새로 쓰는 글은 모든 빈칸을 채워 주어야 하지만 수정하는 경우에는 이미 모든 칸이 채워져 있다는 것입니다. 채워져 있는 글에서 고치고 싶은 부분을 찾아내어 수정하는 것이 그 차이라고 할 수 있습니다.

그래서 HTML은 글 쓰기 폼과 거의 유사하며 입력된 글을 데이터베이스에서 읽어와서 해당 폼에 값을 출력하는 일을 추가로 해주어야 합니다. 따라서 글 쓰기 폼 (write.php) 파일의 소스 코드를 그대로 사용할 예정입니다. write.php 파일의 이름을 edit.php 파일로 변경하든지 아니면 소스 코드를 복사해서 edit.php 파일을 생성하도록 합니다. 그런 다음 복사한 edit.php 파일의 내용을 알맞게 수정하여 글 수정하기 페이지를 만들겠습니다.

우선 폼의 내용이 변경되는 부분을 살펴봅니다.

```
<form action=update.php?id=<?=$id?> method=post>
```

제일 먼저 새로운 action을 설정합니다. 글 쓰기에서는 insert.php 파일로 폼 정보를 전달하면 되었지만 글 수정하기에서는 update.php 파일로 수정된 정보를 전달해야 합니다. 따라서 위와 같이 update.php로 전달하도록 설정합니다. 그런데 여기서 update.php 파일명 뒤에 GET 방식으로 id 값을 전달하는 부분이 추가되어 있음을 알 수 있습니다. 수정할 때 id값을 전달해주는 이유는 update.php 파일에서 수정해야 할 글의 번호를 알 수 있게 하기 위해서입니다. 글의 번호를 전달해주지 않는다면 update.php 파일에서는 수정된 폼 정보를 어떤 글에 적용해야 하는지 알 수가 없을 것이기 때문입니다.

그렇다면 id 값은 어디에서 넘어올까요? 글을 수정할 때는 글 읽기 페이지의 글 수정하기 버튼을 통해서 글 수정하기 페이지로 이동해 옵니다. 이때 읽고 있던 글의 번호를 같이 넘겨주는 것입니다. update.php에 id 값을 넘겨주는 것과 마찬가지로 edit.php 페이지에서도 수정할 글을 데이터베이스에서 읽어와야 하기 때문에 글 번호가 반드시 필요합니다.

폼 정보를 수정하였으니 이제 본격적으로 데이터베이스에서 해당 글의 정보를 가져오도록 합시다.

```
include "db_info.php";
```

데이터베이스를 사용하는 페이지에는 반드시 위처럼 db_info.php 파일을 인클루드해야 한다고 했습니다. (잊지 않으셨죠?)

이제 기존 글의 내용을 불러올 차례입니다. 그렇다면 데이터베이스에서 글의 내용을 가져오는 쿼리를 작성해 보겠습니다. 어떤 쿼리를 써야 할까요? 네. 맞습니다. SELECT 쿼리를 사용하면 됩니다. 아마도 성격이 급해서 UPDATE라고 생각하신 분들도 계실 겁니다. 너무 앞서가지는 마세요. 지금은 데이터베이스에서 정보를 가져오는 부분을 하고 있지 수정하는 부분을 하는 것이 아닙니다. 만약 대답으로 INSERT나 DELETE라고 하신 분이 계시다면 죄송하지만 영어부터 다시 공부하는 것을 권해 드립니다. 후훗.

SELECT 문을 다시 한 번 떠올려 봅시다.

```
SELECT 출력될 항목들 FROM 테이블이름 WHERE 검색조건
```

위와 같은 방법으로 쿼리를 작성하면 됩니다. 출력될 항목들을 일일이 명시하기 귀찮다고 생각하는 분은 다음과 같이 모든 항목을 가져오게 해도 무관합니다.

```
SELECT * FROM 테이블이름 WHERE 검색조건
```

두 번째 방법이 간단하고 편리할 때가 많이 있습니다. edit.php 페이지에서처럼 대부분의 항목을 가져와야 하는 경우 일일이 항목을 나열하기보다는 편의를 위하여 모든 항목을 가져오게 하는 것이 더 나은 경우가 많습니다. 그러나 불필요한 항목에 대한 정보를 가져오게 되므로 불필요한 부하가 걸리게 되니 유념하기 바랍니다.

많은 글 중에서 수정해야 할 특정 글을 선택해서 가져와야 하니 검색 조건을 반드시 넣어주어야 할 것입니다. 어떤 조건을 주면 해당 글을 바로 검색해 가져올 수 있을까요? 그것은 바로 절대 중복되지 않는 유일한 글의 번호를 통해서 가능합니다.

```
SELECT * FROM board WHERE id=$id;
```

가장 기본적인 SQL 쿼리이군요. 데이터베이스 장을 열심히 공부하신 분이라면 이미 쿼리를 쉽게 떠올렸을 것으로 생각합니다. 혹시 모르는 분을 위해 다시 한번 설명해 드리면 id 값이 $id 변수의 값과 일치하는 글을 찾아서 모든 항목을 전해 달라는 의미입니다. id는 글의 번호로 유일한 값이니 만약 해당하는 글이 있다면 딱 한 글만 검색될 것입니다.

이제 쿼리를 작성했으니 $query 변수에다 저장합니다.

```
$query = "SELECT * FROM board WHERE id=$id";
```

mysql_query 함수를 통해서 데이터베이스에 쿼리를 전달합니다.

```
$result=mysql_query($query, $conn);
```

이처럼 짧은 쿼리를 전달할 때는 $query 변수를 사용하지 않고 다음과 같이 직접 쿼리를 던지기도 합니다.

```
$result=mysql_query("SELECT * FROM board WHERE id=$id", $conn);
```

단순히 변수를 사용하느냐 안 하느냐의 차이일 뿐 특별히 다른 방법이라고는 할 수 없겠지만 쿼리가 길어지면 변수를 통해서 쿼리를 임시로 저장해두는 것이 바람직합니다.

쿼리를 데이터베이스에 던져주었으니 데이터베이스가 검색한 결과를 돌려주겠죠? 검색된 결과를 가져와서 출력하는 부분을 작성해 보도록 합시다.

```
$row=mysql_fetch_array($result);
```

mysql_fetch_array 함수를 이용하여 검색된 결과를 배열 형태로 받아옵니다. mysql_fetch_array 함수는 테이블의 필드 이름을 통해 검색된 결과인 배열에서 원하는 항목을 찾아낼 수 있습니다. 쿼리에서 모든 항목을 요구했기 때문에 $row[id], $row[name] 혹은 $row[0], $row[1]과 같은 방법으로 검색된 결과를 가져올 수 있습니다. 그리고 나서 가져온 결과를 필요한 부분에 알맞게 출력하면 됩니다.

mysql_fetch_array 함수와 거의 같은 함수가 있습니다. 바로 mysql_fetch_row 함수인데요. 두 함수의 차이는 mysql_fetch_array 함수가 결과를 위에서처럼 필드 이름이나 필드 순서(여기서 필드 순서는 SELECT * FROM table인 경우는 테이블의 내의 필드 순서를 의미하고 SELECT field1, field2 FROM table인 경우에는 필드 이름이 적힌 순서 대로를 의미합니다.)를 통해서 가져올 수 있는 반면 mysql_fetch_row 함수는 필드 순서만 사용할 수 있습니다.

테이블을 2개 이상 JOIN하는 경우 두 테이블에 같은 이름의 필드가 존재할 수 있습니다. 이때 두 테이블의 필드를 모두 가져와야 하는 경우 두 필드가 구분되지 않기 때문에 필드 순서로 인덱싱하거나 아니면 필드 이름에 별명을 붙여주는 방법을 사용해야 합니다. 필드 이름에 별명을 붙이는 방법은 데이터베이스 장을 참고하십시오.

필자가 처음 PHP를 배웠을 때는 mysql_fetch_row 함수를 습관적으로 사용했는데 (공부하던 책에서 mysql_fetch_row 함수를 사용해서일까요?) 불편한 점이 많았습니다. 예를 들어 쿼리문을 변경하면 순서가 변경될 수 있기 때문에 일일이 수정을 해주어야 했습니다. 하지만 필드 이름을 사용하면 필드 이름이 변경되지 않는 한 쿼리가 변경된다고 해서 소스 코드를 수정할 필요가 없습니다.

여기서 잠깐

병역특례로 회사에 다닐 때 있었던 이야기입니다. 퇴사한 개발자 분이 홈페이지 템플릿을 만들어 두었더군요. 여기서 홈페이지 템플릿이란 싸이월드나 네이버 블로그와 같이 찍어내듯 단순한 홈페이지를 만들어 낼 수 있는 프로그램입니다. 그런데 홈페이지 내용을 수정하기 위해서 만들어 놓은 관리자 페이지의 로그인 부분에 비밀번호를 HIDDEN 폼으로 숨겨두었습니다. 다음 페이지에서 사용자가 입력한 비밀번호와 HIDDEN으로 숨겨둔 값을 비교하여 로그인을 검증하는 방식이었습니다. HTML을 조금이나마 아는 사용자는 우연히 웹 페이지 소스 보기를 통해서 쉽게 비밀번호를 알아낼 수 있었을 것입니다.

이분의 의도는 아마도 데이터베이스를 두 번 접근하지 않고 한 번 만에 접근하여 로그인을 구현하려던 것이 아닌가 싶습니다. 두 번의 데이터베이스 접근이란 한 번은 로그인 페이지에서 사용자의 정보를 출력하고자 필요한 것이고 또 한 번은 로그인을 검증하는 페이지에서 비밀번호를 비교하기 위한 접근인데 이를 한 번으로 줄이고 싶었나 봅니다. 한 번은 무조건 사용자의 정보를 출력하기 위해 필요하니 이때 비밀번호도 같이 가져와서 한 번에 해결해 보자고 생각하셨겠죠. 그래서 눈에는 보이지 않게 숨겨진 폼을 사용하여 비밀번호를 숨겨둔 것일 겁니다. 하지만 앞에서도 말씀드렸듯이 누구나 쉽게 비밀번호를 알아낼 수 있다는 것이 큰 문제입니다.

Q 여기서 문제! 위와 같은 경우 한 번의 데이터베이스 접속으로 로그인을 구현할 수도 있습니다. 위험천만한 위의 방법보다 훨씬 안전하게 말이죠. 어떤 방법으로 가능할까요?

이 문제는 상식을 동원해야 합니다. 소스 코드를 떠올리지 말라는 말씀입니다. 힌트를 드리자면 로그인 페이지와 로그인 검증 페이지 중에서 로그인 페이지에서는 한 번의 데이터베이스 접속이 필요합니다. 그렇다면 로그인 검증 페이지에서 어떻게 데이터베이스 접속 없이 비밀번호를 판단할 수 있느냐는 문제로 바꿀 수 있겠네요. 힌트를 더 드리자면 위의 위험천만한 방식을 약간 변형하면 가능합니다. 자. 어떻게 하면 한 번의 접속으로 로그인을 구현할 수 있을까요?

정답은 "비밀번호를 암호화해서 숨겨둔다"입니다. 비밀번호를 보아도 아무도 알 수 없게끔 만들어 버리면 되는 것이죠. 필자가 자주 이용하는 방법을 알려 드리면 MD5라는 해시 함수를 이용하여 쉽게 처리할 수 있습니다. MD5 해시 함수는 암호화된 암호문을 보고 원문(실제 비밀번호)을 절대 알아낼 수 없습니다. 아니 원래 비밀번호를 알아낼 수 없는데 어떻게 비교를 합니까? 사용자가 입력한 비밀번호를 다시 MD5 함수로 암호화를 해서 비교해보면 됩니다.

로그인 검증의 목적은 비밀번호가 같은 것인가를 확인하는 것이지 비밀번호가 무엇인가가 목적이 아닙니다. 로그인 검증에서 비밀번호가 어떤 것이든 상관없이 사용자가 입력한 값이랑 같은지 다른지만 판단하면 되기 때문에 비밀번호도 MD5로 암호화하고 사용자가 입력한 비밀번호도 암호화해서 암호화된 암호문이 같은지만 확인하면 됩니다.

가벼운 마음으로 풀어 보셨나요? 정답을 맞히셨다면 웹 프로그래머로서의 자질이 충분하다 못해 넘치고 넘치는 분이십니다. 그렇다고 못 맞추셨다고 자질이 없는 것은 아니랍니다. 원래 같은 것을 배워도 꼭 튀는 분이 있답니다. 무시하세요. 하하하.

그러나 때로는 필드 순서로 작업하는 것이 편한 경우가 있는데요, 순서대로 모든 값을 출력해야 하는 경우에는 for 문으로 쉽게 출력할 수 있기 때문에 mysql_fetch_row 함수를 사용할 때가 있습니다. 그렇지만 mysql_fetch_array 함수도 필드 순서대로 출력할 수 있다는 거죠.

이제 데이터베이스에서 글 정보를 모두 가져왔으니 필요한 곳에 적절히 출력하면 됩니다. 단, 유념해야 할 부분은 비밀번호를 출력해서는 안 된다는 것입니다.

데이터베이스로부터 가져온 글의 내용을 출력해 보도록 합시다. 출력해야 할 항목은 입력하는 폼 중에서 비밀번호를 제외한 모든 항목입니다. 따라서 name, email, title, content 이렇게 4가지 항목이며 출력하는 방법은 다음과 같습니다.

```
<INPUT type=text name=name size=20 value='<?=$row[name]?>'>
```

그냥 일반적인 텍스트로 출력할 때는 〈?=$row[name]?〉처럼 출력하면 되지만 폼에 글자가 보이게 출력해야 하므로 value 값에 넣어주는 것을 잊으면 안됩니다. 나머지 항목을 적절히 출력하면 다음과 같습니다.

```
<INPUT type=text name=email size=20 value='<?=$row[email]?>'>
<INPUT type=text name=title size=60 value='<?=$row[title]?>'>
<TEXTAREA name=comment cols=65 rows=15><?=$row[comment]?></TEXTAREA>
```

여기서 TEXTAREA 태그는 value 값을 지정하는 것이 아니라 TEXTAREA 태그 사이에 출력해야 합니다. 그리고 비밀번호 부분은 그냥 빈칸으로 두는 걸 잊지 않으셨죠?

이제 글 저장하기 버튼을 클릭하면 update.php 파일로 수정된 글 정보가 전달될 것입니다.

완성된 edit.php 파일의 소스 코드는 다음과 같습니다.

edit.php

```
1  <html>
2  <head>
3  <title>초 허접 게시판</title>
4  <style>
5  <!--
6  td { font-size : 9pt; }
7  A:link { font : 9pt; color : black; text-decoration : none;
8  font-family: 굴림; font-size : 9pt; }
9  A:visited { text-decoration : none; color : black;
10 font-size : 9pt; }
11 A:hover { text-decoration : underline; color : black;
12 font-size : 9pt;}
13 -->
14 </style>
```

```
15 </head>
16
17 <body topmargin=0 leftmargin=0 text=#464646>
18 <center>
19 <BR>
20 <!-- 입력된 값을 다음 페이지로 넘기기 위해 FORM을 만든다. -->
21 <form action=update.php?id=<?=$id?> method=post>
22 <table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
23   <tr>
24     <td height=20 align=center bgcolor=#999999>
25       <font color=white><B>글 수 정 하 기</B></font>
26     </td>
27   </tr>
28 <?
29   //데이터 베이스 연결하기
30   include "db_info.php";
31
32   // 먼저 쓴 글의 정보를 가져온다.
33   $result=mysql_query("SELECT * FROM board WHERE id=$id", $conn);
34   $row=mysql_fetch_array($result);
35 ?>
36 <!-- 입력 부분 -->
37   <tr>
38     <td bgcolor=white> 
39     <table>
40       <tr>
41         <td width=60 align=left >이름</td>
42         <td align=left >
43           <INPUT type=text name=name size=20
44           value=<?=$row[name]?>>
45         </td>
46       </tr>
47       <tr>
48         <td width=60 align=left >이메일</td>
49         <td align=left >
50           <INPUT type=text name=email size=20
51           value=<?=$row[email]?>>
52         </td>
53       </tr>
54       <tr>
55         <td width=60 align=left >비밀번호</td>
56         <td align=left >
57           <INPUT type=password name=pass size=8>
58           (비밀번호가 맞아야 수정가능)
59         </td>
60       </tr>
61       <tr>
```

```
62          <td width=60 align=left >제 목</td>
63          <td align=left >
64            <INPUT type=text name=title size=60
65            value=<?=$row[title]?>>
66          </td>
67        </tr>
68        <tr>
69          <td width=60 align=left >내용</td>
70          <td align=left >
71            <TEXTAREA name=content cols=65 rows=15>
72            <?=$row[content]?></TEXTAREA>
73          </td>
74        </tr>
75        <tr>
76          <td colspan=10 align=center>
77            <INPUT type=submit value="글 저장하기">
78              
79            <INPUT type=reset value="다시 쓰기">
80              
81            <INPUT type=button value="되돌아가기"
82            onclick="history.back(-1)">
83          </td>
84        </tr>
85        </TABLE>
86      </td>
87    </tr>
88 <!-- 입력 부분 끝 -->
89 </table>
90 </form>
91 </center>
92 </body>
93 </html>
```

Section 07

글 수정 반영 – update.php

수정된 글을 저장하는 것에 대해 생각해 봅시다. 글 하나는 중복되지 않게 딱 하나의 번호를 가져야 합니다. 그런데 수정된 글을 글 쓰기와 동일하게 같은 번호로 추가한다면 문제가 생길 것이 분명합니다. 기존 글과 새로 수정한 글이 구분되지 않을 테니까요.

그래서 UPDATE 문을 사용하여 수정하고자 하는 필드만을 수정해야 합니다.

우선 쿼리를 쓰려면 데이터베이스에 접속부터 해야겠지요?

```
include "db_info.php";
```

db_info.php 파일을 인클루드하는 것은 이젠 거의 자동으로 되리라 생각합니다. 그렇다면 다음은? 네~. 쿼리를 작성해야죠.

```
$result=mysql_query("SELECT pass FROM board WHERE  id=$id", $conn);
```

뭔가 이상한가요? 글을 수정하는 쿼리가 나와야 하는데 SELECT 문이라니. 이 부분은 해당 글의 비밀번호를 가져 오는 부분입니다. 일단 글을 수정하기 전에 실제 글쓴이가 맞는지부터 확인해야 할 테니까요.

이전 페이지에서 사용자가 입력한 비밀번호를 전달했습니다. 수정 전 글의 비밀번호와 입력된 비밀번호가 같은지를 체크해서 글쓴이가 맞는지 확인할 수 있습니다. 그렇다면 비밀번호를 가져와 볼까요?

```
$row=mysql_fetch_array($result);
```

$row 변수에 배열 형태로 비밀번호가 잘 저장되어 있을 것입니다. 이제 입력한 비밀번호와 일치하는지를 비교하면 되겠습니다.

```
if ($pass==$row[pass]) { //비밀번호가 일치하는 경우
//글을 수정하는 코드
}
else { // 비밀번호가 일치하지 않는 경우
//에러 메시지를 출력하는 코드
}
```

비밀번호가 일치한다면 글을 수정하는 쿼리를 실행하게 하고 그렇지 않으면 비밀번호가 틀렸음을 알리는 에러 메시지를 출력하면 됩니다. 글의 비밀번호를 비교하는 방법은 한 가지만 있는 것이 아닙니다. 다양한 방법이 있을 수 있는데 위와 같은 방법은 그 중 하나일 뿐입니다.

비밀번호를 검증하였으니 비밀번호가 일치하는 경우 글을 수정하는 쿼리를 작성해 봅시다. 수정해야 할 필드는 name, email, title, content이고 각각 $name, $email, $title, $content 변수에 저장되어 있습니다.

```
UPDATE board SET name='$name', email='$email',
title='$title', content='$content'
```

위와 같이 SQL 문을 작성하면 큰 낭패를 봅니다. 왜일까요? 조건절이 없으므로 모든 글을 수정하기 때문입니다. 실제로도 이런 실수를 자주 하게 되는데 SELECT와 INSERT를 제외한 나머지

UPDATE, DELETE 쿼리는 사용 시에 반드시 조건절을 제대로 작성하였는지 확인하는 버릇을 들여야 합니다. DELETE 쿼리에 조건절을 잊어버리면 모든 글이 삭제됩니다.

제대로 조건절을 추가해서 쿼리문을 작성해보면 다음과 같습니다.

```
UPDATE board SET name='$name', email='$email', title='$title',
content='$content' WHERE id=$id
```

쿼리가 작성이 되었으니 변수에 저장하고 데이터베이스에 전달해 봅시다.

```
if ($pass==$row[pass]) { //비밀번호가 일치하는 경우
  $query = "UPDATE board SET name='$name', email='$email',
  title='$title', content='$content' WHERE id=$id";
  $result = mysql_query($query, $conn);
}
```

UPDATE 쿼리의 경우에는 INSERT, DELETE 쿼리와 마찬가지로 mysql_query 함수를 통해 데이터베이스에 쿼리문을 전송하면 끝이 납니다. SELECT 쿼리는 가져온 정보를 처리하는 것을 생각하면 편한 쿼리가 아닐 수 없습니다.

이제 남은 것은 에러 메시지 출력과 글이 잘 저장되었음을 알리는 것입니다. 우선 자바스크립트를 이용하여 비밀번호가 틀렸음을 경고창을 통해서 알립니다.

```
else { // 비밀번호가 일치하지 않는 경우
echo ("
<script>
alert('비밀번호가 틀립니다.');
history.back();
</script>
");
}
```

자바스크립트를 통해서 비밀번호가 틀렸음을 알았습니다. 그러나 실제로 이 코드를 사용해보면 문제가 있음을 알게 됩니다. 비밀번호가 틀리다는 메시지가 경고창으로 뜨게 되고 확인 버튼을 누름과 동시에 이전 페이지로 이동하게 되어 있습니다. 이런 경우 '이제 이 이후의 코드는 작동하지 않겠군'하고 생각하기 쉬운데 실제로는 그렇지 않기 때문입니다. 실제로 페이지가 이동하기 이전에는 그 아랫부분의 코드가 동작해버립니다. 만약 else 밖에 데이터베이스 관련 코드가 존재한다면 이것이 실행될 가능성이 매우 높습니다. 따라서 페이지가 더는 처리되지 않게 중단하고 싶을 때는 반드시 다음과 같이 exit를 사용해야 합니다.

```
else { // 비밀번호가 일치하지 않는 경우
echo ("
<script>
alert('비밀번호가 틀립니다.');
```

```
history.back();
</script>
");
exit;
}
```

다음은 데이터베이스 연결을 종료하고 1초 뒤에 글 읽기 페이지로 이동하게끔 설정한 뒤 글이 제대로 수정되었음을 출력하면 됩니다. 여기서 글 읽기 페이지로 이동하는 이유는 글의 수정이 모두 글 읽기 페이지로부터 시작되기 때문입니다. 글을 읽다가 수정할 부분이 생겨서 글 수정하기를 누르고 글을 수정한 뒤 제대로 수정이 되었는지 수정된 글을 다시 보여주어야 하는 것입니다.

```
echo ("<meta http-equiv='Refresh' content='1; URL=read.php?id=$id'>");
```

이때 돌아가는 글 읽기 페이지에 잊지 않고 글 번호를 전달해 줍니다. 그 이유는 누누이 말씀드려왔지만 많은 글 중에서 수정된 글을 읽게 하려면 글의 번호가 필요하기 때문입니다.

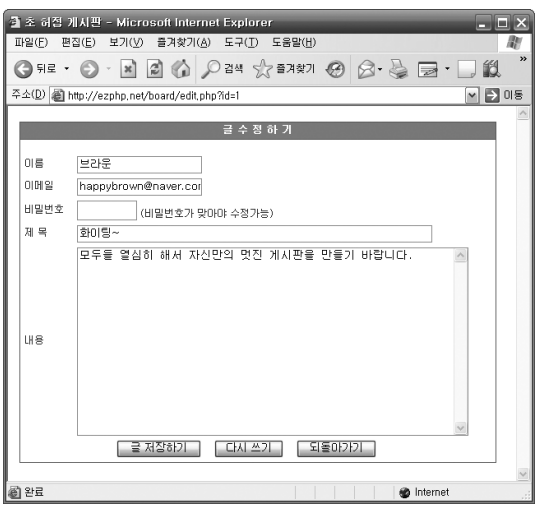
update.php의 완성된 소스 코드는 다음과 같습니다.

update.php

```
1  <?
2    //데이터 베이스 연결하기
3    include "db_info.php";
4
5    // 글의 비밀번호를 가져온다.
6    $query = "SELECT pass FROM board WHERE id=$id";
7    $result=mysql_query($query, $conn);
8    $row=mysql_fetch_array($result);
9
10   //입력된 값과 비교한다.
11   if ($pass==$row[pass]) { //비밀번호가 일치하는 경우
12     $query = "UPDATE board SET name='$name', email='$email',
13     title='$title', content='$content' WHERE id=$id";
14     $result=mysql_query($query, $conn);
15   }
16   else { // 비밀번호가 일치하지 않는 경우
17     echo ("
18     <script>
19     alert('비밀번호가 틀립니다.');
20     history.go(-1);
21     </script>
22     ");
23     exit;
24   }
25
```

```
26    //데이터베이스와의 연결 종료
27    mysql_close($conn);
28
29    //수정하기인 경우 수정된 글로..
30    echo ("<meta http-equiv='Refresh' content='1;
31    URL=read.php?id=$id'>");
32  ?>
33  <center>
34  <font size=2>정상적으로 수정되었습니다.</font>
```

글 수정하기가 완성되었으니 실제로 잘 동작하는지 테스트를 해보도록 합시다.

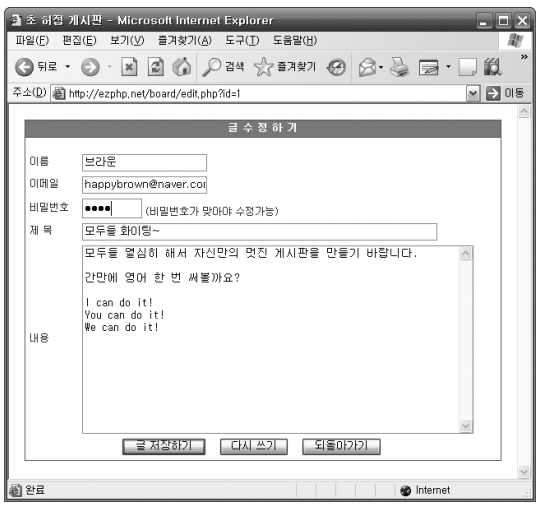


[그림 11-12] 글 수정하기

이 페이지를 보려면 번거롭지만 웹 브라우저 주소창에 다음과 같이 입력을 해야 볼 수 있습니다.

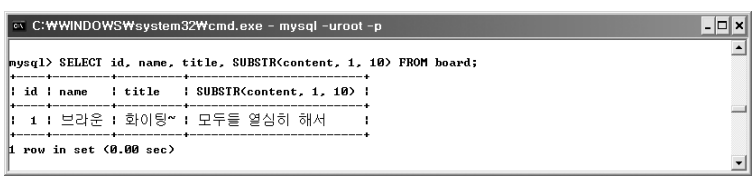
```
http://도메인/edit.php?id=1
```

이 글은 앞서 글 쓰기 부분에서 우리가 처음 작성한 글이므로 글 번호가 1번일 것입니다. 이제 다음과 같이 몇 가지 내용을 수정한 다음에 글 저장하기를 눌러봅니다.



[그림 11-13] 글 수정 중

수정된 결과를 phpmyadmin을 통해서 확인해 보도록 하겠습니다.



[그림 11-14] 수정된 글 정보 확인

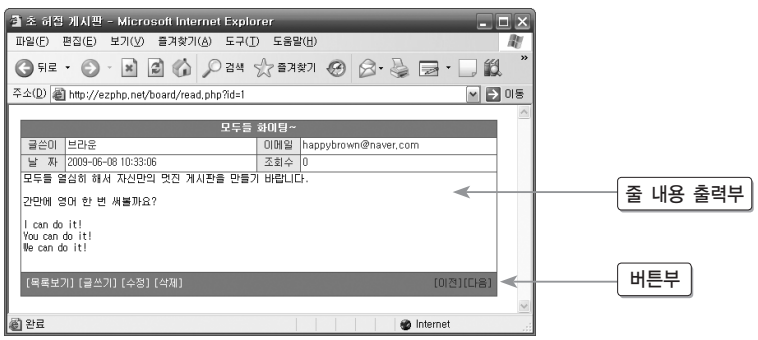
제대로 잘 수정이 되었나요? 여기서 SUBSTR 함수는 PHP의 substr 함수와 마찬가지로 문자열을 자르는 기능을 하는 MySQL 내장 함수입니다. content 필드가 너무 긴 관계로 10글자로 줄여보았습니다.

수정하기 부분은 글 쓰기 부분을 대부분 가져다가 사용할 수 있었습니다. 다른 점이라고 해봐야 글 쓰기 부분에 이미 입력된 값을 보여주는 것과 수정을 하기 위한 쿼리가 변경된 것이 전부였습니다. 좀 수월해지셨나요? 꼭 그렇길 바랍니다.

Section 08

글 읽기 – read.php

글 읽기 페이지는 다음 그림과 같이 저장된 글을 보기 좋게 배치하여 출력하는 페이지입니다.



[그림 11-15] 글 읽기

그다지 보기 좋지만은 않군요. 글 읽기 파일은 크게 두 부분으로 나눌 수 있습니다. [그림 11-15]에서 보이듯이 글의 내용을 출력하는 부분과 목록보기, 글쓰기 등과 같은 버튼 부분입니다. 우선 글의 내용을 출력하는 부분에 대해서 살펴보겠습니다. 글의 내용을 가져와야 하니까 자연스럽게(?) db_info.php를 인클루드시킵니다.

```
include "db_info.php";
```

자연스럽나요? 당연히 자연스러워야 합니다. 다음은 글의 내용을 가져오는 쿼리를 작성하는 것입니다. 이 부분은 앞서 글 수정하기에서 다루었으니 쉽게 넘어가 보겠습니다.

```
// 글 정보 가져오기
$result=mysql_query("SELECT * FROM board WHERE id=$id", $conn);
$row=mysql_fetch_array($result);
```

데이터베이스에서 가져온 글 정보가 모두 $row 변수에 배열로 저장되었습니다. 이제 적절한 곳에다 항목들을 출력합니다.

```
<table width=580 border=0  cellpadding=2 cellspacing=1
bgcolor=#777777>
<tr>
    <td height=20 colspan=4 align=center bgcolor=#999999>
        <font color=white><B><?=$row[title]?></B></font>
    </td>
</tr>
<tr>
```

```
        <td width=50 height=20 align=center bgcolor=#EEEEEE>글쓴이</td>
            <td width=240 bgcolor=white><?=$row[name]?></td>
        <td width=50 height=20 align=center bgcolor=#EEEEEE>이메일</td>
            <td width=240 bgcolor=white><?=$row[email]?></td>
    </tr>
    <tr>
        <td width=50 height=20 align=center bgcolor=#EEEEEE>
            날   짜</td>
        <td width=240 bgcolor=white><?=$row[wdate]?></td>
        <td width=50 height=20 align=center bgcolor=#EEEEEE>조회 수</td>
            <td width=240 bgcolor=white><?=$row[view]?></td>
    </tr>
    <tr>
        <td bgcolor=white colspan=4>
            <font color=black>
                <pre><?=$row[content]?></pre>
            </font>
        </td>
    </tr>
```

여기서 본문 내용을 출력하는 부분을 주의 깊게 볼 필요가 있습니다.

```
<pre><?=$row[content]?></pre>
```

다른 부분과 다르게 〈pre〉 태그를 사용하고 있는데요, 그 이유는 사용자가 입력한 그대로 출력하기 위해서입니다. 사용자가 입력할 때는 줄 바꿈을 엔터로 구분하지만 HTML에서는 엔터가 아무 의미가 없기 때문에 출력 시 줄 바꿈이 무시되어 출력됩니다. 따라서 이것을 입력할 때와 같은 모양으로 출력해야 하는데 가장 간단하게 이를 구현하는 방법이 〈pre〉 태그를 이용하는 것입니다.

이제 글의 내용을 제대로 출력했으니 각종 버튼을 구현해 보도록 합시다. 각 버튼은 모두 링크로 구성되어 있습니다. 아래 표를 통해 각 버튼이 어떤 링크를 가지는지 알아봅시다.

버 튼	기 능	넘겨주는 값 (변수)
목록보기	글의 목록 페이지로 이동	list.php?no=$no , 페이지 구분을 위한 값
글쓰기	글 쓰기 페이지로 이동	write.php, 전달 값 없음
수정	글 수정 페이지로 이동	edit.php?id=$id, 수정될 글 번호를 넘겨줌
삭제	글 삭제 페이지로 이동	predel.php?id=$id, 삭제될 글 번호를 넘겨줌
이전	이전 글이 있다면 링크	read.php?id=이전 글 번호, 이전 글 번호를 넘겨줌
다음	다음 글이 있다면 링크	read.php?id=다음 글 번호, 다음 글 번호를 넘겨줌

[표 11-10] 각 버튼의 링크 정보

글쓰기는 어차피 새로운 글을 쓰는 것이라 넘겨주는 값이 없습니다. 수정과 삭제는 현재 글에 대한 수정과 삭제이므로 현재 글 번호를 해당 파일로 전달해야 합니다. 목록보기는 $no 변수값을 넘겨주

는데 $no 변수는 목록 강좌에서 자세히 배우겠지만 페이지를 구분하는 인자가 되는 값입니다. 즉, $no 값에 따라 첫 번째 페이지로 돌아갈 것인지 아니면 두 번째 페이지로 돌아갈 것인지가 결정됩니다.

그렇다면 목록으로 돌아갈 때는 몇 번째 페이지로 돌아가야 할까요? 일반적으로 현재 글을 보기 전에 있던 목록으로 돌아가는 것이 맞습니다. 만약 첫 번째 페이지부터 순서대로 글을 읽어가고 있는데 10번째 페이지를 읽고 있다고 생각해 봅시다. 이때 글을 다 읽고 다음 글을 읽으려고 목록 보기를 눌렀다면 첫 번째 페이지를 보게 될 것이고 다시금 10번째 페이지를 찾아와야 할 것입니다.

그러면 해당 글이 몇 번째 페이지인지는 어떻게 알 수 있을까요? 데이터베이스에 검색해서 몇 번째 글인지 파악한 다음에 페이지 수를 계산할 수도 있겠지만 너무 번거롭습니다. 다음과 같은 방법으로 쉽게 페이지를 기억할 수 있습니다.

[그림 11-16] 페이지와 글 번호 정보 전달

페이지 정보는 위와 같이 글을 읽으려고 글의 제목을 클릭하면 글 번호와 함께 전달됩니다. 글 읽기 페이지에서 no 값을 $no 변수로 전달받게 되니 글을 읽고 나서 목록보기를 누른다면 이 $no 변수 값을 통해 다시 목록 페이지로 돌아오게끔 될 것입니다.

조금 까다로운 버튼인 이전과 다음 버튼은 현재 글을 기준으로 이전 글과 다음 글의 번호를 알아내는 것이 중요합니다. 일단 이전과 다음의 기준을 먼저 정해야 하는데, 첫 번째 페이지의 첫 글부터 읽는다고 했을 때 아래쪽의 글이 다음 글이 되고 위쪽의 글이 이전 글이 됩니다. 이전이라고 하면 시간 순서의 의미로 판단하여 현재의 글보다 오래된 글이라고 생각할 수 있어 혼동하는 경우가 많이 있습니다. 하지만 여기서 이전이라는 의미는 글 목록에서의 순서를 의미하며 현재 글의 위쪽에 있는 글을 의미합니다.

그렇다면 이전 글과 다음 글은 어떻게 알 수 있을까요? 글 번호가 1씩 순차적으로 증가한다고 했으니 현재 글 번호를 기준으로 ±1씩 하면 되겠습니다. 이 방법은 너무 간단하고 좋은 반면에 극악무도한 복병이 있으니 그것이 바로 삭제입니다. 글을 삭제해 버린다면 그 번호는 사라져 버립니다. 따라서 다음 글이 되었을 글이 이미 삭제되고 없다면 없는 글을 읽는 상태가 됩니다.

자~ 어떻게 하면 이 문제를 해결할 수 있을까요? 답을 보기 전에 한 번쯤 생각해보기 바랍니다.

현재 글을 기준으로 이전 글은 현재 글보다 목록에서 위에 있는 글이니 현재 글 번호보다 큰 번호를 가지는 글일 것입니다. 또한 다음 글은 현재 글보다 목록의 아래에 있는 글이니 현재 글 번호보다는

작은 값을 가지는 글이 될 것입니다. 그렇다면 현재 글 바로 이전의 글과 바로 다음의 글이란 것은 다음과 같이 생각할 수 있습니다.

이전(위) 글: 현재 글 번호보다는 크되 번호가 큰 글 중에서 가장 작은 번호의 글
다음(아래) 글: 현재 글 번호보다는 작되 번호가 작은 글 중에서 가장 큰 번호의 글

한 번 맞는지 곰곰이 생각해 보십시오. 맞는 것 같습니까? 한 번 봐서는 잘 이해가 안 될 수 있으니 몇 번에 걸쳐서 읽어보셔야 차츰 이해할 수 있을 것입니다.

그러면 이제 위의 정의대로 이전 글과 다음 글을 구현하면 됩니다. 방법은 여러 가지가 있을 수 있는데 다음 소스는 그 중 한 가지 방법을 제시하고 있습니다. (최선의 선택은 아닙니다.)

```
//현재 글보다 id 값이 큰 글 중 가장 작은 것을 가져온다. 즉 바로 이전 글
$query=mysql_query("SELECT id FROM board WHERE id > $id LIMIT 1",
$conn);
```

여기서 LIMIT 1은 한 개의 결과만 가져오라는 뜻입니다. $id 값보다 큰 글이 여러 개가 있을 수도 있지만 큰 글 중 결과 하나만을 검색하라는 말입니다. 현재 글보다 id 값이 큰 것 중에서 가장 작은 id 값을 가지고 있다는 설명이 부족한데 여기서는 LIMIT 1이 그 역할까지 해줍니다. 이 쿼리는 우선 ORDER BY id ASC를 함축하고 있습니다. 데이터베이스 장에서 이미 배웠지만 ORDER BY 구문을 작성하지 않으면 키 값을 기준으로 오름차순(ASC) 정렬을 합니다. 여기서 오름차순이란 숫자가 1, 2, 3, 4로 증가하는 순서로 정렬하는 방식을 의미합니다. 내림차순은 이와 반대로 숫자가 줄어들어 가는 순서로 정렬하는 방식을 말합니다.

즉 ORDER BY id ASC는 id 값을 기준으로 오름차순으로 정렬하라는 말이 됩니다. 하지만 여기서 표기하지 않은 이유는 오름차순 정렬이 기본적인 정렬이기 때문에 표기하지 않아도 자동으로 정렬되기 때문입니다. 따라서 1, 3, 6, 4, 2, 5, 7, 9번의 글들이 있고 현재 글이 5번이라고 했을 때 위의 쿼리를 실행하면 6,7,9와 같이 정렬되고 그 중 제일 첫 번째로 검색된 6만이 선택된 결과로 나오게 됩니다.

그렇다면 다음 글은 어떻게 될까요?

```
$query=mysql_query("SELECT id FROM board WHERE id < $id ORDER BY id
DESC LIMIT 1", $conn);
```

위와 같이 이전 글과 정반대로 구현하면 됩니다. 1, 3, 6, 4, 2, 5, 7, 9번의 글 중에 5번보다 작은 글들은 1, 2, 3, 4 이렇게 4개의 글입니다. ORDER BY id DESC 구문이 있으니 id 값을 기준으로 내림차순으로 정렬해야 합니다. 내림차순으로 저 숫자들을 정리해보면 4, 3, 2, 1과 같이 검색될 것입니다. 그런데 LIMIT 1이 있으니 그중에 첫 번째 검색된 4가 결과로 나오게 됩니다.

위와 같이 쿼리를 이용하면 만약 글이 삭제되었다 하더라도 쉽게 다음 글과 이전 글을 찾아낼 수 있을 것입니다.

마지막으로 글을 읽었으니 조회 수를 증가시키도록 합니다.

```
// 조회 수 업데이트
$result=mysql_query("UPDATE board SET view=view+1 WHERE id=$id",
$conn);
```

혹시 데이터베이스에서 현재 글 정보로부터 읽어온 조회 수에 1을 더하여 UPDATE 하는 것을 생각했다면 위의 방법을 추천해 드립니다. 현재 글 정보로부터 읽어온 조회 수에 1을 더하는 방식은 문제가 생길 소지가 있습니다.

> 영희와 철수가 같은 글을 읽으려고 합니다. 그런데 영희는 오늘따라 인터넷 속도가 굉장히 느리고 철수는 오늘따라 인터넷이 빛의 속도보다 빠른 듯합니다. 이때 영희가 먼저 글을 읽어서 데이터베이스에서 글 정보를 읽어왔는데 인터넷이 느려서 HTML이 다 출력되지 않았습니다. 잠시 후에 철수가 같은 글을 읽게 되었고 철수는 빠른 인터넷 때문에 금방 글이 출력되어 조회 수를 0에서 1로 바꾸었습니다. 하지만 영희는 아직 모래시계만 돌아가고 있습니다. 3초 후 철이가 글을 한창 읽고 있을 때쯤 영희의 모니터에는 글이 모두 출력되었습니다. 이때 영희는 철수가 조회 수를 업데이트 하기 이전에 조회 수를 읽어들였기 때문에 결과적으로 0에 1을 더하여 조회 수를 1로 변경합니다. 실제로는 두 명이 글을 읽었지만 결과적으로 조회 수는 1이 됩니다.

이 예는 얼핏 굉장히 비약이 심해 보입니다. 또한 조회 수 1 차이가 나는 것이 무어 대수냐고 생각할 수 있습니다. 하지만 글이 아니라 통장에 있는 돈을 인출하는 것이었다면 어떨까요?

> 영희와 철수가 같은 통장에서 돈을 인출하려 합니다. 그런데 영희가 사용하는 현금 인출기가 오늘따라 유난히 느리고 철수가 사용하는 현금인출기는 오늘따라 빛의 속도보다 빠른 듯합니다. 이때 영희가 먼저 현금카드를 넣어서 데이터베이스에서 통장 잔고를 읽어왔는데 인출기가 느려서 출금이 지연되었습니다. 잠시 후에 철수가 통장을 이용해 현금 인출을 시도했고 빠른 처리로 인해 금방 돈을 인출받을 수 있었습니다. 이때 인출기는 총 잔고 10만 원에서 5만 원을 인출하여 잔액이 5만 원이 있음을 데이터베이스에 기록합니다. 하지만 영희는 아직까지 처리가 되지 않고 철이가 5만 원을 인출하고 나서야 돈을 인출할 수 있었습니다. 이때 영희가 사용하는 인출기는 철수가 잔고를 업데이트 하기 이전에 잔액을 조회하였기 때문에 결과적으로 10만 원에서 5만 원을 인출하여 잔액이 5만 원이라고 데이터베이스에 저장합니다. 실제로는 두 명이 각각 5만 원씩 10만 원 모두를 출금하였지만 결과적으로 잔액은 5만 원이 됩니다.

앞의 예를 은행 버전으로 바꾸어 보았는데 조금 중요한 듯이 보이나요? 10만 원 인출했는데 5만 원 남아서 좋겠다고요? 하지만 은행 입장에서는 5만 원을 손해 보게 된 경우입니다. 이런 식으로 현금 인출기를 만들었다면 은행은 망해버릴 것입니다. 그래서 은행이나 금융기관과 같은 돈을 취급하는 곳에서는 이런 부분에 매우 민감하여 데이터베이스 쿼리를 매우 신중히 작성합니다.

마지막으로 데이터베이스의 연결을 종료하면 글 읽기의 구현이 모두 완료됩니다. 글 읽기의 완성된 소스 코드는 다음과 같습니다.

read.php

```
<html>
<head>
<title>초 허접 게시판</title>
<style>
<!--
td {font-size : 9pt;}
A:link {font : 9pt; color : black; text-decoration : none;
font-family : 굴림; font-size : 9pt;}
A:visited {text-decoration : none; color : black; font-size : 9pt;}
A:hover {text-decoration : underline; color : black;
font-size : 9pt;}
-->
</style>
</head>

<body topmargin=0 leftmargin=0 text=#464646>
<center>
<BR>
<?
  //데이터 베이스 연결하기
  include "db_info.php";

  // 글 정보 가져오기
  $result=mysql_query("SELECT * FROM board WHERE id=$id", $conn);
  $row=mysql_fetch_array($result);
?>
<table width=580 border=0 cellpadding=2 cellspacing=1
bgcolor=#777777>
<tr>
  <td height=20 colspan=4 align=center bgcolor=#999999>
    <font color=white><B><?=$row[title]?></B></font>
  </td>
</tr>
<tr>
  <td width=50 height=20 align=center bgcolor=#EEEEEE>글쓴이</td>
```

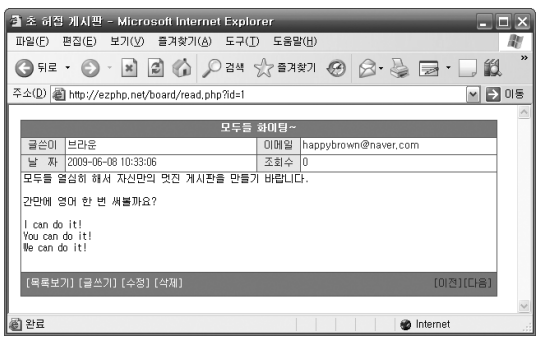
```
36    <td width=240 bgcolor=white><?=$row[name]?></td>
37    <td width=50 height=20 align=center bgcolor=#EEEEEE>이메일</td>
38    <td width=240 bgcolor=white><?=$row[email]?></td>
39  </tr>
40  <tr>
41    <td width=50 height=20 align=center bgcolor=#EEEEEE>
42    날   짜</td>
43    <td width=240 bgcolor=white><?=$row[wdate]?></td>
44    <td width=50 height=20 align=center bgcolor=#EEEEEE>조회수</td>
45    <td width=240 bgcolor=white><?=$row[view]?></td>
46  </tr>
47  <tr>
48    <td bgcolor=white colspan=4>
49    <font color=black>
50    <pre><?=$row[content]?></pre>
51    </font>
52    </td>
53  </tr>
54  <!-- 기타 버튼 들 -->
55  <tr>
56    <td colspan=4 bgcolor=#999999>
57    <table width=100%>
58      <tr>
59        <td width=200 align=left height=20>
60          <a href=list.php?no=<?=$no?>><font color=white>
61          [목록보기]</font></a>
62          <a href=write.php><font color=white>
63          [글쓰기]</font></a>
64          <a href=edit.php?id=<?=$id?>><font color=white>
65          [수정]</font></a>
66          <a href=predel.php?id=<?=$id?>>
67          <font color=white>[삭제]</font></a>
68        </td>
69        <!-- 기타 버튼 끝 -->
70        <!-- 이전 다음 표시 -->
71        <td align=right>
72  <?
73    // 현재 글보다 id 값이 큰 글 중 가장 작은 것을 가져온다.
74    // 즉 바로 이전 글
75    $query=mysql_query("SELECT id FROM board WHERE id > $id LIMIT 1",
76    $conn);
77    $prev_id=mysql_fetch_array($query);
78
79    if ($prev_id[id]) // 이전 글이 있을 경우
80    {
81      echo "<a href=read.php?id=$prev_id[id]>
82      <font color=white>[이전]</font></a>";
```

```
83    }
84    else
85    {
86      echo "[이전]";
87    }
88
89    $query=mysql_query("SELECT id FROM board WHERE id < $id
90    ORDER BY id DESC LIMIT 1", $conn);
91    $next_id=mysql_fetch_array($query);
92
93    if ($next_id[id])
94    {
95      echo "<a href=read.php?id=$next_id[id]>
96      <font color=white>[다음]</font></a>";
97    }
98    else
99    {
100     echo "[다음]";
101   }
102 ?>
103       </td>
104     </tr>
105   </table>
106   </b></font>
107   </td>
108 </tr>
109 </table>
110 </center>
111 </body>
112 </html>
113 <?
114   // 조회 수 업데이트
115   $result=mysql_query("UPDATE testboard SET see=see+1 WHERE id=$id",
116   $conn);
117
118   mysql_close($conn);
119 ?>
```

글 읽기가 완성되었으니 글 수정하기에서 수정한 글을 read.php 페이지를 통해 읽어보도록 하겠습니다. 글을 읽을 때도 마찬가지로 웹 브라우저의 주소창에 다음과 같이 글 번호를 명시해야 합니다.

```
http://도메인/read.php?id=1
```

그 결과는 다음과 같습니다.



[그림 11-17] 글 읽기 구현 결과

이제부터는 값이 제대로 입력되어 있는지 확인하기 위해 굳이 MySQL 콘솔을 사용하지 않아도 되겠습니다.

Section 09 글 삭제 – predel.php

글을 삭제한다고 하면 제일 먼저 글쓴이인지 아닌지 판단할 필요가 있습니다. 만약 글 삭제 버튼을 누름과 동시에 바로 삭제된다면 어느 누구나 다른 사람의 글을 지울 수 있다는 이야기가 되어 버립니다. 그래서 글쓴이인지를 판단하고 나서 글쓴이라고 밝혀지면 삭제하는 것이 맞을 것입니다. 물론 그전에 로그인과 같은 절차를 통해서 글쓴이라고 인증이 되었다면 이런 절차가 필요 없을 것입니다.

그렇다면 글쓴이인지 아닌지는 어떻게 판단할 수 있을까요? 지문인식시스템을 통해서? 망막인증? 손등의 정맥검사를 통해서?

우리는 앞서 글을 쓸 때 비밀번호를 입력하게 했습니다. 바로 이 비밀번호가 있으니 이것을 통해서 글쓴이인지 판단할 수 있습니다.

여기서 잠깐

비밀번호를 묻는 페이지를 별도로 만드는 것을 피하고자 레이어를 통해서 처리하기도 합니다. 레이어에 비밀번호 입력 폼을 만들고 곧바로 del.php 파일로 비밀번호를 전달하는 것이지요. 꼭 한 번 만들어 보기 바랍니다.

비밀번호를 이용하면 된다는 것은 알았는데 그러면 어떻게 비밀번호를 입력받아야 할까요? 비밀번호를 입력받는다는 것은 폼(form)을 이용한다는 이야기인데, 글 삭제 버튼 옆에다가 덩그러니 입력창을 달아놓을 수도 없고 말입니다. 그래서 결국 어쩔 수 없이 비밀번호를 입력받는 페이지를 하나 만들었습니다.

predel.php 파일은 앞서 비밀번호를 입력받고자 만든 페이지이므로 PHP 소스 코드는 거의 필요 없습니다. 그럼 웹 에디터나 텍스트 에디터를 띄워서 아래와 같은 비밀번호 입력 폼을 만들어 봅시다.

[그림 11-18] 비밀번호 입력 폼

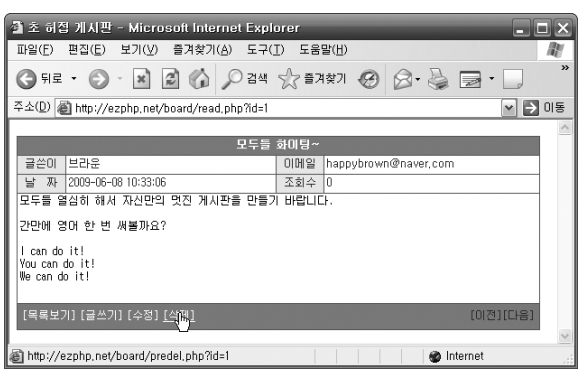
비밀번호를 입력받을 입력창에는 이름을 pass라고 지정하고 타입을 패스워드로 지정합니다.

```
<INPUT type=password name=pass size=8>
```

그리고 폼의 action을 del.php 파일로 설정합니다.

```
<form action=del.php method=post>
```

여기서 한 번 테스트를 해볼까요?



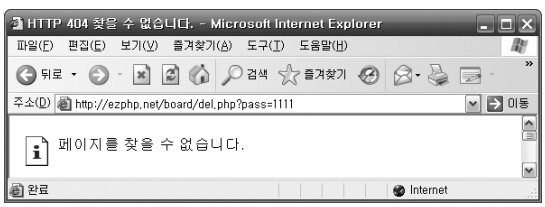
[그림 11-19] 삭제 링크 정보

글을 삭제하기 위해 삭제 버튼을 클릭하는 모습입니다. 삭제 버튼의 링크가 predel.php로 향하는 것을 알 수 있고 id 값을 전해주고 있음을 알 수 있습니다. 왜일까요? 네. 이제 하산하셔도 될 것 같

습니다. 해당하는 글만 삭제하기 위해 id 값(글 번호)을 넘겨주는 것입니다.

[그림 11-20] 전달받은 삭제할 글 번호

삭제 버튼을 클릭하면 위처럼 predel.php 파일로 이동합니다. 앞서 넘겨준 글 번호가 따라다니는군요. 그럼 비밀번호(1111)를 입력하고 확인 버튼을 눌러봅시다. 비록 아직 del.php 파일을 만들지는 않았지만요.



[그림 11-21] GET 방식으로 전달된 pass 값 (테스트를 위해 GET으로 변경했음)

pass 값이 1111이라고 넘어왔습니다. 근데 뭔가 이상합니다. "페이지를 찾을 수 없습니다."라고 나온 것 말고요. 자세히 보니 id 값이 사라져 버렸네요. 글 번호가 없으면 글을 삭제하질 못하는 데 말이죠. 왜 사라졌을까? predel.php까지는 잘 넘어왔는데 del.php에서는 사라졌다? 그렇다면 predel.php에서 값을 못 넘겨 준 것이군요.

아하~! id 값을 넘겨주어야 하는데 비밀번호 넘겨주는 것만 신경 쓰다가 미쳐 생각을 못했습니다. 다음과 같이 action을 수정하도록 합니다.

```
<form action=del.php?id=<?=$id?> method=post>
```

그런데 새로운 방식이죠? POST 와 GET을 동시에 쓰다니. 가끔 유용하게 쓰이는 방법이니 꼭 기억해두기 바랍니다. 하지만 일반적으로는 다음과 같이 숨겨진 폼을 이용하여 전달하는 경우가 많습니다.

```
<INPUT type=hidden name=id value="<?=$id?>">
```

그리고 하나 더. 앞서 위의 테스트를 위해 POST를 GET으로 바꾸었다고 말씀드렸습니다. 왜 일부러 비밀번호를 POST로 전송했을까요? 네. 앞에서 잠깐 언급한 적이 있는데, GET 방식을 사용하여

비밀번호를 전송하면 위처럼 비밀번호가 URL을 통해 드러나게 됩니다. 여럿이 쓰는 컴퓨터라면 누군가 엿볼 수도 있을 것입니다. 따라서 비밀번호와 같이 숨기고 싶은 것을 전송할 때는 POST를 반드시 사용해야 합니다.

벌써 설명이 모두 끝나버렸군요. 나머지는 HTML 부분이니 충분히 이해하리라 믿습니다. 완성된 predel.php 파일의 소스 코드는 다음과 같습니다.

predel.php

```
<html>
<head>
<title>초 허접 게시판</title>
<style>
<!--
td {font-size : 9pt;}
A:link {font : 9pt;color : black;text-decoration : none;
font-family: 굴림;font-size : 9pt;}
A:visited {text-decoration : none; color : black; font-size : 9pt;}
A:hover {text-decoration : underline; color : black;
font-size : 9pt;}
-->
</style>
</head>
<body topmargin=0 leftmargin=0 text=#464646>
<center>
<BR>
<!-- 입력된 값을 다음 페이지로 넘기기 위해 FORM을 만든다. -->
<form action=del.php?id=<?=$id?> method=post>
<table width=300 border=0 cellpadding=2 cellspacing=1
bgcolor=#777777>
<tr>
  <td height=20 align=center bgcolor=#999999>
    <font color=white><B>비 밀 번 호 확 인</B></font>
  </td>
</tr>
<tr>
  <td align=center >
    <font color=white><B>비밀번호 : </b>
    <INPUT type=password name=pass size=8>
    <INPUT type=submit value="확 인">
    <INPUT type=button value="취 소" onclick="history.back(-1)">
  </td>
</tr>
</table>
```

Section
10 글 삭제 반영 – del.php

del.php 파일은 데이터베이스에서 글을 삭제하는 기능을 합니다. 글 수정에서도 배웠듯이 일단 먼저 글쓴이인지 인증을 해야겠네요. 데이터베이스에서 글 번호를 통해 글의 비밀번호를 알아냅니다. 이 부분은 update.php 페이지에서 이미 해본 부분이니 쉽게 넘어가 보겠습니다.

첫 번째, 데이터베이스에 연결합니다.

```
include "db_info.php";
```

두 번째, 해당 글의 비밀번호를 가져오는 쿼리를 작성하고 그 결과를 가져옵니다.

```
$result=mysql_query("SELECT pass FROM board WHERE id=$id", $conn);
$row=mysql_fetch_array($result);
```

세 번째, 비밀번호를 확인하고 맞으면 글을 삭제하고 틀리면 에러 메시지를 출력합니다.

```
if ($pass==$row[pass])
{
  // 삭제 코드
}
else
{
  // 에러 메시지 출력
}
```

네 번째, 글을 삭제하는 쿼리를 작성합니다. 앞서 UPDATE 할 때도 언급했지만 삭제 조건을 정확히 기입해야 실수를 줄일 수 있습니다. 해당 글을 지우려면 글 번호가 필요하겠죠? 글 번호를 조건에 추가하여 다음과 같은 쿼리를 완성합니다.

```
DELETE FROM board WHERE id=$id
```

이제 조금 쿼리 작성하는 것에도 탄력이 붙는 것 같습니다. 조금 쉬워진 것 같지 않나요?

```
if ($pass==$row[pass])
{
  $query = "DELETE FROM board WHERE id=$id ";
  $result=mysql_query($query, $conn);
}
else
{
  echo ("
    <script>
```

```
        alert('비밀번호가 틀립니다.');
        history.go(-1);
        </script>
        ");
        exit;
    }
```

마지막으로 글을 삭제했으니 되돌아갈 곳이 마땅치 않습니다. 글 읽기 페이지로 갈 수도 없는 노릇이고 그래서 글 목록으로 이동하도록 합니다.

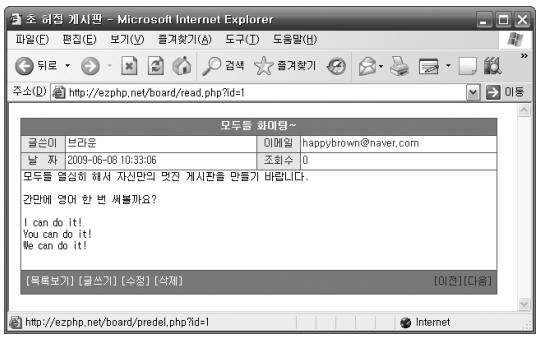
```
<meta http-equiv='Refresh' content='1; URL=list.php'>
```

앞에서 이미 다뤄본 내용이라 쉽게 넘어갈 수 있었습니다. 사실 del.php 페이지는 update.php 페이지를 가져다가 쿼리와 메시지 등 약간만 수정하면 그대로 사용할 수 있습니다. 완성된 del.php 페이지의 소스 코드는 다음과 같습니다.

del.php

```
1  <?
2  //데이터 베이스 연결하기
3  include "db_info.php";
4
5  $result=mysql_query("SELECT pass FROM board WHERE id=$id",
6  $conn);
7  $row=mysql_fetch_array($result);
8
9  if ($pass==$row[pass] )
10 {
11   $query = "DELETE FROM board WHERE id=$id ";
12   $result=mysql_query($query, $conn);
13 }
14 else
15 {
16   echo ("
17   <script>
18   alert('비밀번호가 틀립니다.');
19   history.go(-1);
20   </script>
21   ");
22   exit;
23 }
24 ?>
25 <center>
26 <meta http-equiv='Refresh' content='1; URL=list.php'>
27 <FONT size=2 >삭제되었습니다.</font>
```

삭제하기 기능을 모두 구현하였으니 실제로 글을 삭제해 보겠습니다.



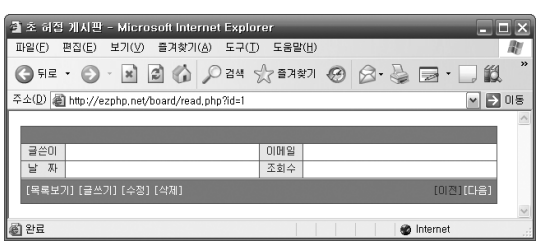
[그림 11-22] 글 읽기 페이지

글 읽기 페이지에서 삭제 버튼을 눌러봅시다.

[그림 11-23] 삭제를 위한 비밀번호 입력

비밀번호를 넣고 확인 버튼 클릭! 삭제가 완료되었습니다. 어디 확인해 볼까요? 다시 글 읽기 페이지를 불러봅니다.

```
http://도메인/read.php?id=1
```



[그림 11-24] 삭제된 글 정보

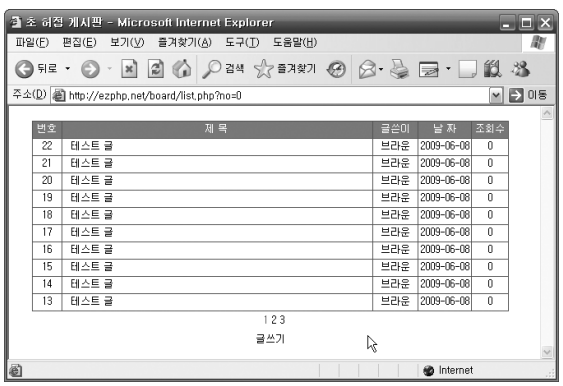
이런 모든 항목이 비워졌네요. 즉, 글이 깨끗이 지워졌다는 뜻입니다. 그렇다고 지워진 글의 번호가 살아있고 내용만 지워진 것이 아닙니다. 에러 메시지가 없을 뿐이지 글을 찾지 못해서 각 항목을 보여줄 수가 없어서 빈칸으로 출력된 것입니다. 잘 지워지는 것 같으니 게시판의 마지막 목록 페이지로 넘어가 봅시다.

Section 11

글 목록 – list.php

이제 게시판의 마지막 글 목록 페이지만 남겨두었습니다. 글 목록 페이지는 게시판의 핵심이라고 볼 수 있습니다. 지금까지와는 다르게 신중히 생각해야 하는 부분이 많으므로 차근차근 따라오기 바랍니다.

일단 목록 페이지가 어떤 일을 하는지 알아보겠습니다. 다음 그림은 일반적인 형태의 게시판 목록을 나타냅니다.



[그림 11-25] 글 목록 보기의 구성

글 목록 페이지는 위 그림을 통해서 알 수 있듯이 다음과 같은 세 가지 기능으로 구분할 수 있습니다.

① 글의 목록을 보여준다.

② 다음 페이지로 이동할 수 있는 페이지 리스트를 보여준다.

③ 글을 쓸 수 있는 버튼을 보여준다.

list.php 페이지는 글의 목록을 보여주는 역할을 하면서 또한 글을 볼 수 있게 해주는 다리이기도 합니다. 글의 목록 자체에 글의 내용을 다 보여주는 방명록과는 달리 글의 제목을 보여주면서 읽어 볼 글을 선택할 수 있게 한 것입니다. 대체로 방명록은 짧은 글들로 이루어져 있기 때문에 내용을 전부 보여준다고 하더라도 글 목록의 길이가 그렇게 길지 않습니다. 하지만 게시판은 방명록과 비교해 다소 긴 글들로 이루어지기 때문에 방명록과 같이 목록을 출력한다면 매우 긴 페이지를 보게 될 것입니다.

글의 목록에서는 위와 같이 방문자로 하여금 약간의 정보를 통해 글을 선택하여 읽을 수 있게끔 하는 것이 주요 목적입니다. 게시판의 성격에 따라 보여주는 항목에 차이는 있지만 일반적으로 다음

과 같은 항목을 출력합니다.

① 글의 번호
② 글의 제목
③ 글쓴이
④ 글 쓴 날짜
⑤ 조회 수

글의 번호의 용도는 주로 글의 순서를 인지시키고자 하는 용도와 "몇 번 글"이라는 게시물을 구분 짓는 용도 등으로 사용됩니다. 사실 글의 번호는 게시판 사용자보다는 게시판을 만들고 있는 프로그래머에게 유용한 항목이어서 간혹 글 번호를 보여주지 않는 게시판도 많이 있습니다. 글의 번호는 일반적으로 게시물이 등록된 순서이며 중간에 삭제된 글이 있다면 삭제된 글을 무시한 번호입니다. 예를 들면 게시판에 총 5개의 글이 올라왔는데 그 중 하나의 글이 삭제되었다면 마지막에 올라온 게시물의 번호는 5가 아니라 4인 것입니다. 즉, 우리가 글을 구분짓고자 사용하는 id 값과 글의 번호는 실질적으로 같지 않다는 것을 의미합니다. 게시판을 개발하는 프로그래머는 id 값을 글 번호라고 생각하고, 게시물을 보는 사용자는 실제 게시판에 존재하는 글 중에서 몇 번째 글인가를 글 번호라고 생각한다는 것으로 이해하면 됩니다. 이렇게 실제 id 값과 글의 번호가 다른 이유는 id 값을 글의 번호로 보여주는 경우 중간에 여러 개의 글이 삭제되면 글 번호가 2에서 갑자기 15가 되는 등 혼란스러워질 가능성이 있기 때문입니다.

다음으로 글의 제목을 보여주는 이유는 독자 분들도 쉽게 생각할 수 있듯이 글의 제목이 글의 전체 내용을 반영하기 때문에 많은 글 중에서 해당 게시물을 선택하는 기준이 되기 때문입니다. 글의 제목과 마찬가지로 글쓴이를 보여주는 이유도 게시물의 선택하는 기준이 되기 때문입니다. 만약 유명한 사람이나 자기와 친분이 있는 사람의 글이 등록된다면 더 관심 있게 글을 읽지 않겠습니까? 대표적으로 PC 통신 시절 나우누리에서 활동하시던 견우74님을 들 수 있는데, 그분이 쓰신 글은 조회수가 다른 글들과 비교할 수 없게 엄청났고 저를 포함하여 그 분이 쓰신 새 글이 올라오기만을 손꼽아 기다리는 분들이 매우 많았습니다. 그분이 어떤 분이 길래 그렇게 유명하냐고요? 견우74님은 바로 그 유명한 영화 "엽기적인 그녀"의 원작자이십니다.

글 쓴 날짜를 보여주는 이유는 최근 글임을 알려서 사용자로 하여금 읽게 하려는 의도이며 마지막으로 조회 수를 보여주는 이유는 조회 수가 높을수록 글의 내용이 많은 사람이 읽을만한 글이라고 생각하는 성향 때문입니다.

그렇다면 위와 같은 항목을 목록으로 보여주는 쿼리를 한 번 만들어 봅시다. 일단 목적은 글의 제목과 같은 몇 개의 항목을 가져오는 것입니다.

```
SELECT * FROM board ORDER BY id DESC
```

쉽게 위와 같은 쿼리를 통해 글 목록을 가져올 수 있습니다. 위의 쿼리를 번역하면 “board 테이블에서 id 값을 기준으로 내림차순으로 정렬한 후 모든 글에 대한 항목을 가져오라”는 의미입니다. 앞서 사용할 때는 대부분 WHERE 절이 있어서 검색 조건이 붙었는데 이번 쿼리에서는 검색 조건이 사라졌습니다. 모든 글이 다 검색된다는 것입니다.

그런데 위와 같은 쿼리를 사용하면 목록을 출력할 때 심각한 문제가 생깁니다. 처음에 글이 5개, 10개 이렇게 적은 수의 글이 있을 때는 문제가 없었지만 글이 수십 개 혹은 수 백 개로 늘어나면 글의 목록이 늘어나서 보기가 싫어질 뿐만이 아니라 목록 출력이 다소 느려지게 됩니다. 100개의 목록이라 스크롤의 압박도 느껴집니다.

100개의 글, 1000개의 글을 한 페이지에다가 모두 출력한다는 것은 너무나 어리석은 일입니다. 느려지는 것을 떠나서 글을 보는 사람들도 짜증이 날 것입니다. 이러한 문제를 해결하기 위해서 대부분의 게시판이 페이지를 나눠 한 페이지에 몇 개의 글만을 보여줍니다. 간혹 ActiveX로 구현된 게시판은 스크롤을 내릴 때마다 특정 개수의 글을 가져오게 하는 경우도 있습니다.

그렇다면 우리는 한 페이지당 10개씩 글의 목록을 보여주게 만들어 봅시다. 그러면 일단 글 10개만 가져오는 쿼리를 만들어야겠네요.

```
SELECT * FROM board ORDER BY id DESC LIMIT 0,10;
```

다행히 MySQL에는 LIMIT라는 것이 있었습니다. 앞에서 이전 글, 다음 글을 표시할 때도 사용했었습니다. 그때의 LIMIT 1은 하나의 글만 가져오는 것이라고 배웠습니다. 그렇다면 LIMIT 5는 5개의 글만 가져오는 것일 겁니다. 근데 이번에는 LIMIT 0, 10과 같이 숫자가 두 개가 붙어 있습니다.

이번에 사용된 LIMIT는 위와 같이 시작번호와 가져올 개수를 사용합니다. 예를 들어 LIMIT 0, 10이라는 것은 "0번 글부터 10개를 가져오라"는 뜻이 됩니다. LIMIT 100,10은 "100번 글부터 10개를 가져오라"는 뜻이 되겠지요. 그런데 100번째 글이란 건 어떤 것을 기준으로 100번째란 말일까요?

그것은 검색 조건이나 정렬에 따른 순서입니다. 여기에는 ORDER BY id DESC라고 명시되어 있으니 id가 가장 큰 수가 첫 번째입니다. 그래서 위의 쿼리를 이용하면 가장 최근에 올라온 글 10개가 검색됩니다.

그런데 지금까지는 검색 결과가 늘 하나씩이었습니다. 여러 개가 검색되는 것은 처음인데 어떻게 해야 할까요? 그럴 때는 다음과 같이 하면 됩니다.

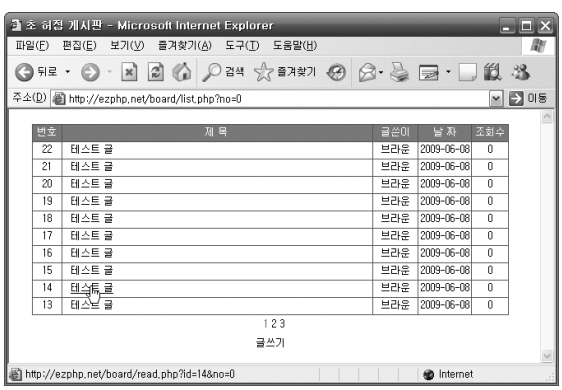
```
while($row=mysql_fetch_array($result)) {
    echo $row[id];
    echo $row[name];
    echo $row[title];
    echo substr($row[wdate], 0, 10);
    …
}
```

위와 같이 하면 검색된 모든 글을 출력할 수 있습니다. mysql_query 함수를 통해서 검색한 결과를 mysql_fetch_array 함수를 통해 한 행씩 가져옵니다. 가져온 결과값은 $row 변수에 배열의 형태로 저장되고 이를 id, name과 같은 필드 이름을 이용해서 원하는 값을 선택할 수 있습니다. 날짜의 경우 변수 $row[wdate]에는 시간 정보가 포함되어 있어서 날짜 부분만 잘라낼 필요가 있습니다. 따라서 substr() 함수를 이용하여 앞 10자를 잘라내어 출력합니다. 이렇게 원하는 곳에 적절하게 출력을 하고 나면 다음 결과를 가져와야 할 텐데요. 루프를 통해 다시금 mysql_fetch_array 함수가 실행되고 그 결과가 다시 $row 변수에 저장되기 때문에 검색된 여러 개의 결과를 계속해서 출력할 수 있습니다. 모든 결과를 처리하고 다시 mysql_fetch_array 함수가 불리면 더 이상 결과가 없으므로 FALSE를 되돌려주고 $row 변수값이 FALSE가 되면서 루프가 종료됩니다.

이 방법은 매우 정형화된 방법으로 이와 같은 작업을 할 때는 대부분 위와 같은 코드가 사용됩니다. 따라서 이 방법을 꼭 기억해 두었다가 여러 검색 결과에 대한 결과값을 출력하거나 처리해야 할 때 유용하게 사용하기 바랍니다.

이제 목록은 뿌려줬지만 제목만 보면 뭐하겠습니까? 앞서 말씀드렸지만 글 목록 페이지는 글을 선택해서 읽게 해주는 기능을 하기 때문에 목록 중의 하나를 선택해서 해당 글의 내용을 볼 수 있는 글 읽기 페이지로 이동시켜야 할 의무(?)가 있습니다.

그렇다면 어떤 식으로 링크를 걸어줘야 할까요?



[그림 11-26] 글 목록에서 제목의 링크 정보

read.php에서 글을 읽을 때 글을 구분하는 기준은 글의 번호(id)였습니다. 따라서 글을 읽으려면 반드시 글 번호를 넘겨야 합니다. 이와 더불어 페이지 정보 또한 넘겨주게 되는데 그 이유는 글을 읽다가 다시 목록으로 돌아오고자 할 때 그 글이 있던 목록 페이지로 이동하기 위해서입니다. 만약 페이지 정보를 넘겨주지 않는다면 글 읽기 페이지에서 목록 보기를 클릭했을 때 항상 제일 첫 번째 페이지로 돌아가 버릴 것입니다. 네 번째 페이지의 목록에 있는 글을 보고 있었는데 첫 번째 페이지로 이동해 버린다면 사용자가 무척 짜증내지 않겠습니까?

따라서 보기 좋게 테이블로 정렬하고 제목에 링크를 달아보면 다음과 같습니다.

```
<table>
<tr>
<td>번호</td><td>제목</td><td>글쓴이</td><td>날짜</td><td>조회 수</td>
</tr>
<?
  while($row=mysql_fetch_array($result))
  {
?>
<tr>
<td>
  <a href=read.php?id=<?=$row[id]?>&no=<?=$no?>><?=$row[id]?></a>
</td>
<td>
<a href=read.php?id=<?=$row[id]?>&no=<?=$no?>><?=$row[title]?></a>
</td>
<td>
  <a href="mailto:<?=$row[email]?>"><?=$row[name]?></a>
</td>
<td ><?=substr($row[wdate], 0, 10)?></td>
<td ><?=$row[view]?></td>
</tr>
<?
  } // end While
?>
```

이제 검색된 결과 10개 혹은 글이 10개보다 적다면 그 개수만큼 목록을 출력할 수 있게 되었습니다. 여기까지는 그다지 어렵게 느끼지 않으셨으리라 생각됩니다. 하지만 이제 머리를 조금 써야 하는 부분에 도달했습니다. 사실 이 게시판 제작에서 가장 어려운 부분이라고 생각됩니다. 지금까지 열심히 달려오신 분은 잠깐 바람이라도 쐬었다가 오시기 바랍니다. 왜냐하면 지금부터 게시판의 꽃, 페이지 나누기를 할 예정이기 때문입니다. 먼저 이 글을 보시기 전에 바람을 쐬면서 어떻게 페이지 리스트를 구현할지 한 번 나름대로 생각을 해보기 바랍니다.

[그림 11-27] 페이지 나누기

우리는 위와 같은 것을 만들려고 합니다. 1~10, 11~20, 21~30과 같은 페이지 리스트를 뿌려주고

자 하는 것입니다.

"그냥 for 문으로 1부터 10 이런 식으로 출력하면 되지 않나요? 쉬워 보이는데."

사실 기본적으로는 for 문을 이용하여 10개 단위씩 출력을 하면 됩니다. 어떻게 보면 간단해 보이기도 하는데요. 실제로 고려해야 할 사항이 많이 존재합니다.

① 현재 페이지는 몇 번째 페이지인가?
② 현재 페이지는 몇 번 페이지부터 시작하는가?
③ 현재 페이지는 몇 번 페이지까지 출력하는가?
④ 10개를 출력하였다면 그 이후의 페이지가 더 존재하는가?
⑤ 첫 번째 출력된 페이지 번호보다 작은 페이지가 존재하는가?
⑥ 현재 페이지의 첫 번째 글은 몇 번의 글인가?
⑦ 이외 여러 가지 사항

위와 같은 사항을 고려해야만 페이지 목록을 구현할 수 있습니다. 페이지 목록은 단순히 목록을 출력해주는 것으로 끝나는 것이 아니라 쿼리와도 연계가 되기 때문에 차근차근 이해해 나가야 합니다.

그러려면 일단 페이지에 대한 정의를 해야 합니다.

변수명	역 할
$no	페이지가 시작되는 첫 글의 순번
$page_size	한 페이지당 보여줄 게시물의 수
$page_list_size	페이지 리스트에 보여줄 페이지의 수
$total_row	총 게시물의 수
$total_page	총 페이지 수
$current_page	현재 페이지 (0부터 시작)
$start_page	페이지 리스트에서 첫 번째 페이지 (0부터 시작)
$end_page	페이지 리스트에서 마지막 페이지 (0부터 시작)
$prev_list	이전 페이지 리스트
$next_list	다음 페이지 리스트

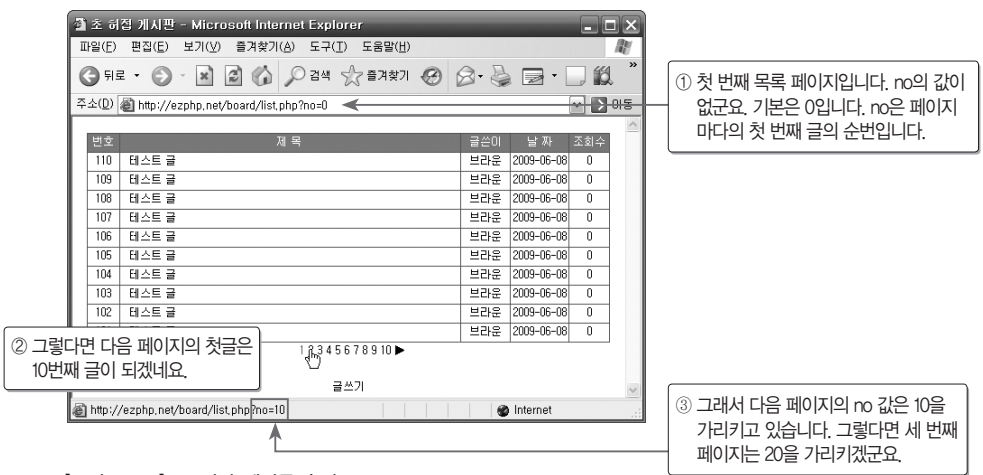
[표 11-11] 페이지 관련 변수 정보

의외로 페이지 목록을 구현하는데 많은 변수가 사용되고 있음을 알 수 있습니다. 그만큼 페이지 목록이 어렵지만 게시판의 꽃으로 불리는 이유입니다. 페이지 나누기만 끝내면 웬만한 웹 프로그램은 다 만들 수 있습니다. 그러니 기죽지 말고 힘을 내서 하나씩 차근차근 배워봅시다. 우선 $no 변수부터 시작해 봅시다.

앞서 $no 변수는 현재 페이지를 나타낸다고 했습니다. 그렇다면 첫 번째 페이지는 1번, 두 번째 페이지는 2번 이런 식일까요? 안타깝지만 아닙니다. 물론 이처럼 되도록 구현할 수 있습니다. 그러나 편의를 위하여 $no 변수는 0, 10, 20과 같이 10씩 증가하는 값을 갖습니다.

왜 10씩 증가하는 값일까요? 이유는 실제로 $no 변수가 페이지 번호를 가리키는 것이 아니라 그 페이지의 첫 번째 글의 순서를 가리키기 때문입니다. 여기서 명심해야 할 것은 글의 번호(id)가 아니라 검색된 글의 순서입니다. 즉, $no 변수가 0이라면 앞서 만든 쿼리에서 검색된 결과의 첫 번째 행이 이 페이지의 첫 번째 글이 된다는 의미입니다. 만약 $no 변수값이 10이라면 검색된 결과 중 11번째(0부터 시작했으므로)에 존재하는 글이 페이지의 첫 번째 글이 됨을 의미합니다.

이해가 잘 안 되신다면 다음 그림이 도움될지도 모르겠습니다.



[그림 11-28] no 값과 게시물 순서

위의 목록을 보시면 110개의 글이 존재함을 알 수 있습니다. 여기서 번호는 글의 번호가 아니라 앞서 말씀드렸듯이 총 게시물에 대한 등록된 순서입니다. 즉, 가장 최근에 게시된 글이 가장 큰 값을 갖습니다. 여기서는 110번 글이 가장 최근에 등록된 글입니다.

이 페이지의 $no 값은 특별히 지정되어 있지 않은데 값이 지정되지 않았으면 첫 번째 페이지 즉, $no 값이 0임을 의미합니다. 왜냐하면 일반적으로 게시판의 주소를 입력할 때 대부분 http://도메인/board/list.php라고 입력하기 때문입니다. 여기에 http://도메인/board/list.php?no=0 이라고 입력하게 하는 귀찮음을 덜어주기 위해 다음과 같은 코드를 사용합니다.

```
if (!$no || $no < 0) $no=0; //$no 값이 안 넘어오거나 음수값이 넘어오는 경우 0
```

앞서 작성한 쿼리에서 알 수 있듯이 id 값이 큰 것이 먼저 검색되기 때문에 번호가 높은 글이 첫 번째 페이지에 위치하게 됩니다. 왜 no란 값이 생겼는지는 앞서 만든 목록을 뿌려주는 쿼리와 연관이 있습니다. 우리는 총 100개의 글이 있으면 그 중 10개만 뿌려주고 싶었고 그러고자 LIMIT 0,10을

사용했습니다. 이렇게 쿼리를 만든다면 두 번째 페이지도 세 번째 페이지도 0번부터 9번까지만 보이는 사태가 일어납니다. 분명히 의도는 두 번째 페이지는 10번부터 보여야 하는 데 말입니다. 결국 LIMIT 0, 10에서 0의 값을 매 페이지마다 다르게 설정을 해주어야 한다는 결론이 납니다. 첫 번째 페이지는 0번이, 두 번째 페이지는 10번이, 세 번째 페이지는 20번이 첫 글이 될 것입니다. 매 페이지마다 이렇게 목록의 첫 번째 글의 순번을 알고 있다면 페이지 나누기가 쉬워질 것입니다. 그래서 $no 값이 생겼습니다. 이해되시나요?

그럼 LIMIT 0, 10은 LIMIT $no, 10으로 바뀌면 페이지별로 10개씩의 글을 검색해 올 수 있게 됩니다.

```
SELECT * FROM board ORDER BY id DESC LIMIT $no, 10
```

두 번째로 $page_size 변수에 대해 알아보겠습니다. 이 변수의 경우는 페이지 나누기에 꼭 필요하지는 않으나 한 페이지에 10개의 글을 보여주고 싶다가 20개로 바꾸고 싶어하는 경우와 같은 때에 사용하기 위해서 만들어진 변수입니다. 한 페이지 내의 글 목록 수가 변경되면 쿼리가 변경되어야 하고 $no 변수도 변경된 수만큼 증가하게 변경해야 합니다. 물론 일괄적으로 수정하면 가능하겠지만 보다 손쉽게 이를 수정하기 위해 만들어진 변수입니다. 따라서 쿼리는 다음과 같이 수정됩니다.

```
SELECT * FROM board ORDER BY id DESC LIMIT $no, $page_size
```

세 번째로 총 게시물을 나타내는 $total_row 변수에 대해 알아보겠습니다. 페이지 목록을 나누려면 우선 총 게시물의 숫자를 알아야 합니다. 그래야만 한 페이지당 표시할 글의 수($page_size)를 통해서 총 몇 개의 페이지가 존재하는지 계산할 수 있기 때문입니다. 게시물이 총 몇 개인지는 COUNT 내장 함수를 사용하면 쉽게 알아낼 수 있습니다.

```
SELECT count(*) FROM board
```

MySQL 내장 함수인 COUNT 함수를 이용하여 board 테이블에 존재하는 글이 총 몇 개인지 알아내면 다음과 같이 $total_row 변수에 저장하도록 합니다.

```
$result_count=mysql_query("SELECT count(*) FROM board",$conn);
$result_row=mysql_fetch_row($result_count);
$total_row = $result_row[0];
```

여기서 기존의 것과는 조금 다른 것을 알 수 있습니다. 거의 매번 쿼리를 사용할 때는 $result와 $row 변수를 사용했었는데 여기서는 이들을 사용하지 않고 $result_count와 $count_row를 사용하고 있습니다. 이렇게 새로운 변수를 사용하여 쿼리의 결과를 가져오는 이유는 기존의 목록을 가져오기 위한 쿼리와 충돌을 방지하기 위해서입니다. 만약 다른 쿼리 둘을 하나의 변수를 이용하여 작업하는 경우 먼저 실행한 작업에 대한 결과가 덮어써져서 첫 번째 결과를 사용할 수 없게 됩니다.

또한 여기서는 mysql_fetch_array 함수 대신에 mysql_fetch_row 함수를 사용하고 있습니다. 기

존에 하던 방식으로 한다면 mysql_fetch_array 함수를 사용하여 $count_row 변수에 배열로 저장하고 값을 가져오고자 할 때는 다음과 같은 방식을 썼을 것입니다.

```
$count_row[count(*)]
```

그러나 이와 같은 방법은 동작하지 않습니다. 왜냐하면 count(*)는 컬럼의 이름이 아니기 때문입니다. 따라서 mysql_fetch_row 함수를 사용하여 첫 번째 컬럼값을 가져오도록 한 것입니다. 만약 여기서 기존의 방식을 계속해서 사용하고 싶다면 다음과 같이 쿼리를 수정해야 합니다.

```
SELECT COUNT(*) as total FROM board
```

위와 같이 별명을 이용하여 $count_row[total]과 같은 방법으로 값을 가져올 수 있습니다. 이 값을 이용하여 몇몇 게시판은 게시판의 상단에 총 게시물이 몇 개가 존재하는지 보여주기도 합니다.

총 게시물의 수를 구하였으니 이를 이용하여 총 페이지의 수를 계산해 보도록 합시다. 총 페이지의 수를 나타내는 변수는 $total_page입니다.

총 게시물의 수로 총 페이지의 수를 어떻게 계산할 수 있을까요? 연습장을 하나 꺼내어 계산을 한 번 해봅시다. 만약 총 게시물의 수가 45라고 해봅시다. 한 페이지에 10개씩 글을 출력한다면 10+10+10+10+5=45이기 때문에 총 5페이지가 될 것입니다. 여기서 마지막 페이지는 10개의 글이 아닌 5개의 글이 출력됩니다. 이를 수식으로 만들어보면 다음과 같습니다.

```
//총 게시물 수/  한 페이지에 출력되는 글의 수의 결과를 올림한다.
ceil($total_row/$page_size)
```

앞에서 언급한 45로 확인해보면 45/10=4.5가 되고 올림하면 5가 됩니다. 올림하는 이유는 게시물이 1개만 더 있더라도 한 페이지가 추가되기 때문입니다. 즉, 10개면 1페이지지만 11개인 경우는 1.1페이지가 아니라 2페이지가 됩니다. 이를 소스 코드로 표현하면 다음과 같습니다.

```
if ($total_row <= 0) $total_row = 0; //총 게시물의 값이 0이나 비정상이라면 0으로 설정
$total_page = ceil($total_row / $page_size);
```

총 페이지 수를 구하였으니 이제 현재 페이지의 번호인 $current_page를 구해봅시다. 현재 페이지는 $no 변수와 $page_size를 통해서 구할 수 있습니다. $no 변수를 통해서 현재 페이지의 첫 번째 글이 검색 결과 중 몇 번째 글인지 알 수 있기 때문에 총 페이지로 나누어서 몇 번째 페이지인지 구할 수 있습니다.

총 페이지의 수와 마찬가지로 현재 페이지의 번호도 구할 수 있습니다.

```
$current_page = ceil(($no+1)/$page_size);
```

여기서 $no 변수값에 1을 더해주는 이유는 $no 변수가 0부터 시작하기 때문입니다. 앞에서 총 페이지의 수를 구하고자 총 게시물의 수를 나누는 방법을 사용했습니다. 다르게 생각하면 총 45개의

글이 존재한다면 45번째 글이 몇 번째 페이지에 속하느냐라는 문제로 생각할 수 있습니다. 따라서 위에서 총 페이지 수를 구하는 공식을 그대로 사용할 수 있다는 뜻입니다. 그런데 문제가 있으니 바로 $no 변수입니다. $total_row 값은 1부터 시작하는 값이지만 $no 변수는 0부터 시작하는 값이기 때문입니다. 따라서 같은 공식을 사용하기 위해서는 $no 변수가 1부터 시작하게 할 필요가 있습니다. 그래서 위처럼 $no 변수에 1을 더하게 된 것입니다.

현재 페이지는 매우 중요한 역할을 합니다. 페이지 리스트를 뿌려줄 때 요긴하게 쓰이기 때문입니다. 페이지 리스트의 첫 번째 페이지와 마지막 페이지가 어떻게 될 것인지 결정하는 녀석이 바로 이 녀석입니다. 만약 현재 페이지가 13페이지라면 페이지 리스트는 11 ~ 20까지 출력해야 할 테니까요.

현재 페이지 번호를 이용하여 페이지 목록의 첫 번째 페이지 번호인 $start_page 값을 구해보도록 합시다. 그런데 여기서 페이지의 목록의 수를 조절할 수 있게 하고자 $page_list_size라는 변수를 추가합니다. 기본적으로는 10이란 값을 갖도록 하는 $page_list_size 변수를 조절함으로써 목록에 표시되는 수를 조절할 수 있게 말입니다. 이 $page_list_size 변수를 이용하여 수식을 옮겨보면 다음과 같습니다.

```
$start_page = floor(($current_page-1) / $page_list_size)
* $page_list_size + 1;
```

뭔가 계산이 많이 복잡해졌습니다. 수식이 복잡해진 이유는 0부터 시작하는 숫자와 1부터 시작하는 숫자 때문에 1을 빼주거나 1을 더해주어서 그렇습니다. 결과적으로 우리는 $start_page 값이 1, 11, 21과 같은 값을 가지길 원합니다. 그래서 floor 부분은 0, 1, 2와 같은 식으로 증가하고 거기에 페이지 목록의 수인 10이 곱해지면 0, 10, 20이 됩니다. 마지막으로 1을 더해주면 1, 11, 21이 되어 우리가 원하는 값을 갖게 됩니다.

페이지 리스트의 시작 페이지를 구했으니 마저 마지막 페이지인 $end_page 값도 구해보도록 합시다. 마지막 페이지는 두 가지로 나뉘는데 페이지가 아직 많이 남아 있어서 $page_list_size만큼을 채우는 경우와 페이지가 $page_list_size만큼 없어서 모자라게 출력되는 경우입니다. 일단 페이지가 넉넉히 남아 있을 경우를 계산해보면 다음과 같습니다.

```
$end_page = $start_page + $page_list_size - 1;
```

시작 페이지가 1인 경우 $page_list_size만큼의 페이지를 출력하면 되는데 1 + 10은 11이 됩니다. 1부터 11까지 출력한다면 10개가 아니라 11개가 출력되니까 오류가 생깁니다. 따라서 계산된 결과에 1을 빼주어야 합니다.

이제 페이지가 모자란 경우를 계산해 보겠습니다. 모자란 경우는 다음과 같이 계산된 마지막 페이지가 총 페이지 수보다 큰 경우를 의미합니다. 따라서 다음과 같이 표현할 수 있습니다.

```
if ($total_page < $end_page) $end_page = $total_page;
```

시작 페이지와 마지막 페이지를 검증해보면 다음과 같습니다.

현재 범위(A)	목록 크기(B)	(A-1)/B=C	floor(C)	floor(C) * B	첫 번째	마지막
1 ~ 10	10	0 ~ 0.9	0	0	1	10
11 ~ 20	10	1 ~ 1.9	1	10	11	20
21 ~ 30	10	2 ~ 2.9	2	20	21	30
31 ~ 35	10	3 ~ 3.4	3	30	31	35
1 ~ 15	15	0 ~ 14/15	0	0	1	15
16 ~ 30	15	1 ~ 29/15	1	15	15	30
31 ~ 45	15	2 ~ 44/15	2	30	30	45
46 ~ 50	15	3 ~ 49/15	3	45	45	50

[표 11-12] 시작 페이지와 마지막 페이지의 검증

휴~. 뭔가 많은 산을 넘어온 것 같습니다. 이제 정상이 얼마 남지 않았습니다. 남아 있는 것은 목록 이전의 페이지가 존재하거나 이후의 페이지가 존재하는 경우 다음 목록으로 넘어가기 위한 링크를 만드는 것입니다. 마저 힘내 봅시다. 파이팅!

목록 이전의 페이지로 넘어가려면 반드시 현재의 페이지가 적어도 2번째 목록 페이지에 있어야 합니다. 그래야 이전 목록 페이지로 이동하여 첫 번째 목록 페이지로 이동할 테니까요. 만약 페이지 목록의 크기보다 시작 페이지의 번호가 더 크다면 분명히 이전에 페이지 목록이 존재할 것입니다.

```
if ($start_page > $page_list_size) {
//이전 페이지 목록이 존재함
}
```

또한 페이지 정보는 $no 변수를 통해 이루어지므로 이전 페이지로 돌아가는 것을 $no 값의 변화로 수정해야 합니다. 만약 현재 목록의 첫 번째 페이지가 11일 때, 10페이지로 돌아간다면 목록이 1~10까지로 출력될 것입니다. 이런 원리를 이용하여 현재 목록의 첫 번째 페이지보다 하나 전 페이지의 $no 값을 구해봅시다.

$no 값은 항상 $page_size 값의 배수입니다. 첫 번째 페이지의 $no 값은 0이고 두 번째 페이지의 $no 값은 10, 만약 11번째 페이지라면 $no 값은 100일 것입니다. 즉, 목록을 시작하는 페이지의 $no 값은 다음과 같습니다.

```
($start_page - 1) * $page_size
```

따라서 목록의 첫 번째 페이지보다 한 페이지 이전의 $no 값은 다음과 같이 구할 수 있습니다.

```
$prev_list = ($start_page - 2) * $page_size;
```

이를 이용하여 이전 목록 페이지로 이동하는 부분은 다음과 같이 구현할 수 있습니다.

```
if ($start_page > $page_list_size) {
     $prev_list=($start_page-2)*$page_size;
     echo "<a href=\"$PHP_SELF?no=$prev_list\">◀</a>\n;
}
```

만약 이전 페이지가 존재한다면 좌측 화살표가 나타나고 그렇지 않다면 화살표는 나타나지 않습니다.

다음으로 페이지 리스트를 출력해 봅시다. 이미 앞에서 시작하는 페이지 번호와 끝나는 페이지 번호를 계산했으므로 for 문을 이용하여 출력하면 됩니다.

```
for ($i=$start_page; $i <= $end_page; $i++) {
     $page = ($i-1) * $page_size; // 페이지 값을 no 값으로 변환.
     if ($no != $page){ //현재 페이지가 아닐 경우만 링크를 표시
         echo "<a href=\"$PHP_SELF?no=$page\">";
     }
     echo "$i" //페이지를 표시
     if($no!=$page){
        echo"</a>?";
     }
}
```

마지막으로 다음 페이지 리스트를 구현해 보겠습니다. 다음 페이지 리스트가 필요할 때는 총 페이지가 마지막 리스트의 페이지보다 클 때입니다. 리스트를 다 뿌리고도 더 뿌려줄 페이지가 남았을 때 다음 버튼이 필요할 것입니다.

앞에서 이미 다루었지만 페이지를 $no 값으로 변환하는 공식은 다음과 같습니다.

```
($current_page -1) * $page_size;
```

그래서 현재 페이지에서 다음 페이지의 $no 값은 다음과 같이 표현됩니다.

```
$current_page * $page_size;
```

따라서 목록에서 마지막 페이지 다음의 페이지 $no 값은 $end_page * $page_size로 표현할 수 있습니다. 이를 구현하면 다음과 같습니다.

```
if($total_page > $end_page)
{
  $next_list=$end_page*$page_size
  echo "<a href=$PHP_SELF?no=$next_list>▶</a><p>";
}
```

이제 목록과 페이지 나누기 모두를 만들었습니다. 이제 마지막으로 글쓰기 버튼만 추가해주면 정말

끝입니다. 다음은 완성된 목록 보기의 소스입니다.

list.php

```
<?
//데이터 베이스 연결하기
include "db_info.php";

# LIST 설정
# 1. 한 페이지에 보여질 게시물의 수
$page_size=10;

# 2. 페이지 나누기에 표시될 페이지의 수
$page_list_size = 10;

if (!$no || $no < 0) $no=0;

// 데이터베이스에서 페이지의 첫 번째 글($no)부터
// $page_size만큼의 글을 가져온다.
$query = "SELECT * FROM board ORDER BY id DESC LIMIT $no,$page_size";
$result = mysql_query($query, $conn);

// 총 게시물 수를 구한다.
$result_count=mysql_query("SELECT count(*) FROM board",$conn);
$result_row=mysql_fetch_row($result_count);
$total_row = $result_row[0];
//결과의 첫 번째 열이 count(*)의 결과다.

# 총 페이지 계산
if ($total_row <= 0) $total_row = 0;
$total_page = ceil($total_row / $page_size);

# 현재 페이지 계산
$current_page = ceil(($no+1)/$page_size);
?>
<html>
<head>
<title>초 허접 게시판</title>
<style>
<!--
td {font-size : 9pt;}
A:link {font : 9pt;color : black;text-decoration : none; fontfamily
: 굴림;font-size : 9pt;}
A:visited {text-decoration : none; color : black; font-size : 9pt;}
A:hover {text-decoration : underline; color : black; font-size : 9pt;}
-->
</style>
```

```
44 </head>
45 <body topmargin=0 leftmargin=0 text=#464646>
46 <center>
47 <BR>
48 <!-- 게시판 타이틀 -->
49 <font size=2>자~ 버그를 찾읍시다~ 버그를~~</a>
50 <BR>
51 <BR>
52 <!-- 게시물 리스트를 보이기 위한 테이블 -->
53 <table width=580 border=0 cellpadding=2 cellspacing=1
54 bgcolor=#777777>
55 <!-- 리스트 타이틀 부분 -->
56 <tr height=20 bgcolor=#999999>
57   <td width=30 align=center>
58     <font color=white>번호</font>
59   </td>
60   <td width=370 align=center>
61     <font color=white>제 목</font>
62   </td>
63   <td width=50 align=center>
64     <font color=white>글쓴이</font>
65   </td>
66   <td width=60 align=center>
67     <font color=white>날 짜</font>
68   </td>
69   <td width=40 align=center>
70     <font color=white>조회수</font>
71   </td>
72 </tr>
73 <!-- 리스트 타이틀 끝 -->
74 <!-- 리스트 부분 시작 -->
75 <?
76 while($row=mysql_fetch_array($result))
77 {
78 ?>
79 <!-- 행 시작 -->
80 <tr>
81   <!-- 번호 -->
82   <td height=20 bgcolor=white align=center>
83     <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
84     <?=$row[id]?></a>
85   </td>
86   <!-- 번호 끝 -->
87   <!-- 제목 -->
88   <td height=20 bgcolor=white> 
89     <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
90     <?=strip_tags($row[title], '<b><i>');?></a>
91   </td>
```

```
92    <!-- 제목 끝 -->
93    <!-- 이름 -->
94    <td align=center height=20 bgcolor=white>
95      <font color=black>
96      <a href="mailto:<?=$row[email]?>"><?=$row[name]?></a>
97      </font>
98    </td>
99    <!-- 이름 끝 -->
100   <!-- 날짜 -->
101   <td align=center height=20 bgcolor=white>
102     <font color=black><?=$row[wdate]?></font>
103   </td>
104   <!-- 날짜 끝 -->
105   <!-- 조회수 -->
106   <td align=center height=20 bgcolor=white>
107     <font color=black><?=$row[see]?></font>
108   </td>
109   <!-- 조회수 끝 -->
110 </tr>
111 <!-- 행 끝 -->
112 <?
113 } // end While
114 //데이터베이스와의 연결을 끝는다.
115 mysql_close($conn);
116 ?>
117 </table>
118 <!-- 게시물 리스트를 보이기 위한 테이블 끝-->
119 <!-- 페이지를 표시하기 위한 테이블 -->
120 <table border=0>
121 <tr>
122   <td width=600 height=20 align=center rowspan=4>
123   <font color=gray>
124    
125 <?
126 $start_page = floor(($current_page - 1) / $page_list_size)
127         * $page_list_size + 1;
128
129 # 페이지 리스트의 마지막 페이지가 몇 번째 페이지인지 구하는 부분이다.
130 $end_page = $start_page + $page_list_size - 1;
131
132 if ($total_page < $end_page) $end_page = $total_page;
133 if ($start_page >= $page_list_size) {
134   # 이전 페이지 리스트값은 첫 번째 페이지에서 한 페이지 감소하면 된다.
135   # $page_size를 곱해주는 이유는 글 번호로 표시하기 위해서이다.
136
137   $prev_list = ($start_page - 2)*$page_size;
138   echo "<a href=\"$PHP_SELF?no=$prev_list\">◀</a>\n";
139 }
```

```
140
141 # 페이지 리스트를 출력
142 for ($i=$start_page;$i <= $end_page;$i++) {
143   $page= ($i-1) * $page_size;// 페이지값을 no 값으로 변환
144   if ($no!=$page){ //현재 페이지가 아닐 경우만 링크를 표시
145     echo "<a href=\"$PHP_SELF?no=$page\">";
146   }
147
148   echo " $i "; //페이지를 표시
149
150   if ($no!=$page){
151     echo "</a>";
152   }
153 }
154
155 # 다음 페이지 리스트가 필요할때는 총 페이지가 마지막 리스트보다 클 때이다.
156 # 리스트를 다 뿌리고도 더 뿌려줄 페이지가 남았을 때 다음 버튼이 필요할 것이다.
157 if($total_page > $end_page)
158 {
159   $next_list = $end_page * $page_size;
160   echo "<a href=$PHP_SELF?no=$next_list>▶</a><p>";
161 }
162 ?>
163   </font>
164   </td>
165 </tr>
166 </table>
167 <a href=write.php>글쓰기</a>
168 </center>
169 </body>
170 </html>
```

Section 12

게시판은 사용을 금합니다

가장 기본적인 게시판을 모두 구현했습니다. 그러나 이 게시판은 사용을 금합니다. 왜냐하면 너무나 많은 문제점을 가지고 있기 때문입니다. 게시판을 작성하면서 비밀번호를 제외한 부분에서 보안

요소를 고려한 부분이 한 군데도 존재하지 않습니다. 이것은 매우 위험한 발상이며 특히 웹 프로그램은 모든 사람에게 공개되기 때문에 (무엇보다도 이런 공개되는 게시판은 소스 코드를 누구나 볼 수 있기 때문에) 많은 익명의 사용자로부터 공격을 받을 수 있습니다. 따라서 웹 프로그램을 개발할 때는 반드시 보안이라는 단어를 머릿속에 항상 떠올리고 있어야 합니다. 이 부분이 어떤 보안 허점이 존재하지는 않을까? 혹은 최근에 이러한 공격이 많이 있다는데 내가 만든 프로그램에도 비슷한 공격을 당할 여지는 없을까? 하고 많이 고민해야 합니다.

이 게시판은 배움의 목적으로 일부러 많은 버그와 보안 구멍을 만들어 두었습니다. 왜냐하면 이러한 문제들을 하나씩 고쳐나감으로써 어떤 문제들이 실제로 있으며 어떻게 수정해야 하는지 알아 갈 수 있기 때문입니다. 다음 장에서는 이 장에서 제작한 게시판의 비밀을 모두 파헤쳐서 보다 안전한 게시판으로 거듭나게 만들 것입니다.

Chapter

12

계층형 게시판 만들기

이미 우리는 '11장. 게시판 만들기'에서 가장 기초가 되는 게시판을 배웠습니다. 앞선 게시판은 게시판을 처음 만들어보는 사람에게는 게시판의 기본 원리를 배우기에 별 부족함이 없지만 실제로 홈페이지에 적용하기에는 부족함이 많은 게시판이었습니다.

예전에는 앞서 제작한 게시판 기능만으로도 사용하던 때가 있었지만 강산도 세월에 변해가듯이 게시판도 차츰 성장하여 기존의 게시판의 기능에 여러 가지 기능이 덧붙여지게 되었습니다. 예를 들어 기존 게시판의 경우 어떤 게시물에 대한 응답 글을 쓰면 제목으로 해당 게시물의 응답 글임을 확인하고 찾아 읽어야 하는 문제가 있었습니다. 즉, 너무 일방향적인 게시판 성향을 띄었던 것입니다. 그래서 등장한 것이 응답형 게시판입니다. 응답형 게시판은 기존 게시판에서 해당 글의 덧글을 작성할 수 있게 하고 덧글을 원래 게시물의 하단에 보여줘서 누구나 쉽게 해당 글과 연관된 덧글임을 인지할 수 있게 해주는 게시판입니다. 이 응답형 게시판을 구현하기 위한 방법 중 대표적인 것이 계층 구조를 이용하는 것입니다. 이에 사람들은 응답형 게시판을 계층형 게시판이라고 부르기도 합니다.

이 장에서 만드는 게시판은 실제 홈페이지에 적용할 수 있을만한 비교적 쓸만한(?) 게시판이 될 것입니다. 그렇다면 구현하기 어렵다거나 이해하기 힘든 것이 아닐까 걱정하는 분이 있으리라 생각합니다. 이러한 분들에게 필자의 좌우명이기도 한 문구를 하나 말씀드리려 합니다.

"아직 일어나지 않은 일에 대해 걱정하지 마라!"

실제 어려울지 쉬울지는 닥쳐봐야 아는 것입니다. 미리 겁먹고 두려워할 필요가 없다는 말이지요. 사실 계층형 게시판은 앞서 배운 게시판과 거의 유사합니다. 실제로 몇 개의 파일은 그대로 사용할 수 있을 정도입니다. 그래서 앞서 사용한 소스를 최대한 활용하여 몇 부분 수정하는 것으로 계층형 게시판을 만들어 보겠습니다.

Section 01

계층형 게시판의 형태

계층형 게시판이란 말 그대로 글이 계층을 이룬다는 말입니다. "글이 계층을 이룬다." 뭔가 말이 어려워 보입니다. 그러나 말이 거창한 것치고 속내까지 거창한 것은 별로 없는 것 같습니다. 사실 영업의 측면에서 보자면 작은 기능을 최대한 그럴듯하게 포장하기 위한 최고의 방법이 이름 짓기가 아닐까 생각합니다. 이처럼 계층형 게시판도 별로 어려울 것이 없습니다. 단순히 말하자면 기존 게시판에 답변 기능을 추가한 게시판입니다. 그래서 응답형 게시판, 답변형 게시판 등으로도 불립니다. 영어로는 일반적으로 쓰레드(thread)형 게시판이라고 말을 합니다.

계층형 게시판은 기존 게시판과 달리 답변이 있기 때문에 기존의 게시물에 어떻게 답변 글을 잘 출력할 수 있을지를 고민하는 것으로 시작합니다. 하지만 다행히도 이미 많은 프로그래머와 사용자가 시행착오를 거치면서 다음과 같은 구조가 그럭저럭 쓸만하다고 결론을 내렸습니다.

번호	제 목	글쓴이	날 짜	조회수
9	후훗	브라운	2004-02-07	2
4	네번째 글	브라운	2004-02-07	4
3	세번째 글	브라운	2004-02-07	6
6	->RE:세번째 글	브라운	2004-02-07	3
7	->RE:RE:세번째 글	브라운	2004-02-07	3
8	->RE:RE:RE:세번째 글	브라운	2004-02-07	1
2	두번째 글	브라운	2004-02-07	2
5	->RE:두번째 글	브라운	2004-02-07	0
1	첫번째 글	브라운	2004-02-07	0

1

글쓰기

[그림 12-1] 계층형 게시판의 모양

그림의 글 순서를 잘 고려해가면서 보면 게시물이 어떻게 출력되어야 하는지 어렴풋이 알 수 있을 것입니다. 단순히 말하면 "덧글은 원래 글의 하단에 위치하도록 한다."입니다. 기존 게시판은 게시물의 작성 순서대로 출력되었지만 계층형 게시판은 게시물 사이사이 답변 글이 끼어 있다는 차이가 있습니다.

Section 02

계층형 게시판의 세대 변화

계층형 게시판의 세대 변화를 살펴보면서 보다 자세히 계층형 게시판의 구조에 대해 알아보도록 합

시다. 앞서 게시판은 시간이 지나감에 따라서 변화가 있었다고 했습니다. 여러 가지 자그마한 변화가 많이 있었지만 크고 굵직굵직한 변화를 기준으로 게시판의 세대 변화를 정의해 보았습니다.

① 할아버지 세대 (0세대) : 답변도 그냥 새 글처럼 (계층형이 아님)

번호	제목	종류
4	RE : 두 번째 글	답변 글
3	세 번째 글	새 글
2	두 번째 글	새 글
1	첫 번째 글	새 글

[표 12-1] 0세대 게시판

이 게시판은 계층형 게시판은 아니지만 응답형 게시판의 시작이라고 할 수 있습니다. 답변의 필요성으로 생기긴 하였지만 제목에 답변 글을 표시하는 문구(일반적으로 'RE')가 추가되었을 뿐 실제로는 답변 글이 새 글처럼 등록이 되는 문제가 있습니다.

② 아버지 세대 (1세대) : 답변 글이 어찌 새 글이여~ 답변은 답변답게!! (업데이트형)

번호	제목	종류
4	세 번째 글	새 글
3	두 번째 글	새 글
2	RE : 두 번째 글	답변 글
1	첫 번째 글	새 글

[표 12-2] 1세대 게시판

초기의 계층형 게시판입니다. 현재랑 모습은 같지만 구현방법이 지금과는 차이가 있습니다. 일반적으로 업데이트형이라고 불리는 이 게시판의 원리는 다음과 같습니다.

번호	입력된 순서	제목	종류
3	3	세 번째 글	새 글
2	2	두 번째 글	새 글
1	1	첫 번째 글	새 글

[표 12-3] 1세대 게시판의 원리(1)

1, 2, 3 이렇게 세 개의 글이 있고 2번 글에 답변이 달아 보겠습니다.

번호	입력된 순서	제목	종류
4(+1)	3	세 번째 글	새 글
3(+1)	2	두 번째 글	새 글

2	4	RE : 두 번째 글	답변 글
1	1	첫 번째 글	새 글

[표 12-4] 1세대 게시판의 원리(2)

답변 글은 항상 원문의 아랫줄에 위치해야 하기 때문에 첫 번째 글과 두 번째 글 사이에 끼워 넣습니다. 출력 순서상으로 첫 번째 글 다음이 답변 글이 되므로 답변 글의 번호는 2번이 됩니다. 그런데 2번이 중복되므로 2번 글은 3번이 되게 합니다. 하지만 역시 3번 글이 중복되므로 새로 끼어든 답변 글 위에 존재하는 모든 글은 1씩 번호가 밀리게 됩니다.

이렇게 글 번호를 저장하면 글 번호를 기준으로 정렬했을 때 위와 같은 모양이 출력될 것을 예상할 수 있습니다. 이 부분이 충분히 이해가 되셨다면 왜 이 게시판을 업데이트형이라고 불리는지 알 수 있을 것입니다. 어떤 글에 답변이 달리면 답변이 달리는 글부터 그 이후 글 모두의 글 번호가 1씩 증가(업데이트)하게 만들어야 합니다. 그래서 글 번호를 업데이트해준다고 해서 업데이트형이라고 불립니다.

하지만 이 게시판은 큰 문제를 하나 가지고 있습니다. 업데이트형 게시판의 고질적인 문제로 글의 양이 적고 답변 글이 거의 없는 경우에는 특별한 이상을 느낄 수 없지만 글이 많다거나 자주 답변 글이 등록되는 경우에는 글 번호를 매번 수정해야 하기 때문에 많은 과부하가 일어나는 것입니다.

만약 글이 10000개가 있는데 10번째 글에 답변을 달게 되면 9990개의 글 번호를 업데이트해야 합니다. 물론 대부분의 경우 답변이 최근 글에 달리기 때문에 이처럼 수천 개의 글을 수정하게 되는 일은 드물지만 그렇다고 무시하고 사용하기에는 잠재적인 문제점이 적다고 할 수만은 없습니다.

③ 우리 세대 (2세대) : 답변도 답변이지만 성능도 고려해야지~(반업데이트형)

번호	깊이	제목	종류
300	0	세 번째 글	새 글
200	0	두 번째 글	새 글
199	1	RE : 두 번째 글	답변 글
100	0	첫 번째 글	새 글

[표 12-5] 2세대 게시판

1세대의 업데이트로 인해 부하가 많이 걸리는 단점을 보완하고자 만든 게시판입니다. 반업데이트형이라는 말에서부터 업데이트를 하고 있음을 알 수 있는데요. 1세대와 다른 점은 부분적으로만 업데이트를 한다는 것입니다.

예를 들면 위처럼 글 번호를 100의 간격으로 등록하고 200번의 글에 답변이 달리면 그 답변 글을 199번으로 만드는 것입니다.

번호	깊이	제목	종류
300	0	세 번째 글	새 글
200	0	두 번째 글	새 글
199	1	RE : 두 번째 글의 두 번째 답변	답변 글
198	1	RE : 두 번째 글	답변 글
100	0	첫 번째 글	새 글

[표 12-6] 2세대 게시판의 원리(1)

200번 글에 다시 답변이 달리면 199번 글을 198번 글로 바꾼 다음에 199번 글로 등록합니다.

번호	깊이	제목	종류
300	0	세 번째 글	새 글
200	0	두 번째 글	새 글
199	1	RE : 두 번째 글의 두 번째 답변	답변 글
198	2	RE : 두 번째 글의 두 번째 답변의 답변	답변 글
197	1	RE : 두 번째 글	답변 글
100	0	첫 번째 글	새 글

[표 12-7] 2세대 게시판의 원리(2)

199번 글에 답변이 달리면 100번과 199번 사이에 있는 글들을 모두 -1씩 업데이트해 줍니다. 만약 198번 글이 있었다면 197번 글이 되었을 테니 198번 자리가 비게 되지요? 그 자리에 답변 글을 추가해서 넣습니다. 이렇게 되면 이전 방식과 달리 전체를 업데이트하는 게 아니라 이 경우에는 최대 99개만 업데이트됩니다. 새 글과 새 글 사이만 업데이트되기 때문입니다. 이러한 방법으로 성능을 좀 더 향상시키면서 계층형 게시판을 구현할 수 있습니다.

그러나 눈치채신 분도 계시겠지만 여기는 제약이 존재합니다. 어떤 새 글에 대한 답변의 수가 한정되어 있다는 것입니다. 번호의 차이를 100이 아닌 1000으로 바꾼다면 답변의 수가 늘어나겠지만 한 글에 대한 답변이 100개가 달리기란 정말 어려운 일입니다. 그러나 불가능한 일은 아니지요. 그래서 적당한 수를 생각해서 그것을 기준으로 프로그래밍하면 됩니다.

깊이는 단지 제목에서 들여쓰기를 하기 위해 만든 것입니다. 답변 글이 새 글과 구분되기 위해 새 글에 대한 답변인지, 답변에 대한 답변인지를 깊이로 나타냅니다. 이를 통해 1칸의 여백을 둘지 2칸의 여백을 둘 지가 결정됩니다.

우리는 이 방식을 사용하여 계층형 게시판을 구현할 것입니다. 다음 세대 게시판이 있는데 이 방식으로 구현하는 것은 첫째 알고리즘이 간단하고, 둘째 구현이 간단하고, 셋째 비교적 훌륭한 성능이기 때문입니다.

여기서 잠깐

이 게시판은 하나의 글에 대한 답변 글의 수가 제한되어 있다는 단점이 있습니다. 이를 해결하기 위해 100 단위의 글 번호를 생성하는 대신 테이블에서 몇 개의 필드를 추가하여 답변 글에 대한 제한을 없앨 수 있습니다.

④ 아들, 딸 세대(3세대) : 부분 업데이트도 부담된다~ 업데이트 반대!! (비업데이트형)

비업데이트형 게시판은 1, 2세대와 달리 다른 글에 대한 업데이트를 하지 않는 방법을 사용합니다. 이를 구현하기 위한 여러 가지 방법이 있는데, 일반적으로 문자열을 사용하여 정렬한다거나 소수점을 이용하여 정렬하는 등의 방법을 사용합니다.

입력 순서	정렬 값	제목	종류
3	02	두 번째 글	새 글
1	01	첫 번째 글	새 글
5	0102	RE : 첫 번째 글의 두 번째 답변	답변 글
2	0101	RE : 첫 번째 글	답변 글
4	010101	RE : 첫 번째 글 답변의 답변	답변 글

[표 12-8] 3세대 게시판 – 문자열을 사용하는 예

입력 순서	정렬 값	제목	종류
2	2	두 번째 글	새 글
3	1.001	RE : 두 번째 글	답변 글
5	1.001002	RE : 두 번째 글 답변의 두 번째 답변	답변 글
4	1.001001	RE : 두 번째 글 답변의 답변	답변 글
1	1	첫 번째 글	새 글

[표 12-9] 3세대 게시판 – 소수점을 사용하는 예

그러나 이 방법은 비교적 간단하지 않고 복잡한 알고리즘을 요구합니다. 많은 알고리즘이 연구되고 있으나 그중에는 반업데이트형보다 성능이 못한 경우도 많습니다. 하지만 좋은 알고리즘과 개발 환경에서의 뒷받침이 있다면 충분히 기대해볼 만합니다.

여기서 잠깐

게시판의 흐름을 보면 점차로 답변 기능의 비중이 줄어들고 간단한 덧글(코멘트 기능)의 비중이 높아지고 있습니다. 대부분의 경우 짧은 코멘트로도 충분히 답변 기능을 대체할 수 있기 때문에 그런 듯 합니다. 또한 블로그가 인기를 끌면서 답변 글 기능을 사용하지 않는 형태의 게시판이 많이 사용되고 있습니다. 답변 기능 대신에 짧은 코멘트 기능을 강화하여 코멘트에 대해 다시 코멘트를 달 수 있는 게시판으로 변모해가고 있습니다.

테이블 설계

항목	영어 항목	변수형	크기	비 고
글의 번호	id	int	11	글의 번호는 숫자
글쓴이	name	varchar	20	varchar는 많이 길지 않는 문자열
이메일 주소	email	varchar	30	
글의 비밀번호	pass	varchar	10	
글의 제목	title	varchar	70	
글의 내용	content	text		충분히 긴 문자열
글 쓴 날짜	wdate	datetime		날짜 및 시간
IP 주소	ip	varchar	16	
조회 수	see	int	11	

[표 12-10] 이전 게시판의 스키마

자 여기다가 몇 가지를 덧붙여 계층형 게시판으로 만들어 보겠습니다. 앞서 우리는 반업데이트형 게시판을 만들 것이라고 했습니다. 반업데이트형 게시판을 설명하면서 기존 게시판과 달라지는 부분에 대해 슬쩍 언급했었는데 기억이 나시나요? 두 가지가 있는데 다음과 같습니다.

① 일정 간격(100 또는 1000 등등)으로 증가하는 내부적인 글 번호

② 답변의 경우 들여쓰기를 위한 깊이 값

그래서 이 부분을 추가해야 합니다. 위의 각각 항목에 이름을 정해봅시다.

우선 내부적인 글 번호는 계층형 게시판을 구성하기 위한 번호이니 'thread'라고 이름을 지어봅니다. 물론 'inner_id'라던가 'virtual_id' 등 원하시는 이름을 정해서 사용해도 무방합니다. 저는 계층형 게시판의 영문 이름이 thread 게시판이어서 명색이 thread 게시판인데 thread 필드가 없는 것이 불쌍하여 억지로 만들어 보았습니다. 그리고 글의 깊이는 깊이를 나타내는 depth라는 단어를 사용하기로 합니다.

이제 새로 추가된 필드에 변수형을 지정하면 새로운 게시판의 스키마가 완성됩니다.

내부적인 글 번호를 뜻하는 thread는 앞서 설명하였듯이 일정한 값(100 또는 1000 등등)으로 증가하는 정수값입니다. 크게 잡아서 1000씩 증가하는 값이라고 했을 때 10만 개의 새 글이 등록되면 100,000,000(1억)이란 값을 갖게 됩니다. Integer 값은 −2,147,483,648 ~ 2,147,483,647 사이의 값이고 부호없는 정수(unsigned integer)는 0 ~ 4,294,967,295까지의 값을 가질 수 있습니다. thread는 음수를 갖지 않는 정수이므로 부호 없는 정수로 설정하면 총 4백만 개 이상의 글을 쓸 수 있게 됩니다. 400만 개의 글이 올라올 만한 게시판이라면 이 게시판보다는 다른 더 나은 성능의 게시판을 선택하는 것이 좋을 터이니 이로써 충분한 듯합니다.

글의 깊이를 의미하는 depth는 thread 값의 일정한 증가 값에 따라 변하지만 아무리 크게 잡아도 Integer 범위를 넘어서지는 않을 것이니 이 또한 Integer로 설정하도록 합니다.

새로 추가되는 thread와 depth 필드의 정의가 끝났습니다. 변경된 계층형 게시판의 스키마는 다음과 같습니다.

항목	영어 항목	변수형	크기	비 고
글의 번호	id	Int	11	글의 번호는 숫자
내부 번호	**thread**	**Int**	**11**	**일정 값으로 증가하는 값**
글의 깊이	**depth**	**Int**	**11**	**들여쓰기를 위한 값**
글쓴이	name	varchar	20	varchar는 많이 길지 않는 문자열
이메일 주소	email	varchar	30	
글의 비밀번호	pass	varchar	10	
글의 제목	title	varchar	70	
글의 내용	content	text		충분히 긴 문자열
글 쓴 날짜	wdate	**int**		시간을 UnixTime으로 저장
IP 주소	ip	varchar	16	
조회 수	view	int	11	

[표 12-11] 수정된 계층형 게시판의 스키마

위 표에서 볼드 부분이 추가되거나 수정된 부분입니다. 추가된 부분은 앞에서 이미 다루었으니 수정된 부분에 대해 알아봅시다. 수정된 부분은 글을 쓴 날짜인데 반드시 수정해야 하는 건 아니고 날짜를 다루는 다른 방식을 배우려고 일부러 수정했습니다. 원래는 datetime형이었는데 int형으로 정의했습니다. 이 부분에서 날짜를 어떻게 숫자로 표시할지 의문을 가지는 분이 계실 듯합니다. 일반적으로 날짜와 시간은 "몇 년 몇 월 몇 일"과 같이 표시되는 값인데 숫자로 이를 어떻게 표현할지 막막할 것입니다. 그러나 PHP에는 유닉스 타임(unixtime)이라는 것이 있습니다.

유닉스 타임이란 1970년 1월 1일부터 현재 시간까지를 초 단위로 계산한 값입니다. 즉, 1970년 1월 1일부터 몇 초가 흘렀느냐 하는 값입니다. 데이터베이스에서 특별히 만들어 놓은 날짜 시간 형식의 타입을 쓰지 않고 굳이 숫자 값으로 변환해서 쓰는 이유는 날짜를 다루거나 할 때 여러 가지로 편리한 점이 많기 때문입니다. 일반적으로 게시판에서 int형의 날짜는 별 의미가 없습니다. 왜냐하면 게시판에서 날짜 시간의 의미는 단순히 언제 글을 작성하였느냐는 의미밖에 없기 때문입니다. 복잡한 날짜 계산이 잦은 프로그램의 경우 날짜 시간 형식보다 int형을 사용하는 것이 더 편리합니다. 일단은 날짜 시간을 이런 식으로도 처리할 수 있구나 하고 생각하기 바랍니다.

각 항목별 타입과 크기를 정하였으니 마지막으로 항목별 NULL 허용 여부와 키(Primary Key) 등을 설정해 보도록 합시다.

항목	영어 항목	키	필수 항목	기타
글의 번호	id	Primary Key	필수	1씩 증가하는 0 이상의 수
내부 번호	thread		필수	일정 값으로 증가하는 값
글의 깊이	depth		필수	기본값 0
글쓴이	name		필수	
이메일 주소	email		옵션	
글의 비밀번호	pass		필수	
글의 제목	title		필수	
글의 내용	content		필수	
글 쓴 날짜	wdate		필수	
IP 주소	ip		필수	
조회 수	view		필수	기본값 0

[표 12-12] NULL과 키 제약 정보

나머지는 기존 게시판과 동일하니 변경된 부분만 언급하면, 내부 번호는 글 정렬의 기준이 되는 값이기 때문에 반드시 필수적인 값입니다. 글 입력 시에 항상 자동으로 입력을 해주어야 하는 부분입니다. 글의 깊이는 답변 글이 아닌 경우에는 기본값 0이 설정되게 하면 됩니다. 답변의 깊이가 깊어질수록 depth의 값은 커지기 때문에 0은 답변 글이 아닌 일반 글 임을 의미합니다.

테이블 생성문 만들기

테이블 스키마가 완성되었으니 테이블 생성문을 만들어 봅시다. 일단 아래의 생성문을 보지 말고 직접 작성해보기 바랍니다.

```
CREATE TABLE threadboard (
  id int(11) unsigned NOT NULL auto_increment,
  thread int(11) unsigned NOT NULL,
  depth int(11) unsigned NOT NULL default '0',
  name varchar(20) NOT NULL,
  email varchar(30),
  pass varchar(10) NOT NULL,
  title varchar(70) NOT NULL,
  content text NOT NULL,
  wdate int(11) NOT NULL,
  ip varchar(16) NOT NULL,
  view int(11) NOT NULL default '0',
 PRIMARY KEY  (id)
) ENGINE=MyISAM DEFAULT CHARSET=euckr;
```

이미 테이블 생성문을 만들어본 경험이 있으니 이해하는데 특별한 문제는 없으리라 생각합니다. 이 생성문을 데이터베이스에 접속하여 실행하면 테이블이 생성됩니다.

```
mysql> desc threadboard;
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
+----------+------------------+------+-----+---------+----------------+
| id       | int(11) unsigned | NO   | PRI | NULL    | auto_increment |
| thread   | int(11) unsigned | NO   |     |         |                |
| depth    | int(11) unsigned | NO   |     | 0       |                |
| name     | varchar(20)      | NO   |     |         |                |
| email    | varchar(30)      | YES  |     | NULL    |                |
| pass     | varchar(10)      | NO   |     |         |                |
| title    | varchar(70)      | NO   |     |         |                |
| content  | mediumtext       | NO   |     |         |                |
| wdate    | int(11)          | NO   |     |         |                |
| ip       | varchar(16)      | NO   |     |         |                |
| view     | int(11)          | NO   |     | 0       |                |
+----------+------------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

테이블이 만들어졌으니 이제 본격적으로 계층형 게시판을 제작해 보도록 합시다. 아자! 아자! 파이팅!

Section 03 데이터베이스 연결 파일

데이터베이스에서 정보를 검색하거나 값을 입력하려면 우선 데이터베이스에 연결하는 코드를 작성해야 한다고 앞서 말씀드렸습니다. 앞서 사용한 db_info.php 파일을 약간 수정하여 다음과 같은 코드를 만들어 보았습니다.

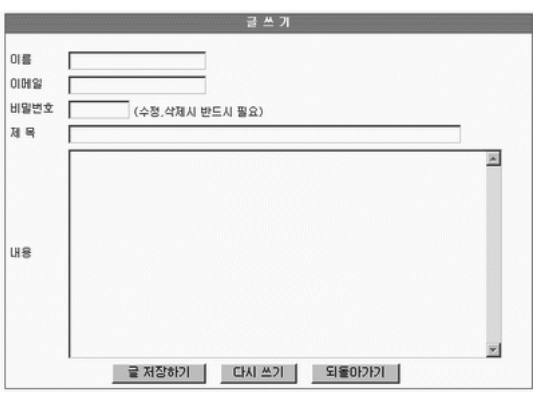
```
<?
  $board="threadboard";
  $conn=mysql_connect("localhost","아이디","패스워드");
  mysql_select_db("데이터베이스이름", $conn);
  mysql_query('set names euckr');
?>
```

수정된 부분은 딱 한 가지로 $board 변수가 생겼습니다. 이는 약간의 편리를 위한 변수로 이제부터는 이 변수를 통해 게시판 테이블 이름을 대신해서 사용하려고 합니다. 즉, 기존에는 SELECT * FROM threadboard라고 하였던 것을 SELECT * FROM $board라고 할 것이라는 말입니다. 왜

굳이 이렇게 바꾸느냐고 하실 분이 있으리라 생각합니다. 이렇게 변수로 변경하는 이유는 테이블 이름이 바뀌거나 하는 경우 여러 개의 소스 파일에서 모두 테이블 이름을 수정해야 하는 번거로움을 막기 위함입니다. 지금 만들어 가는 계층형 게시판은 다중 게시판이 아니기 때문에 홈페이지에 여러 개의 게시판을 추가해야 할 경우 디렉토리 이름을 달리하여 소스 코드를 복사해서 사용해야 합니다. 그러면 이때 테이블 이름이 각각 달라지기 때문에 이를 수정해야 합니다. 만약 위와 같이 변수로 지정되어 있다면 db_info.php 파일 하나만 수정하는 것만으로 수정 작업이 끝날 것입니다.

Section 04

글 쓰기 – write.php



[그림 12-2] 글 쓰기 페이지

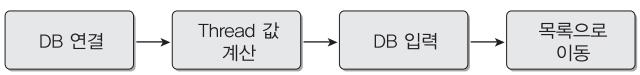
글을 입력받는 글 쓰기 폼은 기존 게시판과 동일합니다. 기존의 파일을 그대로 사용하면 됩니다.

Section 05

글 쓰기 반영 – insert.php

글을 입력받았으니 입력된 정보를 데이터베이스에 저장해 봅시다.

계층형 게시판으로 변경되면서 이전 게시판과 차이가 나는 부분이 있으니 그 부분을 유념하면서 따라오면 어렵지 않게 이해할 수 있을 것입니다. 일단 글 쓰기의 기능을 크게 구분해보면 다음과 같습니다.



[그림 12-3] 글 쓰기 반영의 과정

절차를 살펴보면 일단 데이터베이스 작업을 위해서 데이터베이스에 연결합니다. 그 후 내부 번호를 구하고 thread 값을 계산합니다. 이전 게시판과 차이가 나는 부분이 바로 이 부분입니다. 이전 게시판에는 내부 글 번호와 외부 글 번호의 구분이 없었기 때문에 굳이 이러한 작업을 할 필요가 없었습니다. 그러나 계층형 게시판에서는 100 혹은 1000단위씩 글의 번호가 증가해야 하기 때문에 현재 존재하는 글 중에서 가장 큰 thread 값을 찾아서 다음 thread 값을 계산하는 과정이 필요합니다.

thread 값의 계산이 모두 끝났다면 이제 이전 게시판과 마찬가지로 입력된 글 정보를 데이터베이스에 전송하여 등록하면 됩니다. 또한 글이 잘 등록되면 목록으로 이동시킵니다.

데이터베이스 연결

이제 데이터베이스에 접속하는 것은 누워서 떡 먹기 일 거라는 생각이 듭니다. db_info.php 파일을 인클루드합니다.

```
include "db_info.php";
```

thread 값 계산

thread 값을 계산해 봅시다. 이제부터 thread 값은 여유 있게 1000씩 증가시키도록 하겠습니다. 첫 번째 글의 thread 값은 1000이 될 것입니다. 두 번째 글의 thread 값은 당연히 2000이 되겠지요. 그렇다면 답변 글을 작성하지 않았다고 가정하고 N번째 새 글은 thread 값이 얼마일까요? 네.

맞습니다. N*1000이 됩니다.

하지만 N번째 글을 작성할 때까지 답변 글이 하나도 없으리라고 생각할 수는 없습니다. 분명히 한두 개쯤은 답변 글이 있을 겁니다.

예를 들어 총 4개의 글이 있는데 1, 2, 4번째 글은 모두 새 글이고 3번 글은 2번의 답변 글이라고 가정해 봅시다. 그렇게 되면 다음과 같은 thread 값을 가지게 될 겁니다.

번호	thread
1	1000
2	2000
3	1999
4	3000

[표 12-13] 게시물의 예

4번째 글의 thread 값 3000은 어떻게 나온 것일까요? 바로 2번 글의 thread 값 2000에 1000을 더한 값이 됩니다. 그렇다면 왜 3번 글이 아니라 2번 글의 thread 값을 이용한 것일까요? 그 이유는 4가 입력되기 전 상황을 살펴보면 2000이란 값이 가장 큰 값을 갖기 때문입니다. 만약 3번 글이 답변 글이 아닌 일반 글이었다면 3번이 가장 큰 값을 가지고 있었을 것입니다. 그렇다면 3번 글의 thread 값에 1000을 더했을 것입니다.

이 과정을 단순화하면 "기존의 글 중에서 thread 값이 가장 큰 것을 찾아서 그 값에 1000을 더해주면 된다."가 될 수 있습니다. 이렇게 단순하면 좋으련만 여기에는 문제가 있습니다. 위와 같은 상황에서 글쓴이가 2번 글을 삭제하게 되는 경우 때문입니다.

번호	thread
1	1000
~~2~~	~~2000~~
3	1999
4	3000

[표 12-14] 답변 글이 있는 2번 글을 삭제한 경우

2번 글이 삭제되면 thread 값 중 가장 큰 값이 1999가 됩니다. 여기에 1000을 더하면 2999가 되어버립니다. 2999는 답변 글의 번호입니다. 이미 알고 계시겠지만 답변 글이 아닌 일반 글은 모두 1000의 배수가 되어야 합니다. 1000의 배수가 아닌 1999와 같은 thread 값을 가지는 글은 모두 답변 글입니다. "기존 게시판에서 1씩 증가하는 글 번호를 1000씩 증가하게 하자"고 정의한 것에서부터 그 이유를 찾을 수 있습니다.

그런데 2번 글이 삭제된 상황에서도 4번 글의 thread 값은 여전히 3000이 되어야 합니다. 1999 다

음의 1000의 배수는 2000인데 왜 3000이 되어야 할까요? 그 이유는 새 글의 thread 값이 2000이 되면 난데없이 새 글에 답변 글이 생기기 때문입니다. 1001부터 1999까지의 thread 값을 갖는 글은 모두 thread 값이 2000인 글의 답변 글입니다. 따라서 새 글의 thread 값을 2000으로 정해버리면 1999 값을 갖는 3번 글이 4번 글의 답변 글이 되는 오류가 생깁니다.

물론 다음의 경우와 같이 답변 글이 존재하지 않는 경우에 글이 삭제되면 아무 상관이 없습니다.

번호	thread
1	1000
2	2000
~~3~~	~~3000~~
4	3000

[표 12-15] 답변 글이 없는 3번 글을 삭제한 경우

3번 글이 삭제되면 가장 큰 값이 2000이 되어 다음번 값은 3000이 됩니다. 3000이 되어도 무방한 것이 3번 글에 답변 글이 존재하지 않기 때문에 새로 등록된 4번 글의 답변 글이 갑자기 생겨버릴 염려가 없기 때문입니다. 또한 4번 글에 답변 글이 등록된다고 하더라도 2999번이 되어 정상적으로 사용할 수 있게 됩니다.

다시 처음의 상황으로 돌아가 봅시다.

번호	thread
1	1000
~~2~~	~~2000~~
3	1999
4	3000

[표 12-16] 답변 글이 있는 2번을 삭제한 경우

위와 같은 상황에서 4번이 3000이란 값을 가지려면 1999를 통해서 3000을 얻어야 합니다. 한번 곰곰이 생각해 보세요. 어떻게 하면 항상 올바른 thread 값을 얻을 수 있을까요?

야구에는 영구결번이라는 것이 있습니다. 영구결번은 은퇴한 유명선수의 등 번호를 다른 선수들이 사용하지 못하도록 영원히 비워두는 번호를 의미하는데요. 얼마 전 국내에서 포수로 명성을 떨쳤던 이만수 선수가 미국 메이저리그 화이트 삭스에서 코치로 영구결번이 되는 영광을 얻기도 했습니다. 왜 갑자기 야구 이야기를 꺼냈느냐구요? 쌩뚱 맞기는 하지만 이처럼 답변이 있는 글이 삭제되면 그 번호에 영구결번을 부여하도록 합니다. 단, 답변 글까지 모두 삭제되는 경우에는 반납하기로 합시다.

그렇다면 영구결번된 그 값을 계산해내고 거기에 1000을 더하면 되겠네요. 그렇다면 1999 혹은 1950으로부터 2000이란 영구결번 값을 구해내려면 어떻게 해야 할까요? 여러 가지 방법이 있겠습니다만 다음과 같이 소수점을 이용하기로 합니다.

① 1999을 1000으로 나눈다.

② 소수점으로 나온 값을 올림한다.

③ 결과값에 다시 1000을 곱한다.

④ 마지막으로 결과값에 1000을 더한다.

1999를 1000으로 나누면 1.999가 됩니다. 이 값을 올림을 하면 2가 되겠죠? 얻어진 값에다 다시 1000을 곱하면 짜잔~ 2000이 됩니다. 영구결번 값을 구했군요. 영구결번 값에 1000을 더하면 3000. 새로운 thread 값이 계산되었습니다.

이를 수식으로 나타내면 다음과 같습니다.

```
ceil( thread 최대값 / 1000 ) * 1000 + 1000;
```

이제 실제로 소스 코드를 작성해 봅시다. 일단 thread 값의 최대값을 구합니다. 데이터베이스에 존재하는 값에서 최대값을 찾으려면 MAX라는 MySQL 내장 함수를 사용하면 됩니다. 따라서 쿼리는 다음과 같이 작성할 수 있습니다.

```
SELECT max(thread) FROM $board
```

MySQL 내장 함수인 MAX는 해당 컬럼의 값 중에서 가장 큰 값을 찾아서 결과로 되돌려 줍니다. 따라서 이 쿼리의 결과를 위와 같은 수식으로 처리하면 새로운 thread 값을 얻을 수 있습니다.

```
//현재 글 중에서 가장 큰 값을 가져온다.
$max_thread_result = mysql_query("SELECT max(thread) FROM $board",
$conn);
$max_thread_fetch = mysql_fetch_row($max_thread_result);

//새 글의 thread 값을 계산한다.
$max_thread = ceil($max_thread_fetch[0]/1000)*1000+1000;
```

데이터베이스에 등록

새로운 thread 값을 계산하였으니 입력된 글의 정보를 데이터베이스에 등록하도록 합시다.

우선 입력해야 하는 필드에 대해 정리를 해보면 다음과 같습니다.

항목	영어 항목	필수 항목
글의 번호	id	필수
내부 번호	thread	필수
글의 깊이	depth	필수
글쓴이	name	필수
이메일 주소	email	옵션
글의 비밀번호	pass	필수
글의 제목	title	필수
글의 내용	content	필수
글 쓴 날짜	wdate	필수
IP 주소	ip	필수
조회 수	view	필수

[표 12-17] 저장해야 할 필드 정보

입력해야 하는 항목은 총 11개입니다. 이 중에 이메일을 제외한 모든 값이 필수적으로 입력되어야 합니다. 이를 염두해 두면서 쿼리를 작성해 봅시다.

```
INSERT intO $board
(thread, depth, name, pass, email, title, view, wdate, ip, content)
VALUES
($max_thread, 0, '$_POST[name]', '$_POST[pass]', '$_POST[email]', '$_
POST[title]', 0, UNIX_TIMESTAMP(), '$REMOTE_ADDR', '$_POST[content]');
```

여기서 기존 게시판과 달리 $_POST[항목]로 모두 변경했습니다. 앞에서도 언급했듯이 register_globals=Off로 되어 있으면 $name과 같은 방법으로 폼 입력값을 얻을 수 없습니다. 그래서 register_globals=On으로 해서 사용하게 했으나 호스팅과 같은 환경에서는 php.ini 파일을 수정할 수 있는 권한이 없기 때문에 관리자가 Off로 해두면 사용자는 이를 수정할 수 있는 방법이 없습니다. 따라서 범용적으로 게시판을 사용하기 위해서 이제부터는 모두 외부로부터 넘어온 변수에 대해 출처가 드러나도록 전역 변수를 이용하여 사용하도록 합니다. 실제로는 범용성뿐만 아니라 이것만으로 좀 더 안전한 프로그램으로 변모할 수 있습니다.

또한 UNIX_TIMESTAMP()라는 MySQL 내장 함수를 사용하였는데요. 기존에는 날짜 시간 항목이 datetime형이었기 때문에 NOW() 함수를 사용했었습니다. 그런데 계층형 게시판의 경우 int형으로 변환했기 때문에 그대로 NOW() 함수를 써버린다면 년도 값이 저장되어 버립니다. 왜냐하면 NOW() 함수의 결과가 2007-01-01 10:00:00과 같아서 2007까지는 숫자로 인식하여 입력하고 그 뒤부터는 하이픈 문자 때문에 무시됩니다. 그래서 2007만 입력되는 것이죠. 이번에 사용되는 UNIX_TIMESTAMP() 함수는 유닉스 시간을 되돌려주는 MySQL 내장 함수입니다. 앞서 int형의 날짜 시간 항목에 유닉스 시간 값을 저장할 것이라고 언급하였는데 기억나시나요? 왜 UNIX_TIMESTAMP() 함수를 사용하였는지 이해했으리라 생각합니다.

완성된 쿼리를 $query 변수에 저장하고 데이터베이스에 쿼리를 전달합니다.

```
$query = "INSERT intO $board (thread, depth, name, pass, email, title,
view, wdate, ip, content)
VALUES ($max_thread, 0, '$_POST[name]', '$_POST[pass]',
'$_POST[email]', '$_POST[title]', 0, UNIX_TIMESTAMP(), '$REMOTE_ADDR',
'$_POST[content]')";
$result=mysql_query($query, $conn);
```

올바른 쿼리를 전송했다면 데이터베이스와의 연결을 종료합니다.

```
mysql_close($conn);
```

데이터베이스에 글이 저장되었으니 마지막으로 사용자에게 글이 제대로 잘 저장되었음을 알려주고 글 목록으로 이동합니다.

```
echo ("<meta http-equiv='Refresh' content='1; URL=list.php'>");
```

수정된 insert.php 소스 파일은 다음과 같습니다.

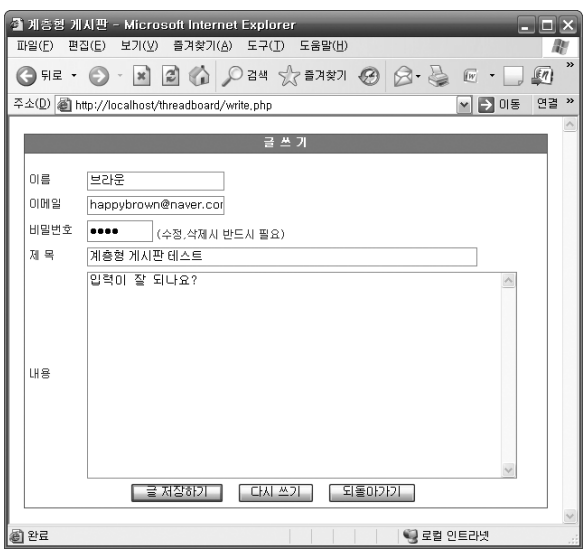
insert.php

```
1  <?
2  //데이터 베이스 연결하기
3  include "db_info.php";
4
5  //thread 값을 계산한다.
6  $query = "SELECT max(thread) FROM $board";
7  $max_thread_result = mysql_query($query, $conn);
8  $max_thread_fetch = mysql_fetch_row($max_thread_result);
9
10 $max_thread = ceil($max_thread_fetch[0]/1000)*1000+1000;
11
12 $query = "INSERT intO $board (thread, depth, name, pass, email,
13 title, view, wdate, ip, content)
14 VALUES ($max_thread, 0, '$_POST[name]', '$_POST[pass]',
15 '$_POST[email]', '$_POST[title]', 0,
16 UNIX_TIMESTAMP(), '$REMOTE_ADDR', '$_POST[content]')";
17 $result=mysql_query($query, $conn);
18
19 //데이터베이스와의 연결 종료
20 mysql_close($conn);
21
22 // 새 글 쓰기인 경우 리스트로
23 echo ("<meta http-equiv='Refresh' content='1; URL=list.php'>");
24 ?>
```

```
25 <center>
26 <font size=2>정상적으로 저장되었습니다.</font>
```

글 쓰기 테스트

글 쓰기 폼과 데이터베이스 입력부를 완성하였으니 이를 이용하여 글을 하나 등록해 보겠습니다.



[그림 12-4] 글 쓰기 테스트

물론 글 저장하기를 클릭하면 잠깐 글이 저장되었음을 알리는 메시지와 페이지를 찾을 수 없다는 페이지를 보게 될 것입니다. 이는 글이 입력된 후 목록으로 돌아가게 되는데 아직 list.php 파일을 만들지 않아서 일어나는 것이니 일단은 무시해도 좋습니다.

글이 정말로 잘 저장되었는지를 확인하기 위해 phpmyadmin을 통해서 테이블에 저장된 값을 확인해 보았습니다.

id	thread	depth	name	email	pass	title
1	1000	0	브라운	happybrown@naver.com	1234	계층형 게시판 테스트

[그림 12-5] 테이블에 저장된 값

이외에 내용이나 IP 등과 같은 다른 정보도 잘 저장되어 있음을 확인할 수 있습니다.

여기서 사용하는 글 입력 부분도 기존 게시판과 달라진 것이 거의 없듯이 많은 문제를 가지고 있습니다. 이 문제들은 다음 장에서 자세히 다룹니다.

Section 06

글 목록 – list.php

11장에서는 리스트 페이지 구현을 위해 많은 설명이 필요하여 제일 뒤로 설명을 미루었지만 이미 앞서 대부분의 로직에 대해 배웠으니 글 목록을 여기서 다룹니다. 이전의 굉장히 길었던 글 목록에 대한 설명을 떠올리며 벌써부터 머리를 싸매는 분이 분명 있으리라 생각되는데 계층형 게시판으로 변하면서 바뀐 것은 몇 줄 밖에 안 되니 너무 걱정하지 마시기 바랍니다.

일단 무엇이 바뀌었는지부터 알아봅시다.

① 목록 출력을 위한 쿼리문

② 목록 출력할 때 답변 글 들여쓰기

③ 날짜 출력

목록 출력은 기존에는 id 값을 기준으로 정렬했던 것을 thread 값을 이용하여 정렬하기로 변경했으니 SELECT 쿼리문을 반드시 수정해야 합니다. 또한 depth를 이용하여 답변 글의 경우 들여쓰기를 하기로 했으니 이 부분도 수정해야 합니다. 마지막으로 날짜와 시간을 int형으로 변경하였으니 이를 날짜 형식으로 출력하는 부분을 만들어야 합니다.

이외에는 모두 기존 게시판의 소스와 동일하니 실제로 수정되는 부분은 몇 줄밖에 안 된다는 것을 알 수 있습니다. 수정되는 부분을 하나씩 살펴보도록 합시다.

목록 출력을 위한 쿼리문 수정

기존의 쿼리문과 새로운 쿼리문을 비교해보면 다음과 같습니다.

기존 쿼리문	SELECT * FROM board ORDER BY id DESC LIMIT $no,$page_size
새로운 쿼리문	SELECT * FROM $board ORDER BY thread DESC LIMIT $no,$page_size

[표 12-18] 수정된 쿼리

기존 쿼리문과 차이는 단 하나, 정렬의 기준이 되는 항목입니다. 앞서도 언급하였듯이 기존에는 id가 순서를 의미하기 때문에 이를 기준으로 정렬했으나 계층형 게시판에서는 글이 입력된 순서일 뿐 출력되는 순서와는 무관합니다. 실제로 id 대신에 thread라는 새로 추가된 항목을 통해서 정렬합니다. 따라서 id 대신에 thread라고 바꿔주는 것만으로 쿼리문의 수정은 끝납니다.

목록 출력할 때 답변 글 들여쓰기

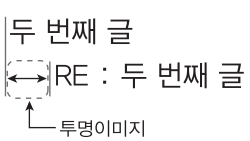
일반적으로 답변 글은 일반 글에 비해서 약간의 공백을 두어 구분하고 있습니다. 답변 글에 또 답변이 있는 경우는 기존 답변 글보다 더 들여쓰기를 해서 표시를 합니다. 다음의 예를 보면 금방 이해할 수 있을 것입니다.

번호	깊이	제목	종류
300	0	세 번째 글	새 글
200	0	두 번째 글	새 글
199	1	RE : 두 번째 글	답변 글
198	2	RE : RE : 두 번째 글	답변 글
100	0	첫 번째 글	새 글

[표 12-19] 답변 글의 들여쓰기

위와 같이 답변 글의 깊이가 깊어질수록 들여쓰기의 정도가 커지게 됩니다. 그래서 답변 글의 깊이 값을 이용하여 들여쓰기를 합니다. 들여쓰기를 하는 방법은 빈 공백( )을 이용하는 방법과 투명한 이미지를 이용하는 방법 등 여러 가지가 있는데 여기서는 이미지를 이용한 들여쓰기를 사용하려고 합니다.

이미지를 이용한 들여쓰기 방법은 이미지의 가로(width) 길이를 조절하는 방법으로 가능합니다. 예를 들어 깊이가 1인 경우 투명한 이미지의 가로 길이를 5, 2인 경우 가로 길이를 10으로 바꾸는 방법입니다. 이 방법은 공백을 넣는 방법에 비해 편리한데 그 이유는 공백을 넣으려면 for 문을 이용하여 깊이만큼 공백을 출력해야 하기 때문입니다.



[그림 12-6] 투명 이미지

이와 같이 들여쓰기를 하려면 당연히 제목 출력 부분의 소스를 수정해야 합니다. 기존에는 제목을 그대로 보여주었던 것을 답변 글인 경우에 투명 이미지를 넣게끔 수정하는 것입니다.

일반 글은 다음과 같이 기존 게시판에서 사용하는 것과 동일하게 출력하면 됩니다.

```
<a href=read.php?id=<?=$row[id]?>&no=<?=$no?>>
<?=strip_tags($row[title]);?></a>
```

만약 답변 글이라면 윗부분에 투명 이미지를 추가하는 부분을 작성하면 될 것입니다. 그러기에 앞서 답변 글의 깊이에 따라 길이가 변하는 투명 이미지를 우선 만들어야 합니다. 이미지의 길이는

width로 조절할 수 있으니 다음과 같은 방법으로 투명 이미지를 만들 수 있습니다.

```
<img height=1 width="<?=$row[depth]*7 ?>">
```

이렇게 되면 깊이에 따라 이미지의 길이가 7씩 늘어나게 되어 깊이에 따른 들여쓰기가 가능합니다. 이를 실제 코드에 적용하면 다음과 같습니다.

```
<? if ($row[depth] > 0) echo "<img height=1 width=" . $row[depth]*7 .
">└";?>
<a href=read.php?id=<?=$row[id]?>&no=<?=$no?>>
<?=strip_tags($row[title]);?></a>
```

이렇게 하면 다음과 같이 답변 글의 깊이에 따라 공백이 생기고 답변 글 앞에는 '└' 기호가 붙게 됩니다.

번호	깊이	제목	종류
300	0	세 번째 글	새 글
200	0	두 번째 글	새 글
199	1	└RE : 두 번째 글	답변 글
198	2	└RE : RE : 두 번째 글	답변 글
100	0	첫 번째 글	새 글

[표 12-20] 답변 글의 들여쓰기 결과

날짜 출력

날짜의 형식이 int형으로 바뀌면서 이를 날짜 형식으로 변경해주는 부분을 추가해야 합니다. 이 부분은 앞서 배운 PHP의 날짜 함수를 이용하여 쉽게 구현할 수 있습니다.

```
<?=date("Y-m-d", $row[wdate])?>
```

date() 함수는 기본적으로 현재의 날짜와 시간을 해당 형식에 맞춰 출력해주지만 형식과 더불어 유닉스 타임 스탬프 값을 입력해주면 입력해준 값에 대한 날짜와 시간을 출력합니다. 따라서 wdate 컬럼으로부터 얻은 유닉스 타임 스탬프 값을 날짜 형식으로 출력하면 매우 간단하게 날짜 출력부가 완성됩니다.

수정된 부분을 기존의 list.php 파일에 적용하면 다음과 같습니다.

list.php

```
<?
//데이터 베이스 연결하기
include "db_info.php";

#########################################################################
# LIST 설정
# 1. 한 페이지에 보여질 게시물의 수
$page_size=10;

# 2. 페이지 나누기에 표시될 페이지의 수
$page_list_size = 10;

#########################################################################
//$no 값이 안 넘어 오거나 잘못된(음수) 값이 넘어오는 경우 0으로 처리
if (!$no || $no < 0) $no=0;
#########################################################################

// 데이터베이스에서 페이지의 첫 번째 글($no)부터 $page_size만큼의 글을 가져온다.
$query = "SELECT * FROM $board ORDER BY thread DESC LIMIT $no,$page_size";
$result = mysql_query($query, $conn);

// 총 게시물 수를 구한다.
//count를 통해 구할 수 있는데 count(항목) 과 같은 방법으로 사용한다. * 는 모든 항목을 뜻한다.
//총 해당 항목의 값을 가지는 게시물의 개수가 얼마인가를 묻는 것이다.
//따라서 전체 글 수가 된다. count(id)와 같은 방법도 가능하지만
//이례적으로 count(*)가 조금 빠르다. 일반적으로는 * 가 느리다.
$result_count=mysql_query("SELECT count(*) FROM $board",$conn);
$result_row=mysql_fetch_row($result_count);
$total_row = $result_row[0];

//결과의 첫 번째 열이 count(*) 의 결과다.
#########################################################################
# 총 페이지 계산
if ($total_row <= 0) $total_row = 0; // 총 게시물의 값이 없을 경우 기본값으로 세팅
// 총 게시물에 1을 뺀 뒤 페이지 사이즈로 나누고 소수점 이하를 버린다.

$total_page = floor(($total_row - 1) / $page_size);
# 총 페이지는 총 게시물의 수를 $page_size로 나누면 알 수 있다.
# 총 게시물이 12개(1을 빼서 11이 된다)이고 페이지 사이즈가 10이라면 결과는 1.1이 나올 것이다.
# 1.1라는 페이지 수는 한 페이지를 다 표시하고도 글이 더 남아있다는 뜻이다.
# 따라서 실제의 페이지 수는 2가 된다. 한 페이지는 2개의 글만 표시될 것이다.
# 그러나 내림을 해주는 이유는 페이지 수가 0부터 시작하기 때문이다. 따라서 1은 두 번째 페이지다.
# 총 게시물에 1을 빼주는 이유는 10페이지가 되면 10/10 = 1이기 때문이다.
# 앞에서도 말했지만 1은 2번째 페이지를 뜻한다.
# 그러나 총 게시물이 10개인 경우 한 페이지에 모두 출력되어야 한다.
```

```
46  # 그래서 1을 빼서 10개인 경우 (10-1) / 10 = 0.9 로 한 페이지에 출력한다.
47  # 글이 0개가 있는 경우는 결과가 -1이 되지만 -1은 무시된다.
48  # ( floor는 내림을 하는 수학함수이다.)
49
50  ##########################################################################
51  # 현재 페이지 계산
52  $current_page = floor($no/$page_size);
53  # $no을 통해서 페이지의 첫 번째 글이 몇 번째 글인지 전달된다.
54  # 따라서 페이지 사이즈로 나누면 현재가 몇 번째 페이지인지 알 수 있다.
55  # $no이 10이고 페이지 사이즈가 10이라면 결과는 1이다. 앞서 페이지는 0부터
56  # 시작이라고 했으니 두 번째 페이지임을 나타낸다.
57  # 그렇다면 $no이 11이라면 1.1이 되어 버린다. 11번째 글도 두 번째 페이지에 존재하므로 0.1
58  # 은 무의미하니 버린다.
59  # 그런데 $no이란 값이 $page_size만큼씩 증가되는 값이기 때문에(0,10,20,30과 같은
60  # 등차수열) 내림을 하는 것 또한 무의미하다.
61  # 그러나 내림을 하는 이유는 $no 값에 11과 같은 값이 들어와도 제대로 출력되기를 바라는
62  # 마음에서 해놓은 것이다.
63  ?>
64  <html>
65  <head>
66  <title>계층형 게시판</title>
67  <style>
68  <!--
69  td { font-size : 9pt; }
70  A:link { font : 9pt; color : black; text-decoration : none;
71  font-family: 굴림; font-size : 9pt; }
72  A:visited { text-decoration : none; color : black;
73  font-size : 9pt; }
74  A:hover { text-decoration : underline; color : black;
75  font-size : 9pt;}
76  -->
77  </style>
78  </head>
79  <body topmargin=0 leftmargin=0 text=#464646>
80  <center>
81  <BR>
82  <!-- 게시물 리스트를 보이기 위한 테이블 -->
83  <table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
84  <!-- 리스트 타이틀 부분 -->
85  <tr height=20 bgcolor=#999999>
86    <td width=30 align=center>
87      <font color=white>번호</font>
88    </td>
89    <td width=370 align=center>
90      <font color=white>제 목</font>
91    </td>
92    <td width=50 align=center>
```

```
93      <font color=white>글쓴이</font>
94    </td>
95    <td width=60 align=center>
96      <font color=white>날 짜</font>
97    </td>
98    <td width=40 align=center>
99      <font color=white>조회수</font>
100   </td>
101 </tr>
102 <!-- 리스트 타이틀 끝 -->
103 <!-- 리스트 부분 시작 -->
104 <?
105   while($row=mysql_fetch_array($result))
106   {
107 ?>
108 <!-- 행 시작 -->
109 <tr>
110   <!-- 번호 -->
111   <td height=20 bgcolor=white align=center>
112     <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
113     <?=$row[id]?></a>
114   </td>
115   <!-- 번호 끝 -->
116   <!-- 제목 -->
117   <td height=20 bgcolor=white> 
118     <?
119     if ($row[depth] > 0)
120       echo "<img height=1 width=" . $row[depth]*7 . ">└";
121     ?>
122     <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
123     <?=strip_tags($row[title]);?></a>
124   </td>
125   <!-- 제목 끝 -->
126   <!-- 이름 -->
127   <td align=center height=20 bgcolor=white>
128     <font color=black>
129     <a href="mailto:<?=$row[email]?>"><?=$row[name]?></a>
130     </font>
131   </td>
132   <!-- 이름 끝 -->
133   <!-- 날짜 -->
134   <td align=center height=20 bgcolor=white>
135     <font color=black><?=date("Y-m-d",$row[wdate])?></font>
136   </td>
137   <!-- 날짜 끝 -->
138   <!-- 조회 수 -->
139   <td align=center height=20 bgcolor=white>
```

```
140      <font color=black><?=$row[view]?></font>
141    </td>
142 <!-- 조회 수 끝 -->
143 </tr>
144 <!-- 행 끝 -->
145 <?
146    } // end While
147
148 //데이터베이스와의 연결을 끝는다.
149 mysql_close($conn);
150 ?>
151 </table>
152 <!-- 게시물 리스트를 보이기 위한 테이블 끝-->
153 <!-- 페이지를 표시하기 위한 테이블 -->
154 <table border=0>
155 <tr>
156    <td width=600 height=20 align=center rowspan=4>
157    <font color=gray>
158     
159 <?
160 ##########################################################################
161 # 페이지 리스트
162 # 페이지 리스트의 첫 번째로 표시될 페이지가 몇 번째 페이지인지 구하는 부분이다.
163
164 $start_page = (int)($current_page / $page_list_size)
165    * $page_list_size;
166
167 # 현재 페이지를 페이지 리스트 수로 나누면 현재 페이지가 몇 번째 페이지 리스트에 있는지 알게 된다.
168 # 이 또한 0을 기준으로 하기에 형변환(타입 캐스팅)을 해주었다.
169 # 형변환은 앞 강좌에서 배웠지만 위의 나누어지는 수가 1.2와 같이 유리수로 표시되기 때문에
170 # int(정수)형으로 형변환을 하면 소수점 자리가 사라진다.
171 # 즉, 위에서 사용한 floor랑 같은 효과를 하게 되는 것이다.
172 # 여기에 floor 함수를 취하거나 위의 floor를 형변환해도 상관없다.
173 # 페이지 리스트의 마지막 페이지가 몇 번째 페이지인지 구하는 부분이다.
174
175 $end_page = $start_page + $page_list_size - 1;
176 if ($total_page < $end_page) $end_page = $total_page;
177
178 # 보여주는 페이지 리스트 중에서 마지막 페이지가 되는 경우는 두 가지다.
179 # 1. 페이지가 페이지 리스트 크기보다 더 많이 남아있을 때
180 # 10개씩 뿌려주는데 총 11페이지가 존재한다면 11페이지는 두 번째 목록 페이지에 뿌려진다.
181 # 그렇다면 마지막 페이지 리스트는 10페이지 즉, 첫 번째 페이지 + 9번째 페이지이다.
182 # 2. 10개씩 뿌려주는데 5페이지밖에 없다면?
183 # 마지막 리스트 페이지는 5페이지가 된다.
184
185 ##########################################################################
186 # 이전 페이지 리스트 보여주기
```

```
187 # 페이지 리스트가 10인데 13번째 페이지에 있다면 두 번째 목록 페이지를 보고 있는 것이다.
188 # 이전 목록 페이지로 가고 싶을 때 사용한다.
189 # 이전 페이지 리스트가 필요할 때는 페이지 리스트의 첫 페이지가 페이지 리스트 수보다 클 때다.
190 # 페이지가 적어도 페이지 리스트 수보다는 커야 이전 페이지 리스트가 존재할 테니까 말이다.
191 # 페이지 리스트의 수가 10인데 총 5페이지밖에 없다면 이전 페이지 리스트는 존재하지 않는다.
192
193 if ($start_page >= $page_list_size) {
194
195 # 이전 페이지 리스트값은 첫 번째 페이지 리스트에서 뒤로 리스트의 수만큼 이동하면 된다.
196 # $page_size를 곱해주는 이유는 글 번호로 표시하기 위해서이다.
197
198   $prev_list = ($start_page - 1)*$page_size;
199   echo "<a href=\"$PHP_SELF?no=$prev_list\">◀</a>\n";
200 }
201 # 페이지 리스트를 출력
202 for ($i=$start_page;$i <= $end_page;$i++) {
203   $page=$page_size*$i; // 페이지값을 no 값으로 변환.
204   $page_num = $i+1; // 실제 페이지 값이 0부터 시작하므로 표시할 때는 1을 더해준다.
205
206   if ($no!=$page){ //현재 페이지가 아닐 경우만 링크를 표시
207     echo "<a href=\"$PHP_SELF?no=$page\">";
208   }
209
210   echo " $page_num "; //페이지를 표시
211
212   if ($no!=$page){
213     echo "</a>";
214   }
215 }
216 # 다음 페이지 리스트가 필요할 때는 총 페이지가 마지막 리스트보다 클 때이다.
217 # 리스트를 다 뿌리고도 더 뿌려줄 페이지가 남았을 때 다음 버튼이 필요할 것이다.
218 if($total_page > $end_page)
219 {
220   # 다음 페이지 리스트는 마지막 리스트 페이지보다 한 페이지 뒤로 이동하면 된다.
221   $next_list = ($end_page + 1)* $page_size;
222   echo "<a href=$PHP_SELF?no=$next_list>▶</a><p>";
223 }
224 ?>
225 </font>
226 </td>
227 </tr>
228 </table>
229 <a href=write.php>글쓰기</a>
230 </center>
231 </body>
232 </html>
```

목록 테스트

계층형 게시판의 목록 보기 페이지를 모두 구현했습니다. 이제 목록이 제대로 잘 보이는지 확인을 해 봅시다. 목록을 구현하기 전에 먼저 글을 입력하는 부분을 구현했는데요, 지금 만든 글 목록을 통해 그 글이 아직 잘 저장되어 있는지 확인해 보도록 합시다.

[그림 12-7] 글 목록 보기 테스트

아직까지 온전히 잘 살아 있군요. 만족스럽습니다. 하지만 아직 답변 글이 잘 보이는지 확인을 하지 못했기 때문에 답변 달기 부분을 완성한 후에 다시 한번 살펴보도록 하겠습니다.

Section 07

글 읽기 – read.php

글 읽기 페이지는 기존 게시판의 모양과 약간 다르게 꾸며보았습니다. 일단 어떻게 바뀌었는지 그림을 통해 살펴봅시다.

[그림 12-8] 글 읽기 페이지

전체적으로는 거의 비슷한 틀을 가지고 있지만 두 가지가 크게 변했습니다.

① 이전, 다음 글을 윗글, 아랫글로 변경하고 제목과 글쓴이를 같이 보여준다.

② 하단에 이 글과 연관이 있는 글의 목록을 보여준다.

기존 게시판에서는 이전, 다음 글로 표시되어 있어서 많은 사람이 이에 대해 혼동을 하는 경우가 많았습니다. 그래서 이전, 다음 대신에 윗글, 아랫글로 표시하여 목록에서 위에 존재하는 글인지 아니면 아래에 존재하는 글인지로 표시하여 혼동을 줄였습니다. 또한 이를 보여줌에 있어서 기존에는 [이전], [다음] 이라고만 표시되어 있어서 단순히 순서대로 읽어 나갈 때 이외에는 거의 사용을 하지 않았습니다. 그래서 위와 같이 글의 제목과 글쓴이 정보를 같이 보여주어 보다 유용한 기능으로 변모시켰습니다.

그런데 윗글, 아랫글 부분을 제작하면서 한 가지 고민에 빠졌습니다. 기존 게시판은 윗글과 아랫글의 기준이 명확해서 그저 찾기만 하면 되었는데, 계층형 게시판이 되면서 답변 글이란 것이 생겨서 이들의 기준이 모호해졌습니다. 만약 해당 게시물에 답변 글이 있다면 아랫글은 답변 글이어야 하나, 아니면 답변 글은 무시하고 다음번 일반 글이어야 하나를 고민하게 된 것입니다. 여러분은 어떻게 생각하세요? 그래서 고민 끝에 답변 글을 무시하고 일반 글만 보여주기로 마음을 먹었습니다. 그런데 문제가 생겨버렸습니다. 일반 글만 보여주면 답변 글은 못 읽고 넘어가는 경우가 일어난다는 것입니다. 어떻게 해야 하나 고민하다가 결국 위와 같이 페이지 하단에 이 글과 연관있는 모든 글을 목록으로 보여주기로 결정했습니다.

윗글, 아랫글 구현

기존 게시판에서는 이전, 다음 글을 단순히 찾아서 링크만 연결했었지만 보다 유용하게 사용할 수 있도록 제목과 글쓴이 정보를 같이 보여주고자 합니다.

우선 윗글과 아랫글에 대해 정의를 해봅시다.

버 튼	정의	링크
윗글	현재 글을 기준으로 목록 위에 있는 글 중 가장 가까이에 있는 글. 단, 답변 글은 제외	read.php?id=윗글 번호
아랫글	현재 글을 기준으로 목록 아래에 있는 글 중 가장 가까이에 있는 글. 단, 답변 글은 제외	read.php?id=아랫글 번호

[표 12-21] 윗글과 아랫글의 정의

정의를 하였으니 이제 직접 구현을 해볼텐데요, 기존에 작성한 소스 코드를 가져다가 약간 수정을 해야 합니다.

기존 게시판에서는 다음과 같은 쿼리를 사용하여 윗글을 찾았습니다.

```
SELECT id FROM board WHERE id > $id LIMIT 1
```

즉, 현재 글의 id 값보다 큰 것 중에서 가장 작은 값을 찾았습니다. 그러나 계층형 게시판에서는 글의 번호가 가지는 의미가 단순히 작성된 순서와 글을 구분 짓는 번호의 의미이어서 실제 정렬에 사용하는 thread 값을 통해서 윗글을 찾아야 합니다. 따라서 위의 쿼리를 thread 값으로 변경하면 다음과 같습니다.

```
SELECT id FROM $board WHERE thread > $row[thread] LIMIT 1
```

계층형 게시판으로 변경되었으니 테이블 이름을 변경하고 id로 검색하던 것을 thread로 바꾸었습니다. 그런데 여기서 주의할 점은 id를 thread로 바꾼다고 해서 SELECT thread FROM $board와 같이 검색하면 안 된다는 것입니다. 왜냐하면 내부적으로는 thread 값이 정렬에 사용되고 글을 구분하기도 하지만 외부적으로는 id 값이 글을 구분 짓는 키(Primary Key)이기 때문입니다. 따라서 이 쿼리를 통해서 알아내야 하는 값은 윗글의 글 번호이므로 반드시 SELECT id FROM $board와 같이 쿼리문을 작성해야 합니다.

그런데 앞서 윗글과 아랫글을 정의하면서 답변 글은 제외한다고 했습니다. 현재 작성된 쿼리문은 단순히 thread 값이 큰 것을 찾기 때문에 답변 글이 검색될 가능성이 있습니다. 따라서 답변 글을 제외시키는 부분을 추가해야 하는데, 어떻게 할 수 있을까요?

답변 글인지 아닌지 판단하는 방법은 두 가지가 있습니다.

① thread 값이 1000의 배수가 아닌 글은 답변 글이다.

② depth 값이 0이 아닌 글은 답변 글이다.

이 두 가지 중 하나를 선택해서 구분하면 되는데, 딱 보기에도 depth 값을 이용하는 것이 쉽고 편할 것 같군요. depth 값을 통해 쿼리문을 수정해 봅시다.

```
SELECT id FROM $board WHERE thread > $row[thread] and depth = 0 LIMIT 1
```

depth는 일반 글이면 반드시 0이고 답변 글이면 1, 답변 글에 대한 답변이면 2와 같은 식으로 증가하는 값입니다. 따라서 위와 같이 쿼리문을 작성하면 답변 글을 제외한 일반 글들만 검색됩니다.

이와 마찬가지로 아랫글을 구현해 봅시다. 기존에는 아랫글을 다음과 같은 쿼리를 이용하여 찾아냈습니다.

```
SELECT id FROM testboard WHERE id < $id ORDER BY id DESC LIMIT 1
```

이를 계층형 게시판용으로 바꾸어 봅시다. 앞서 윗글을 구현할 때와 마찬가지로 id를 thread로 변

경하면 됩니다.

```
SELECT id FROM $board WHERE thread < $row[thread] and depth=0 ORDER
BY thread DESC LIMIT 1
```

여기서 depth=0을 추가하여 답변 글을 제외하는 것도 잊지 않도록 합니다.

이제 윗글과 아랫글을 위한 기본적인 쿼리문을 완성했습니다. 그런데 계층형 게시판에서는 제목과 글쓴이 정보를 보여준다고 했으니 검색하는 컬럼을 더 추가해야겠군요. 제목과 글쓴이 정보까지 추가하여 최종적으로 작성된 쿼리문은 다음과 같습니다.

버튼	SQL 쿼리
윗글	SELECT id,name,title FROM $board WHERE thread 〉 $row[thread] and depth = 0 LIMIT 1
아랫글	SELECT id,name,title FROM $board WHERE thread 〈 $row[thread] and depth=0 ORDER BY thread DESC LIMIT 1

[표 12-22] 윗글과 아랫글의 쿼리

이제 윗글과 아랫글에 대한 쿼리문을 작성하였으니 이를 이용하여 실제로 출력하는 부분을 구현해 봅시다. 기존 게시판에는 [이전], [다음] 버튼이 [글목록] 버튼과 같은 행에 존재했었는데 제목과 이름 정보를 보여주기에는 공간이 부족하여 바로 밑에 새로 행을 추가하여 출력합니다.

```
$query=mysql_query("SELECT id, name, title FROM $board
WHERE thread > $row[thread] and depth=0 LIMIT 1", $conn);
$up_id=mysql_fetch_array($query);

$query=mysql_query("SELECT id, name, title FROM $board
WHERE thread < $row[thread] and depth=0 ORDER BY thread DESC LIMIT 1",
$conn);
$down_id =mysql_fetch_array($query);

if ($up_id[id]) // 윗글이 있을 경우
{
  //윗글의 제목($up_id[title])과 글쓴이($up_id[name]) 정보를 출력
}
if ($down_id[id]) // 아랫글이 있을 경우
{
  //아랫글의 제목($down_id[title])과 글쓴이($down_id[name]) 정보를 출력
}
```

연관된 글 목록 구현

연관된 글 목록은 해당 글의 답변 글이나 자신이 답변 글인 경우 그 부모글이 되는 일반 글의 목록을 보여주는 것입니다. 연관된 글 목록은 기본적으로 글 목록인 list.php 소스를 이용하며 차이는 해당 글과 관련이 있는 글만 보여준다는 것입니다. 예를 들면 아래에서 보는 것과 같이 2000, 1999, 1998번의 내부 글 번호를 갖는 세 글을 읽는 경우 연관 글 목록에 이 세 글이 모두 목록화되어 나오게 하는 것이 목적입니다.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1999	1	└RE : 두 번째 글	답변 글
1998	2	└RE : RE : 두 번째 글	답변 글
1000	0	첫 번째 글	새 글

[표 12-23] 연관된 글의 목록

그렇다면 문제는 어떻게 관련된 글을 찾아서 목록으로 출력하느냐 하는 것입니다. 2000번의 관련 글은 어떻게 찾을 수 있을까요? 또한 1999번의 관련 글은 어떻게 찾을 수 있을까요?

현재 우리가 사용하는 알고리즘을 잘 생각해보면 그 답을 알 수 있습니다. 언제나 일반 글은 1000의 배수인 번호를 갖고 답변 글은 일반 글 번호에서 1씩 줄어드는 번호를 갖게 됩니다. 그렇기 때문에 일반 글의 번호가 2000번이고 답변 글이 여러 개 달려있다고 하면 그 답변 글이 몇 개가 있든, 1000보다 큰 번호를 가지게 될 것입니다. 즉, 1001 ~ 2000 사이에 존재하는 글은 모두 연관이 있는 글이 된다는 뜻입니다.

그러면 실제로 한번 찾아봅시다.

일반 글에서 관련 글 찾기

현재 2000번의 글을 읽고 있다고 가정하면 연관된 글은 자신인 2000번을 포함하여 1000을 뺀 1000보다 큰 번호를 갖는 모든 글입니다. 따라서 다음과 같은 수식을 얻을 수 있습니다.

일반 글의 thread 값 ≥ 관련 글 〉 일반 글의 thread 값 − 1000

수식이 맞는지 검증해보면 현재 글의 내부 글 번호가 5000이라고 할 때

5000 ≥ 관련 글 〉 4000

위와 같은 결과를 얻습니다. 앞서 말씀드린 대로 올바른 결과를 얻을 수 있군요.

답변 글에서 관련 글 찾기

일반 글에서 관련 글을 찾는 경우는 아주 간단하게 자신의 내부 글 번호에서 1000을 뺀 값보다 큰 범위에 있는 글로 쉽게 찾을 수 있었습니다. 그러나 답변 글인 경우에는 그렇게 하면 안 되는 것을 알 수 있습니다.

현재 읽고 있는 글의 내부 글 번호가 1998이라면 1000을 빼면 998이 되기 때문에 999와 1000이 관련 글의 범위 내에 속하게 됩니다. 그래서 위의 식을 답변 글에는 사용할 수 없습니다. 그래서 답변 글은 먼저 일반 글의 번호를 찾아야 합니다. 즉, 위의 식에 앞서서 일반 글의 번호를 찾는 부분이 더 추가된 셈입니다. 이 방법은 이미 앞에서도 한 번 사용한 적이 있는 것으로 소수점을 이용하여 해결할 수 있습니다.

예를 들어 1998의 경우 2000이 일반 글의 번호가 됩니다. 소수점을 이용하는 방법을 사용하면 1998을 1000으로 나누면 1.998입니다. 1.998을 올림하면 2라는 값을 얻고 다시 조금 전에 나누었던 1000을 곱하면 2000이라는 값을 얻을 수 있습니다. 구해진 일반 글의 번호를 이용해 위에서 사용한 식을 다시 사용하면 관련 글의 범위를 알 수 있습니다.

따라서 일반 글과 답변 글에서 관련 글을 찾는 방법을 정리해보면 다음과 같습니다.

```
$thread_end = ceil($row[thread]/1000)*1000;
$thread_start = $thread_end - 1000;
```

구해진 값을 통해서 쿼리문을 작성해 봅시다. 앞서 목록 페이지에서 사용한 쿼리는 다음과 같습니다.

```
SELECT * FROM $board ORDER BY thread DESC LIMIT $no,$page_size
```

이는 전체 글 중에서 $page_size만큼의 글을 가져오는 쿼리문입니다. LIMIT를 사용하는 이유는 검색된 글의 숫자가 많은 경우에 몇 개씩 묶음으로 나누어 보여주기 위한 것인데 관련 글 목록은 관련된 글 모두를 보여주기 때문에 LIMIT를 사용하지 않아도 됩니다. 따라서 여기에 글의 범위 제약만 추가해주면 쉽게 쿼리문을 완성할 수 있습니다.

```
SELECT * FROM $board
WHERE thread > $thread_start and thread <= $thread_end ORDER BY thread
DESC;
```

다시 한번 쿼리를 해석해보면 "$thread_start보다 크고 $thread_end 이하인 글을 검색하라"는 의미가 됩니다. 글 목록 페이지에서 페이징을 제외한 목록을 출력하는 부분을 가져와서 쿼리문을 위의 것으로 변경하면 아주 쉽게 관련 글 목록이 완성됩니다.

```
<?
  //리스트 출력을 위해 thread를 계산한다.
  $thread_start = (ceil($row[thread]/1000)-1)*1000;
  $thread_end = $thread_start  - 1000;
```

```
    $query = "SELECT * FROM $board WHERE thread <= $thread_end and
    thread > $thread_start ORDER BY thread DESC";
    $result = mysql_query($query, $conn);

    while($row=mysql_fetch_array($result))
    {
        //목록을 출력
    }
?>
```

완성된 소스

윗글 아랫글 구현과 관련 글 목록 구현 이외에 날짜 시간 형식이 int형으로 변경되면서 이 페이지에도 날짜 형식으로 출력하는 코드를 추가해야 합니다. 날짜 형식 변환 부분을 추가하면 다음과 같은 완성된 소스 코드를 얻을 수 있습니다.

read.php

```
<?
//데이터 베이스 연결하기
include "db_info.php";

// 조회수 업데이트
$query = "UPDATE $board SET view=view+1 WHERE id=$_GET[id]";
$result=mysql_query($query, $conn);

// 글 정보 가져오기
$query = "SELECT * FROM $board WHERE id=$_GET[id]";
$result=mysql_query(, $conn);
$row=mysql_fetch_array($result);
?>
<html>
<head>
<title>계층형 게시판</title>
<style>
<!--
td { font-size : 9pt; }
A:link { font : 9pt; color : black; text-decoration : none;
font-family: 굴림; font-size : 9pt; }
A:visited { text-decoration : none; color : black;
font-size : 9pt; }
A:hover { text-decoration : underline; color : black;
font-size : 9pt;}
```

```
-->
</style>
</head>
<body topmargin=0 leftmargin=0 text=#464646>
<center>
<BR>
<table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
<tr>
  <td height=20 colspan=4 align=center bgcolor=#999999>
    <font color=white><B><?=strip_tags($row[title]);?>
    </B></font>
  </td>
</tr>
<tr>
  <td width=50 height=20 align=center bgcolor=#EEEEEE>글쓴이</td>
  <td  width=240 bgcolor=white><?=$row[name]?></td>
  <td width=50 height=20 align=center bgcolor=#EEEEEE>이메일</td>
  <td  width=240 bgcolor=white><?=$row[email]?></td>
</tr>
<tr>
  <td width=50 height=20 align=center bgcolor=#EEEEEE>
    날   짜</td><td width=240 bgcolor=white>
    <?=date("Y-m-d", $row[wdate])?></td>
  <td width=50 height=20 align=center bgcolor=#EEEEEE>조회수</td>
  <td  width=240 bgcolor=white><?=$row[view]?></td>
</tr>
<tr>
  <td bgcolor=white colspan=4 style="word-break:break-all;">
    <font color=black>
    <pre><?=strip_tags($row[content]);?></pre>
    </font>
  </td>
</tr>
<!-- 기타 버튼 들 -->
<tr>
  <td colspan=4 bgcolor=#999999>
  <table width=100%>
  <tr>
    <td width=280 align=left height=20>
      <a href=list.php?no=<?=$no?>><font color=white>
      [목록보기]</font></a>
      <a href=reply.php?id=<?=$id?>><font color=white>
      [답글달기]</font></a>
      <a href=write.php><font color=white>
      [글쓰기]</font></a>
      <a href=edit.php?id=<?=$id?>><font color=white>
      [수정]</font></a>
```

```
73          <a href=predel.php?id=<?=$id?>><font color=white>
74          [삭제]</font></a>
75        </td>
76     </tr>
77     </table>
78     </td>
79  </tr>
80  </table>
81
82  <table width=580 bgcolor=white style="border-bottom-width:1;
83  border-bottom-style:solid;border-bottom-color:cccccc;">
84  <?
85  // 현재 글보다 thread 값이 큰 글 중 가장 작은 것의 id를 가져온다.
86  $query = "SELECT id, name, title FROM $board
87  WHERE thread > $row[thread] LIMIT 1";
88  $query=mysql_query($query, $conn);
89  $up_id=mysql_fetch_array($query);
90
91  if ($up_id[id]) // 이전 글이 있을 경우
92  {
93    echo "<tr><td width=500 align=left height=25>";
94    echo "<a href=read.php?id=$up_id[id]>△ $up_id[title]</a></td>
95    <td align=right>$up_id[name]</td></tr>";
96  }
97
98  // 현재 글보다 thread 값이 작은 글 중 가장 큰 것의 id를 가져온다.
99  $query = "SELECT id, name, title FROM $board WHERE
100 thread < $row[thread] ORDER BY thread DESC LIMIT 1";
101 $query=mysql_query(, $conn);
102 $down_id=mysql_fetch_array($query);
103
104 if ($down_id[id])
105 {
106   echo "<tr><td width=500 align=left height=25>";
107   echo "<a href=read.php?id=$down_id[id]>▽ $down_id[title]</a>
108   </td><td align=right>$down_id[name]</td></tr>";
109 }
110 ?>
111 </table>
112 <BR>
113 <?
114 //리스트 출력을 위해 thread를 계산한다.
115 //출력될 리스트는 글 전체 리스트가 아니라
116 //1000의 배수인 새 글과 이를 포함한 답변 글들의 리스트이다.
117 //답변 글이 없는 경우 원본 글만 리스트에 나오고
118 //답변 글이 있으면 답변 글 모두가 다 나오게 된다.
119 //현재 글이 답변 글이어도 새 글부터 전체 답변 글까지 나온다.
```

```
120 //그럴려면 1000~2000과 같이 새 글 사이에 글들을 모두 뿌려주면 된다.
121
122 $thread_end = ceil($row[thread]/1000)*1000;
123 $thread_start = $thread_end - 1000;
124
125 $query = "SELECT * FROM $board WHERE thread <= $thread_end and
126 thread > $thread_start ORDER BY thread DESC";
127 $result = mysql_query($query, $conn);
128 ?>
129 <!-- 게시물 리스트를 보이기 위한 테이블 -->
130 <table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
131 <!-- 리스트 타이틀 부분 -->
132 <tr height=20 bgcolor=#999999>
133   <td width=30 align=center>
134     <font color=white>번호</font>
135   </td>
136   <td width=370 align=center>
137     <font color=white>제 목</font>
138   </td>
139   <td width=50 align=center>
140     <font color=white>글쓴이</font>
141   </td>
142   <td width=60 align=center>
143     <font color=white>날 짜</font>
144   </td>
145   <td width=40 align=center>
146     <font color=white>조회수</font>
147   </td>
148 </tr>
149 <!-- 리스트 타이틀 끝 -->
150 <!-- 리스트 부분 시작 -->
151 <?
152   while($row=mysql_fetch_array($result))
153   {
154 ?>
155 <!-- 행 시작 -->
156 <tr>
157 <!-- 번호 -->
158   <td height=20 bgcolor=white align=center>
159     <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
160     <?=$row[id]?></a>
161   </td>
162   <!-- 번호 끝 -->
163   <!-- 제목 -->
164   <td height=20 bgcolor=white> 
165     <? //depth 값을 통해서 들여쓰기를 한다. 투명 이미지의 가로 사이즈를 늘이는 방법
166     if ($row[depth] > 0)
```

```
167         echo "<img src=img/dot.gif height=1 width=" .
168         $row[depth]*7 . ">->";
169       ?>
170       <a href="read.php?id=<?=$row[id]?>&no=<?=$no?>">
171       <?=strip_tags($row[title], '<b><i>');?></a>
172     </td>
173     <!-- 제목 끝 -->
174     <!-- 이름 -->
175     <td align=center height=20 bgcolor=white>
176       <font color=black>
177       <a href="mailto:<?=$row[email]?>"><?=$row[name]?></a>
178       </font>
179     </td>
180     <!-- 이름 끝 -->
181     <!-- 날짜 -->
182     <td align=center height=20 bgcolor=white>
183       <font color=black><?=date("Y-m-d",$row[wdate])?></font>
184     </td>
185     <!-- 날짜 끝 -->
186     <!-- 조회 수 -->
187     <td align=center height=20 bgcolor=white>
188       <font color=black><?=$row[view]?></font>
189     </td>
190 <!-- 조회 수 끝 -->
191 </tr>
192 <!-- 행 끝 -->
193 <?
194     } // end While
195 mysql_close($conn);
196 ?>
197 </center>
198 </body>
199 </html>
```

글 읽기 테스트

완성된 글 읽기 페이지가 제대로 동작하는지 확인을 해보도록 하겠습니다. 지금까지 글 하나밖에 입력하지 않아서 윗글 아랫글 테스트를 위해 글 두 개를 더 등록했습니다.

[그림 12-9] 글 읽기 테스트

역시 잘 동작하는군요. 조금씩 게시판의 모습을 찾아가는 것을 보니 뿌듯합니다. 마저 힘을 내서 다 같이 파이팅!

Section 08

글 수정, 삭제 – edit, update, predel, del.php

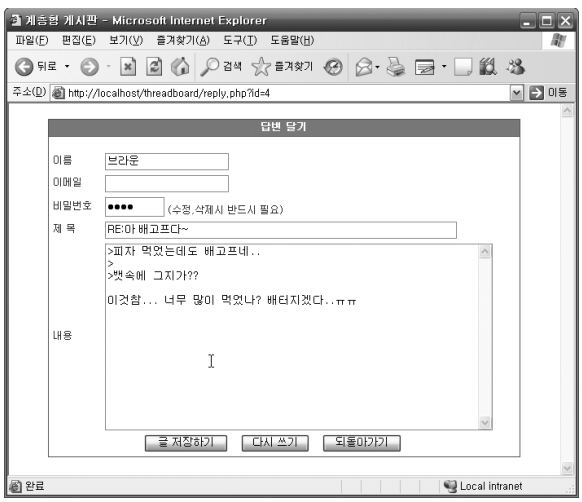
글 수정과 삭제는 기존의 소스를 그대로 이용할 수 있습니다. 대신에 몇 가지를 수정해야 합니다. 수정해야 할 부분은 다음과 같습니다.

- $id → $_GET[id]
- board → $board(게시판 Table 이름)
- name, title, email, content → $_POST[name]처럼 변경.

너무 좋으신가요? 이제 남은 것은 답변 글을 입력할 수 있게 하는 것뿐입니다. 앞서 거의 수정을 하지 않고 그대로 사용할 수 있다고 해놓고 막상 설명을 시작하고 보니 생각보다 많은 사항을 수정한 것 같습니다. 잠깐 쉬어간다 생각하고 한 10~20분쯤 자리에서 일어나서 잠깐 바람을 쐬고 오기 바랍니다. 만약 지금 지하철에서 이 책을 보고 있다면 자리에서 일어나지는 마세요. ^^ 자리 뺏깁니다.

Section 09

답글 달기 – reply.php



[그림 12-10] 답변 달기 페이지

이제 계층형 게시판의 마지막 부분에 도달했습니다. 답글 달기의 경우 글 쓰기 부분의 소스를 거의 대부분 이용하고 약간의 소스 코드만 추가해주면 됩니다. 그러면 우선 일반 글 쓰기와 답변 달기가 어떤 것이 다른지 알아봅시다.

일단 일반 글 쓰기와 답변 달기에서 가장 큰 차이는 답변의 경우 반드시 부모 글이 존재한다는 것입니다. 부모 글을 기준으로 그에 해당하는 답변을 덧붙이는 것입니다. 그래서 일반적으로 답변 글을 작성하고자 할 때 부모 글의 제목과 내용을 입력 폼에 미리 출력을 해놓습니다. 따라서 일반 글 쓰기와는 달리 데이터베이스에서 부모 글의 정보를 가져와 입력 폼에 출력해주는 부분을 작성해야 합니다.

부모 글의 정보 출력

부모 글의 정보를 가져오기 위한 쿼리를 작성해 봅시다. 부모 글의 정보는 글 읽기 페이지에서 구현한 것과 동일하게 작성하면 됩니다.

```
SELECT * FROM $board WHERE id=$_GET[id]
```

id 값은 read.php 파일로부터 부모 글의 id 값을 전달받습니다. 이를 이용하여 read.php 파일에서 글의 정보를 얻는 것과 마찬가지로 부모 글의 정보를 얻을 수 있습니다. 쿼리로부터 글의 정보를 가

져오는 소스 코드는 다음과 같습니다.

```
include "db_info.php";
$parent_result = mysql_query("SELECT * FROM $board
WHERE id='$_GET[id]'", $conn);
$parent_row = mysql_fetch_array($parent_result);
```

하지만 이렇게 가져온 부모 글의 정보를 폼에다가 출력할 때 그대로 출력하면 안 됩니다. 그 이유는 답변으로 작성한 글과 실제 부모 글의 내용이 구분이 안 되기 때문입니다. 그래서 위의 그림과 같이 제목 앞에는 "RE:"와 같은 기호를 추가하고 부모 글의 본문 내용 앞에는 "〉" 표시와 같은 기호를 추가하여 부모 글과 답변 글의 내용을 구분할 수 있게 만듭니다.

이처럼 부모 글의 내용을 구분하기 위해 다음과 같이 합니다.

```
//제목에 RE:를 추가한다.
$parent_title = "RE:" . $parent_row[title];

//아래 부분은 원본 내용에 > 표시를 붙이는 부분이다.
$parent_content = "\n>" . str_replace("\n", "\n>",
$parent_row[content]);
```

여기서 str_replace 함수는 문자열을 치환하는 함수입니다. str_replace(A, B, C); 와 같은 방법으로 사용하며 C 안에 있는 문자 중에 A를 찾아서 B로 바꾸는 역할을 합니다.

처리한 부모 글을 다음과 같이 폼에 출력합니다.

```
<INPUT type=text name=title size=60 maxlength=35
value="RE:<?=$parent_title?>">
<TEXTAREA name=content cols=65 rows=15><?=$parent_content?></TEXTAREA>
```

부모 글의 정보 전달

부모 글의 정보를 폼에 출력하는 것으로 끝이 아닙니다. 부모 글에 대한 정보를 insert_reply.php 파일에 전달해야 합니다. 왜 이런 행동을 해야 하냐면 답변 글의 정보를 데이터베이스에 저장할 때 답변 글의 번호가 될 새로운 thread 값과 답변 글의 깊이를 나타내는 depth 값을 입력해야 하는데 이 두 값이 부모 글의 thread 값과 depth 값을 기준으로 계산되기 때문입니다. 따라서 다음과 같이 숨겨진 폼을 이용하여 insert_reply.php 파일에 부모 글의 정보를 전달합니다.

```
<form action=insert_reply.php method=post>
<input type=hidden name=parent_depth value=<?=$parent_row[depth]?>>
<input type=hidden name=parent_thread value=<?=$parent_row[thread]?>>
```

수정된 소스는 다음과 같습니다.

reply.php

```
<?
include "db_info.php";
$query = "SELECT * FROM $board WHERE id='$_GET[id]'";
$parent_result = mysql_query($query, $conn);
$parent_row = mysql_fetch_array($parent_result);
$parent_title = "RE:" . $parent_row[title];
$parent_content = "\n>" . str_replace("\n", "\n>",
$parent_row[content]);
?>
<html>
<head>
<title>계층형 게시판</title>
<style>
<!--
td { font-size : 9pt; }
A:link { font : 9pt; color : black; text-decoration : none;
font-family: 굴림; font-size : 9pt; }
A:visited { text-decoration : none; color : black;
font-size : 9pt; }
A:hover { text-decoration : underline; color : black;
font-size : 9pt;}
-->
</style>
</head>
<body topmargin=0 leftmargin=0 text=#464646>
<center>
<BR>
<!-- 입력된 값을 다음 페이지로 넘기기 위해 FORM을 만든다. -->
<form action=insert_reply.php method=post>
<input type=hidden name=parent_depth value=<?=$parent_row[depth]?>>
<input type=hidden name=parent_thread value=<?=$parent_row[thread]?>>
<table width=580 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777>
<tr>
  <td height=20 align=center bgcolor=#999999>
    <font color=white><B>답변 달기</B></font>
  </td>
</tr>
<!-- 입력 부분 -->
<tr>
  <td bgcolor=white> 
  <table>
  <tr>
    <td width=60 align=left >이름</td>
    <td align=left >
      <INPUT type=text name=name size=20 maxlength=10>
    </td>
```

```
47   </tr>
48   <tr>
49     <td width=60 align=left >이메일</td>
50     <td align=left >
51       <INPUT type=text name=email size=20 maxlength=25>
52     </td>
53   </tr>
54   <tr>
55     <td width=60 align=left >비밀번호</td>
56     <td align=left >
57       <INPUT type=password name=pass size=8 maxlength=8>
58       (수정,삭제 시 반드시 필요)
59     </td>
60   </tr>
61   <tr>
62     <td width=60 align=left >제 목</td>
63     <td align=left >
64       <INPUT type=text name=title size=60 maxlength=35
65       value="<?=$parent_title?>">
66     </td>
67   </tr>
68   <tr>
69     <td width=60 align=left >내용</td>
70     <td align=left >
71       <TEXTAREA name=content cols=65 rows=15>
72       <?=$parent_content?></TEXTAREA>
73     </td>
74   </tr>
75   <tr>
76     <td colspan=10 align=center>
77       <INPUT type=submit value="글 저장하기">
78         
79       <INPUT type=reset value="다시 쓰기">
80         
81       <INPUT type=button value="되돌아가기"
82       onclick="history.back(-1)">
83     </td>
84   </tr>
85   </TABLE>
86   </td>
87 </tr>
88 <!-- 입력 부분 끝 -->
89 </table>
90 </center>
91 </body>
92 </html>
```

Section 10

답글 저장 – insert_reply.php

계층형 게시판의 마지막이면서 핵심이라고 할 수 있는 실제 계층형 게시판 알고리즘을 적용하는 부분이 바로 답변 글을 저장하는 페이지입니다. 앞서 약간의 설명으로 반업데이트형 게시판이 어떠한 것인지 또한 어떠한 알고리즘으로 동작하는지를 알려 드렸습니다. 실제 구현을 위해서 정확히 어떻게 동작하는지 알아보도록 하겠습니다.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1000	0	첫 번째 글	새 글

[표 12-24] 게시물 초기 상태

처음에 세 개의 새 글이 등록되었습니다. 이제 두 번째 글에 답변을 달아 보겠습니다.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1999	1	ㄴ두 번째 글의 답변	답변 글
1000	0	첫 번째 글	새 글

[표 12-25] 두 번째 글의 답변 글 추가

두 번째 글에 답변이 달리면 위와 같이 바뀔 것입니다. 2000번보다는 작고 1000번보다는 큰 숫자를 글 번호로 해야 번호로 정렬했을 때 위처럼 보일 것입니다. 그래서 2000보다 하나 작은 1999번으로 답변 글을 등록합니다.

다시 2000번의 글에 답변을 하나 더 달아 보겠습니다. 그런데 고민이 생겼습니다. 두 번째 답변 글을 등록하는 방법이 두 가지가 있기 때문입니다. 하나는 첫 번째 답변 글 아래에 등록하는 것이고 또 하나는 반대로 그 위에 등록하는 것입니다. 즉, 1998번으로 답변 등록할 것인지 1999번으로 등록할 것인지가 문제가 됩니다. 만약 1999번으로 등록하려면 기존의 1999번을 1998번으로 밀어내고 그 자리에 등록을 해야 할 겁니다.

단순히 생각하면 밀어내는 방법보다 바로 1998번으로 등록하는 것이 편해 보입니다. 하지만 실제로는 답변이 여러 개가 있을 경우 답변 글에 다시 답변을 등록하면 이 방법을 사용하더라도 어차피 매한가지로 똑같아집니다. 어떤 면에서는 더욱 복잡해지기도 합니다. 거기다가 글 목록에서는 기본적으로 목록의 위쪽에 존재하는 글이 더 최근에 등록된 글입니다. 따라서 답변 글 또한 더 최근

에 작성된 글이 위로 올라오는 것이 맞다고 할 수 있습니다. 그래서 제로보드를 비롯한 많은 게시판이 위와 같이 위쪽에 답변 글을 등록하는 방법을 사용하고 있습니다. 이런 경우를 일컬어 "굴러들어온 돌이 박힌 돌 빼낸다."라고 하지요. 그래서 우리끼리 이 알고리즘을 "굴러들어온 돌이 박힌 돌 빼내는 알고리즘"이라고 부르기로 합시다. 너무 긴가요? 그럼 줄여서 "굴돌박돌빼 알고리즘" 어떻습니까? ^^

이제 굴돌박돌빼(?) 알고리즘을 사용하는 것을 전제로 이야기해보도록 합시다.

이 알고리즘을 이용하면 굴러 들어온 돌을 위해서 박힌 돌을 모두 빼내서 자리를 옮겨주어야 합니다. 일종의 번호 이동이군요. 어떻게 하면 이 박힌 돌 전부를 번호 이동 시킬 수 있을까요? 여기서 1000번에서 2000번 사이에 존재하는 글들이 전부 답변 글이니 그 사이의 글을 찾아서 모두 1씩 빼면 되겠군요. 자 그럼 1999번은 1998번이 되어 1999번의 자리가 비게 되었으니 얼른 그 자리를 낚아채 봅시다.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1999	1	└두 번째 글의 두 번째 답변	답변 글
1998	1	└두 번째 글의 답변	답변 글
1000	0	첫 번째 글	새 글

[표 12-26] 두 번째 글의 두 번째 답변 글 추가

여기에 더 나아가서 1999번 답변 글에 답변 글을 달아 보겠습니다.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1999	1	└두 번째 글의 두 번째 답변	답변 글
1998	2	└두 번째 글의 두 번째 답변의 답변	답변 글
1997	1	└두 번째 글의 답변	답변 글
1000	0	첫 번째 글	새 글

[표 12-27] 두 번째 답변 글에 답변 글 추가

위처럼 새로운 답변 글은 1998번이 되어야 합니다. 이처럼 1998번이 되려면 기존의 1998번을 빼내야 합니다. 앞서 했던 대로 박힌 돌 전부를 번호 이동시켜 봅니다. 1000에서 2000번 사이(양 끝의 수인 1000과 2000을 제외한 수)의 글 번호를 모두 1씩 감소시키는 것이죠.

번호	깊이	제목	종류
3000	0	세 번째 글	새 글
2000	0	두 번째 글	새 글
1998	1	→두 번째 글의 두 번째 답변	답변 글
1998	2	→두 번째 글의 두 번째 답변의 답변	답변 글
1997	1	→두 번째 글의 답변	답변 글
1000	0	첫 번째 글	새 글

[표 12-28] 1000에서 2000번 사이의 글 번호를 모두 1씩 감소한 결과

어라, 1998번이 두 개가 되어 버렸습니다. 왜 이렇게 되어버렸을까요? 그 이유는 1999번이던 글이 1998번이 되었기 때문입니다. 실제로는 1999번 글은 바뀔 필요가 없었는데 1000에서 2000번 사이의 글 모두를 번호 이동시키는 바람에 1998번 글이 두 개가 되어 버린 것입니다.

그렇습니다. 뭔가 감이 잡히십니까? 박힌 돌을 빼낼 때 무조건 1000에서 2000번 사이의 글을 모두 수정하면 안 되고 이 경우에는 1999에서 1000번 사이에 있는 글을 번호 이동해야 한다는 것을 알 수 있습니다. 2000이 기준이냐 1999가 기준이냐 하는 것은 전부 어느 글에 답변이 달리느냐에 따라 결정됩니다. 2000이었을 때는 2000번에 답변 글이 달리는 경우였고 이번의 경우처럼 1999가 기준이 되는 것은 1999번의 글에 답변이 달렸기 때문입니다. 즉, 굴러 들어온 돌을 중간에 끼워 넣는데 옮길 필요가 없는 앞의 돌까지 움직일 필요가 없다는 뜻이지요. 너무나 당연한 이야기지만 실제로 소스 코드로 구현하다 보면 1000에서 2000 사이의 글을 번호 이동시키는 경우가 허다합니다.

이제 굴돌박돌빼 알고리즘이 정리되었으니 실제로 구현을 해보도록 하겠습니다.

박힌 돌 모두 번호 이동시키기

박힌 돌을 빼내서 모두 번호 이동을 시켜보겠습니다. 그러려면 앞서 설명한 것처럼 빼낼 박힌 돌의 범위를 알아내야 합니다. 일단 시작 값은 답변 글이 달릴 글의 내부 번호가 될 테니 아래에 존재하는 일반 글의 내부 번호만 알아내면 됩니다.

```
$prev_parent_thread = ceil($_POST[parent_thread]/1000)*1000 - 1000;
```

이미 여러 번 나왔으니 무엇인지 쉽게 알 수 있으리라 생각합니다. 해당 thread 값으로 일반 글의 thread 값을 찾는 부분에 1000을 빼서 아랫글의 thread 값을 구하는 것으로 읽기 페이지에서 이미 한번 다루었던 것입니다.

이를 이용하여 박힌 돌 모두의 번호를 이동시키는 쿼리문을 작성해 봅시다.

```
UPDATE $board SET thread=thread-1
WHERE thread > $prev_parent_thread and thread < $_POST[parent_thread]
```

이 쿼리문을 데이터베이스로 전송하는 소스 코드를 작성하면 다음과 같습니다.

```
$update_thread = mysql_query("UPDATE $board SET thread=thread-1
WHERE thread > $prev_parent_thread and thread < $_POST[parent_
thread]",$conn);
```

빈자리에 답변 글 끼워넣기

자~ 이제 박힌 돌을 모두 빼내서 번호 이동을 시켰으니 빈자리에 답변 글을 끼워 넣도록 합시다.

```
$query = "INSERT intO $board (thread, depth, name, pass, email, title,
         view, wdate, ip, content)";
$query .= " VALUES ('" . ($_POST[parent_thread]-1) . "'";
$query .= ",'" . ($_POST[parent_depth]+1) ."', '$name', '$pass',
          '$email', '$title', 0";
$query .= ", UNIX_TIMESTAMP(), '$REMOTE_ADDR', '$content')";
$result=mysql_query($query, $conn);
```

쿼리문은 글 쓰기에서와 동일한데 입력 값에 차이가 있습니다. 일단 답변 글의 thread 번호는 부모 글의 thread 값보다 1을 뺀 값이 됩니다. 그 이유는 앞서 박힌 돌을 모두 빼내어 부모 글의 thread 값보다 1이 작은 값의 자리가 비워져 있기 때문입니다.

또한 답변 글의 depth 값은 부모 글의 depth 값에 1을 더한 값이 됩니다. depth는 답변의 깊이를 뜻하는데 부모 글의 depth가 2라면 그 글에 대한 답변 글의 depth는 부모 글보다 하나 더 깊은 3이 될 것이기 때문입니다.

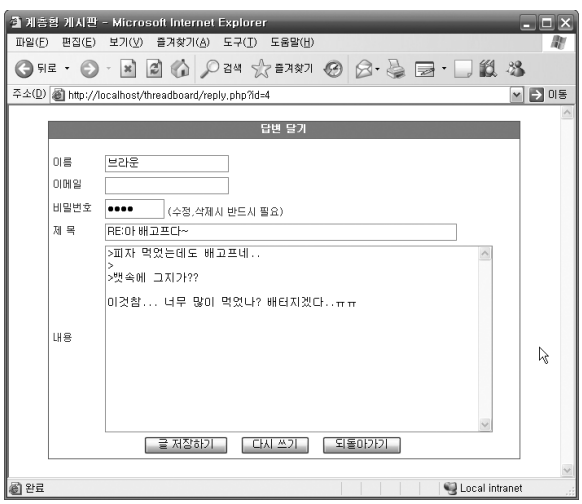
완성된 소스는 다음과 같습니다.

insert_reply.php

```
1  <?
2  //데이터 베이스 연결하기
3  include "db_info.php";
4
5  $prev_parent_thread = ceil($_POST[parent_thread]/1000)*1000 - 1000;
6
7  //원본 글보다는 작고 위값보다는 큰 글들의 thread 값을 모두 1씩 낮춘다.
8  $query = "UPDATE $board SET thread=thread-1 WHERE
9  thread > $prev_parent_thread and thread < $_POST[parent_thread]";
10 $update_thread = mysql_query($query, $conn);
11
12 //원본 글보다는 1작은 값으로 답글을 등록한다.
13 //원본 글의 바로 밑에 등록된다.
```

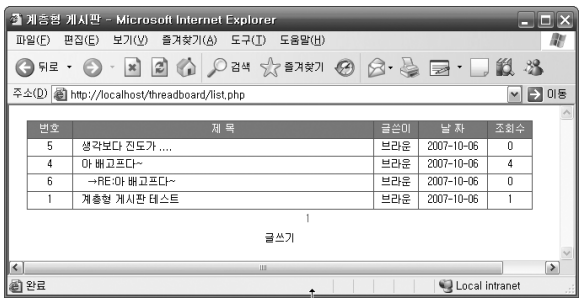
```
14 //depth는 원본 글의 depth + 1이다. 원본 글이 3(이글도 답글이군)이면 답글은 4가 된다.
15 $query = "INSERT intO $board (thread,depth,name,pass,email";
16 $query .= ",title,view,wdate,ip,content)";
17 $query .= " VALUES ('" . ($_POST[parent_thread]-1) . "'";
18 $query .= ",'" . ($parent_depth+1) ."','$name','$pass','$email'";
19 $query .= ",'$title',0, UNIX_TIMESTAMP(),'$REMOTE_ADDR'";
20 $query .= ",'$content')";
21 $result=mysql_query($query, $conn);
22
23 //데이터베이스와의 연결 종료
24 mysql_close($conn);
25
26 // 새 글 쓰기인 경우 리스트로
27 echo ("<meta http-equiv='Refresh' content='1; URL=list.php'>");
28 ?>
29 <center>
30 <font size=2>정상적으로 저장되었습니다.</font>
```

이제 답변 달기의 기능을 모두 구현하였으니 제대로 동작하는지 테스트를 해보기로 합시다.



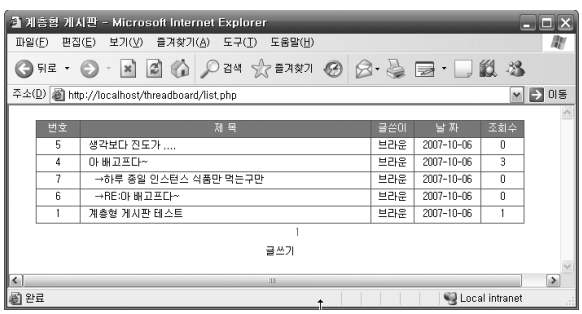
[그림 12-11] 테스트를 위한 답변 글 작성

새 글로 등록된 글 중에서 하나를 선택하여 위와 같이 답변 글을 작성해 보았습니다. 그 결과는 다음과 같습니다.



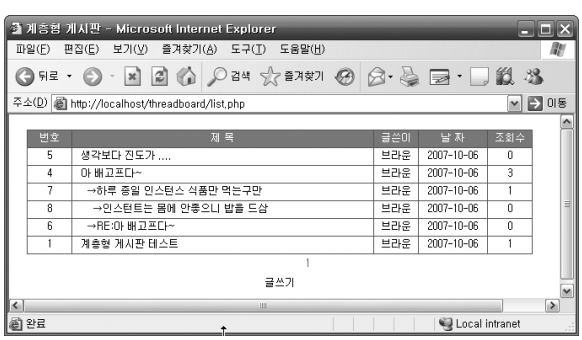
[그림 12-12] 답변 글이 포함된 글 목록 보기

6번 글에 답변 글로 잘 등록되었군요. 아직 못 미더우니 답변 글을 하나 더 등록해 보겠습니다.



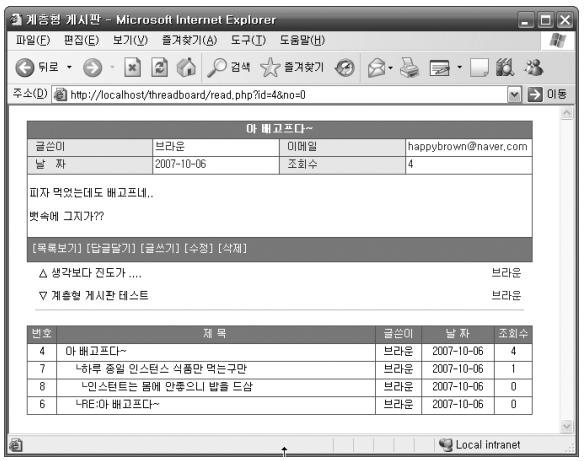
[그림 12-13] 새로 추가된 답변 글

확실히 하기 위해서 답변 글에 답변을 달아 보았습니다.



[그림 12-14] 답변 글에 대한 답변 글

모두 잘 동작하는군요. 이제 답글 달기의 테스트는 마쳤으니 글 읽기에서 관련 글의 목록이 제대로 동작하는지만 확인하면 끝이 납니다.



[그림 12-15] 관련 글 목록 보기

관련 글도 제대로 동작하고 윗글 아랫글에서도 답변 글을 제외한 일반 글들만 검색이 잘됩니다.

Section 11 계층형 게시판 1차전 마무리

계층형 게시판을 모두 마쳤습니다. 물론 1차전에 불과하지만 기존에 일반 게시판을 만들면서 고생한 보람을 충분히 느끼셨으리라 생각합니다. 많은 부분이 추가되고 수정된 것 같지만 실상 대부분의 소스 코드가 재활용되었습니다. 이렇게 배운 것은 다음 과정에 밑거름이 됩니다. 그러니 한 걸음 한 걸음 나아갈 때 "이건 너무 어려워~ 나중에 보자~"라고 생각하지 말고 특별한 경우를 제외하고는 이해가 될 때까지 여러 번 읽어 보기 바랍니다. 웹 프로그래밍에서 3번 정독해서 이해하지 못하는 것은 거의 없습니다. 저의 말을 믿고 한번 시도해 보기 바랍니다.

계층형 게시판은 여기서 끝이 아닙니다. 우리가 지금까지 만들어 본 계층형 게시판은 아직도 많은 버그를 가진 허술한 게시판입니다. 기능도 매우 부족해서 실제로 사용하기에는 부족한 점이 많습니다. 다음 장에서는 이 게시판의 문제점을 알아보고 더 좋은 게시판으로 거듭나기 위해 업그레이드를 해보겠습니다.

Chapter

13

게시판 업그레이드

초급은 모든 것이 서툴고 쉬운 것도 어렵게, 어려운 것은 더 어렵게 생각하는 때입니다. 문법을 배우고 익히기에도 힘들어서 그럴듯한 것을 스스로 만들기에는 너무 부족한 때가 바로 초급입니다. 책이나 인터넷 강의 등에서 배운 예제들을 하나씩 익혀나가면서 PHP라는 것에 감을 익히는 과정입니다.

그럼 중급은 무엇이냐? 중급은 어떻게든 무엇인가를 구현할 수 있는 상태일 때를 말합니다. 문법은 모두 익혔고 꽤 많은 함수를 알고 있으며 이를 사용해본 경험도 다수 있습니다. 그리고 어떤 과제가 생기면 어떻게든 그것을 만들 수 있는 실력을 가진 때를 말합니다. 게시판을 만들어보거나 아니면 블로그와 같은 프로그램에 도전해 보기도 하는 때가 바로 중급입니다.

마지막으로 상급은 어떤 것을 구현함에 있어서 더 나은 방법은 없는지를 고민하는 때를 말합니다. 상급이 되면 그동안의 많은 경험으로 어떤 프로그램을 만들어야 할 때 금세 머릿속에 어떻게 만들어야 할지 그림을 그릴 수 있게 됩니다. 하지만 여기서 멈추면 중급자밖에 될 수 없습니다. 중급자는 프로그램을 만들지만 상급자는 좋은 프로그램을 만들기 위해 노력합니다. 같은 계층형 게시판을 만들더라도 게시판의 성능은 하늘과 땅 차이가 될 수 있습니다. 그 이유는 상급자는 더 좋은 성능을 내기 위한 최선의 알고리즘을 계속해서 추구하기 때문입니다.

이미 우리는 일반 게시판과 계층형 게시판을 만들어 보았습니다. 사실 이것만으로도 여러분은 이제 더 이상 초보 PHPer가 아니라는 의미로 받아들일 수 있습니다. 만약 매뉴얼이나 책의 도움을 받으면서 혼자 게시판을 만들 수 있는 능력이 된다면 이미 중급자의 대열에 끼게 된 것입니다.

이제 중급에서도 더 나아가 보려고 합니다. 지금까지는 어떻게 해서든 계층형 게시판을 만들어 보았습니다. 기본적인 기능은 모두 구현하였고 게시판의 기능이 제대로 동작하지 않는다거나 하는 일은 없었습니다. 겉은 그럴듯하기 때문에 여기에 멋진 디자인을 입힌다면 모두가 겉만 보고 탄성을 지를 만한 게시판으로 변모시킬 수도 있을 겁니다. 하지만 사용을 하다 보면 금방 밑천이 드러나게 되겠지요.

이제 게시판을 좀 더 튼실하게 만들어 보고자 합니다. 그냥 게시판을 어떻게든 만들기만 하면 된다는 생각을 접고 보다 좋은 게시판을 만들어 보자는 생각을 해봅시다. 사실 이것만으로 상급자가 될 수는 없겠지만 중, 상급자의 흉내를 내보면서 자신의 실력이 점점 탄탄해지는 것을 느낄 수 있을 것입니다.

자 그럼 중, 상급의 세계로 빠져볼까요?

Section 01

게시판 속도 업그레이드

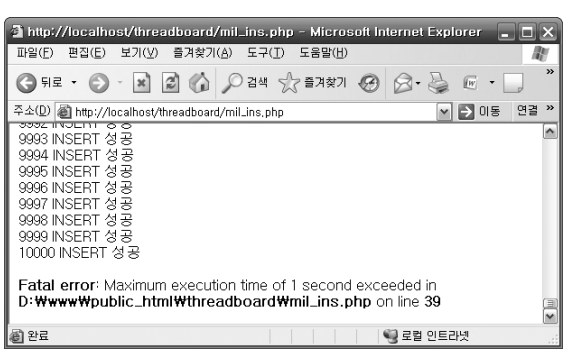
앞서 제작한 계층형 게시판은 실제로 사용해보면 그럭저럭 쓸만하다고 느낄 것입니다. "생각보다 잘 동작하는구나"하고 말이지요. 기본 기능(정상적인 쓰기, 읽기 등)만을 생각했을 때 실제로 이 게시판을 사용해도 특별한 문제는 없을 것입니다. 하지만 그것은 개인적인 홈페이지에서 사용했을 때이고 커뮤니티나 포털과 같은 곳에 이 게시판을 사용한다면 큰 낭패를 보게 될 것이 분명합니다. 먼저 부딪히는 문제가 속도입니다. 그래서 게시판의 속도를 높여보고자 합니다.

100만 개의 글을 등록하기

게시판의 속도를 체감하려면 많은 글이 등록되어 있을수록 좋습니다. 따라서 단순히 테스트의 목적으로 100만 개의 글을 게시판에 등록하여 속도 테스트를 해보고자 합니다. 실제로 하나의 게시판에 100만 개의 글이 등록되는 경우는 거의 희박합니다.

100만 개의 글을 등록하기 위해서 100만 번의 글 쓰기를 한다는 것은 바보 같은 짓이 아닐 수 없겠죠? 1초에 하나씩 등록한다고 하더라도 100만 개면 100만 초 즉, 11일 하고도 한나절 이상을 쉬지 않고 등록해야 100만 개의 글을 등록할 수 있습니다. 하지만 우리는 프로그래머이니까 프로그램으로 글을 등록하게 만들면 되겠네요.

100만 개의 글을 등록하자면 for 문으로 INSERT 문을 100만번 반복시키면 되겠습니다. 컴퓨터는 사람보다 훨씬 빠르니까 1초에 3000개씩 혹은 그 이상을 입력할지도 모릅니다. 그러면 1초에 3000개씩 입력한다고 가정을 하면 대충 5~6분이 소요될 것입니다. 그런데 이렇게 입력을 해보면 대략 10만 개 정도 등록되었을 때 다음과 같은 에러 메시지가 출력됩니다. 바로 실행 시간이 초과되었다는 메시지입니다.



[그림 13-1] 실행 시간 초과 에러(테스트를 위해서 최대 실행 시간을 1초로 설정)

여러 번 다시 시도해 보아도 매번 마찬가지일 것입니다. 왜냐하면 PHP 설정 파일에 PHP의 실행 시간을 최대 30초로 지정해 두었기 때문입니다. PHP의 설정 파일인 php.ini 파일을 열어보면 max_execution_time = 30이라고 적힌 항목을 찾을 수 있을 것입니다.

최대 실행 시간을 이렇게 지정해둔 것은 프로그래머의 실수에 의해 무한루프가 동작하는 프로그램으로 인한 시스템의 자원 낭비를 막기 위해서입니다. 만약 어떤 사용자가 우연히 무한루프에 빠지게 되어 계속해서 시스템 자원을 소모하고 있다면 최악의 경우에 다른 사람들이 웹 서비스를 사용하지 못하는 상황이 일어날 수도 있습니다. 그래서 30초의 시간을 넘기는 프로그램은 무한루프에 빠진 것으로 간주하여 에러 메시지를 출력하고 작업을 중단시키는 것입니다.

하지만 우리가 100만 개의 글을 등록하는 데 사용할 프로그램은 대략 5,6분이 소요될 예정입니다. 그렇다면 시간 제한으로 인해 30초밖에 실행이 안 될 것이 분명합니다. 그렇다고 매번 php.ini 파일을 열어 최대 실행 시간을 늘리자니 너무 번거롭기도 하고 관리자가 아닌 경우에는 불가능한 일이기도 합니다. 또한 하나의 프로그램을 위해 다른 프로그램이 모두 루프로 인한 시스템 성능 저하를 감수해야 하니 빈대 잡으려다가 초가삼간 태워버리는 격이 될 수도 있을 것 같습니다.

그래서 이렇게 오랫동안 동작해야 하는 프로그램을 제작하는 경우를 대비하여 다음과 같은 함수가 존재합니다.

```
set_time_limit(최대 실행 시간);
```

set_time_limit() 함수에 시간을 지정하면 최대 실행 시간이 변경됩니다. 만약 시간 항목에 0을 입력한다면 무제한으로 동작이 가능하게 됩니다. 하지만 이 함수는 php.ini 파일을 수정하는 것과 달리 현재 프로그램만 실행 시간의 제한이 변경됩니다. 따라서 초가삼간 태우는 경우는 없겠습니다.

그럼 소스 코드 제일 위에 set_time_limit 함수를 적어놓고 for 문으로 INSERT하면 모든 게 끝이 날 듯합니다. 하지만 갑자기 컴퓨터가 벅벅거림을 느낄 수 있을 것입니다.

[그림 13-2] INSERT 때의 CPU 사용률

100만 개 글을 입력하게 한 후 CPU의 사용 내용을 확인해 보았습니다. 그래프에서 중간부터 프로그램이 시작되었으며 프로그램이 시작하자마자 CPU의 사용률이 100%에 도달하는 것을 알 수 있습니다. 이렇게 쉬지 않고 루프가 돌기 때문에 시스템 성능에 문제를 일으키게 됩니다. 그래서 다음과

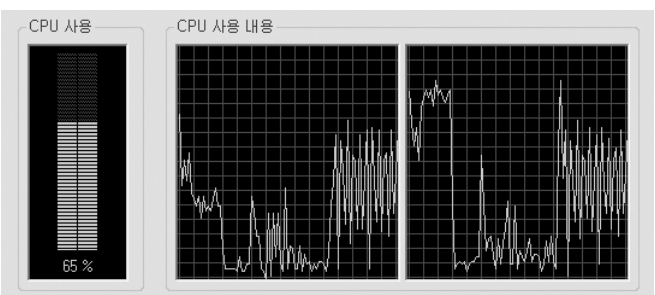
같은 함수를 이용해 루프를 도는 중간 중간에 한 번씩 CPU가 쉴 수 있게 프로그램의 실행을 잠깐씩 멈추게 합니다.

```
sleep(잠재울 초);
```

sleep() 함수는 10을 입력하면 10초간 해당 프로그램의 실행을 멈추고 CPU를 자유롭게 만듭니다. 이 사이에 다른 작업이 있다면 CPU는 다른 작업을 우선적으로 처리합니다. 따라서 중간 중간에 잠깐씩 휴식을 주어 CPU가 현재 프로그램 이외에 다른 사용자에게 제공하는 서비스를 지속적으로 처리할 수 있게끔 해줍니다. 이는 제한적인 시스템의 자원을 여러 서비스가 공유하면서 생기는 현상입니다. 어느 한 프로그램이 시스템 자원을 독점해서 쓴다면 다른 서비스는 모두 느려지거나 반응하지 않게 되어버리기 때문입니다.

```
if(($i%10000) == 0) sleep(1);
```

실제로 이와 같은 코드를 삽입하여 10000개를 입력할 때마다 1초씩 쉬게 해주었습니다. 그 결과는 다음과 같습니다.



[그림 13-3] Sleep을 주었을 때 CPU 사용률

그래프에서 절반 이후부터 프로그램이 시작되었습니다. 이 그래프를 보면 이전의 경우와 달리 CPU를 100% 사용하지 않고 50~60%와 20% 사이를 반복하고 있음을 알 수 있습니다. 바로 50~60%일 때는 글이 등록되는 시점이고 20%일 때는 1초간 휴식을 하는 시점입니다.

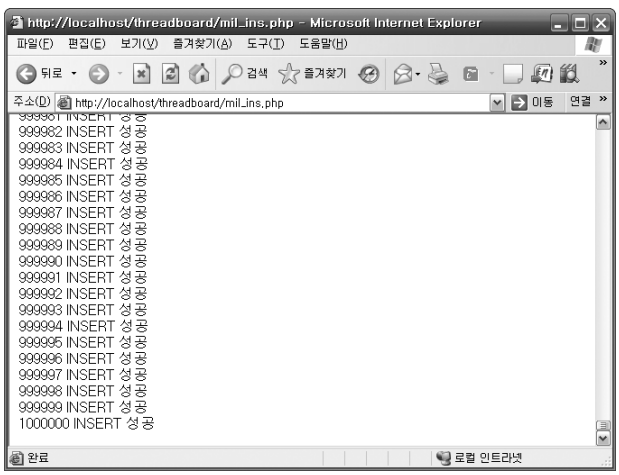
만약 보다 더 짧은 주기로 휴식을 취하게 한다면 CPU의 사용률은 훨씬 더 줄어들 것이 분명합니다. 하지만 100%의 힘으로 100미터를 달리는 것과 50%의 힘으로 100미터를 달리는 경우 당연히 시간은 두 배가 차이가 나게 될 것입니다. 이와 마찬가지로 틈틈이 쉬게 하여 다른 서비스를 제공하는데 어려움이 없게 하지만 그 반면에 처리 속도는 그만큼 느려지게 됩니다.

mil_ins.php

```
1  <?
2  //스크립트 종료할 때까지
```

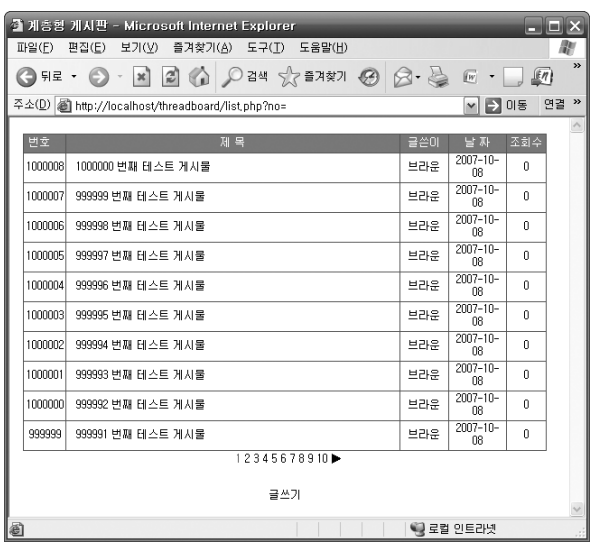
```
3  set_time_limit(0);
4
5  //데이터 베이스 연결하기
6  include "db_info.php";
7
8  // 글 등록에 대한 기본적인 정보
9  $name = "브라운";
10 $pass = "1234";
11 $email = "happybrown@naver.com";
12
13 if (ob_get_level() == 0) ob_start();
14
15 for ($i=1;$i<=1000000;$i++)
16 {
17   $title = "$i 번째 테스트 게시물";
18   $content = "$i 번째 테스트 게시물 내용";
19
20   // 답글을 위해 thread 값은 index 값의 1000배
21   $max_thread = $i * 1000;
22
23   $query = "INSERT INTO $board (id,thread,depth,name,pass,email,
24   title,view,wdate,ip,content) ";
25   $query .= "VALUES ('',$max_thread,0,'$name','$pass','$email',
26   '$title',0, UNIX_TIMESTAMP(), '$REMOTE_ADDR','$content')";
27   $result=mysql_query($query, $conn);
28
29   if ($result) {
30     $success++;
31     print("$i INSERT 성공<BR>\r\n");
32   }
33   else {
34     $failure++;
35     print("$i INSERT <B>실패</B><BR>\r\n");
36   }
37
38   // 이 프로그램이 시스템 자원을 많이 할당받는 것을 막기 위해
39   // 10000번당 1초씩 쉬어줍니다.
40   if(($i%10000) == 0) sleep(1);
41 }
42 //데이터베이스와의 연결 종료
43 mysql_close($conn);
44 ?>
```

완성된 소스를 통해서 실제로 100만 개의 글을 데이터베이스에 등록해 보겠습니다.



[그림 13-4] 100만 개의 글 등록

100만 개의 INSERT가 성공했다고 출력되는군요. 실제로 게시판에 글이 잘 저장되어 있는지 확인해 봅시다.



[그림 13-5] 100만 개의 글이 등록된 게시판

실제 게시판에 글을 등록했더니 글 목록을 보는데 조금 느려진 것 같기도 합니다. 하지만 일단은 100만 개의 글이 잘 저장된 것을 확인했습니다. 이제 게시판 속도를 업그레이드하기 위해서 조금 전에 등록한 100만 개의 글을 가지고 테스트를 할 것입니다.

실제로 데이터베이스를 통해 확인했더니 100만 개의 글이 저장된 테이블의 크기가 127MB인 것을 확인할 수 있었습니다. 정말 엄청난 양이 아닐 수 없습니다.

여기서 잠깐

서버와 설정에 따라서 위 코드를 실행하면 웹 브라우저가 멈춘듯한 현상이 생길 수 있습니다. 이는 출력값이 제대로 웹 브라우저에 출력되지 않아서 생기는 현상입니다. 그 이유는 PHP가 PHP 버퍼에 출력 결과를 저장해두고 웹 브라우저로 출력값을 전송하지 않았기 때문에 발생합니다. 특히 Windows 기반의 웹 서버를 사용하는 경우 PHP 스크립트가 종료될 때까지 PHP 버퍼에 결과를 쌓아두기 때문에 모든 작업이 끝날 때까지 화면에 어느 것도 출력되지 않을 수 있습니다. 그래서 PHP에는 flush() 라는 함수가 존재합니다. 이 함수는 PHP 버퍼에 쌓여 있는 결과값을 강제적으로 웹 브라우저에 전송하도록 명령합니다. 그래서 위와 같이 처리 결과를 순간순간 보고 싶은 경우에는 flush() 함수를 호출하면 됩니다. 위와 같은 소스는 sleep 함수를 호출하기 전에 flush 함수를 호출하면 특정시간마다 PHP 버퍼에 있는 값을 웹 브라우저로 호출하므로 중간중간 그 결과를 볼 수 있습니다.

인덱스를 걸어보자

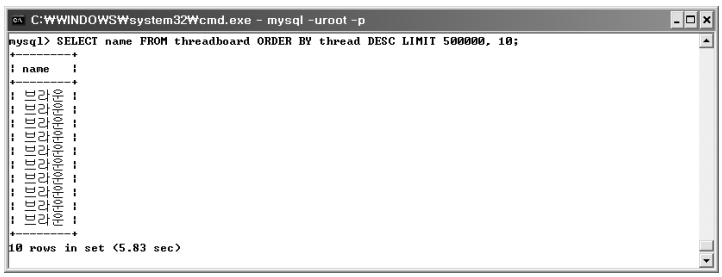
100만 건의 게시물을 등록한 후 실제 사용을 해보면 리스트의 앞부분은 그럭저럭 사용할 만하나 50만 번째 글의 목록처럼 중간 부분이나 가장 뒷부분의 글 목록을 보면 매우 느려지는 것을 알 수 있습니다. 이는 데이터베이스에 레코드가 너무 많아서 이를 정렬하고 출력하는데 매우 많은 시간이 소요되기 때문입니다.

우리는 계층형 게시판을 만들면서 기존의 게시판 구조에 몇 가지 필드를 추가하거나 수정하는 것으로 계층형 게시판의 스키마를 작성했습니다. 여기서 기본키를 여전히 id 값에 두고 있었습니다. id 값이 여러모로 사용되기는 하지만 내부적으로 글 목록을 계산하거나 하는 것은 모두 id 값이 아닌 thread 값을 이용하기 때문에 thread 값에도 인덱스를 설정해주는 것이 필요합니다.

여기서 인덱스를 설정할 때 UNIQUE로 설정합니다. 왜냐하면 thread 값은 중복되지 않는 유일한 값이기 때문입니다. UNIQUE를 설정해두면 테스트 도중에 실수로 중복되어 저장되거나 하나의 글에 대한 답변 글이 1000개가 넘어갔을 때 예를 들어 1999부터 1001까지 모두 답변 글로 채워지고 다시 새로운 답변 글이 등록될 때 1000이 되는 것을 방지해 줍니다. 인덱스에 유일(UNIQUE)하다고 지정해 두었기 때문에 키와 같이 동일한 값이 입력되면 자동으로 에러를 출력해 줍니다.

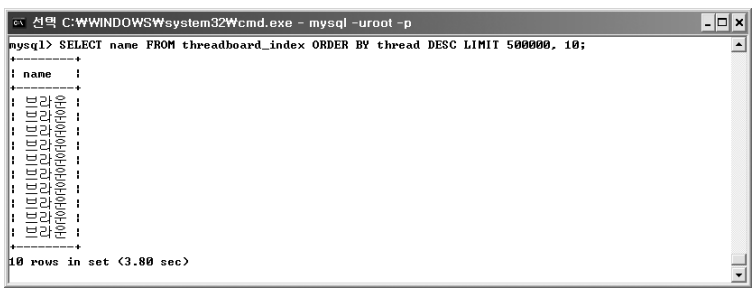
```
ALTER TABLE threadboard ADD UNIQUE thread_index(thread);
```

인덱스의 효과가 어떻게 나타나는지 실제로 테스트를 해보도록 하겠습니다.



[그림 13-6] 인덱스를 설정하지 않은 경우의 성능

thread 필드에 인덱스를 설정하지 않은 상태로 위와 같은 쿼리를 실행했을 때 5.83초의 시간이 소요되었습니다. 동일한 쿼리를 thread 필드에 인덱스를 설정한 후 실행하면 다음과 같습니다.



[그림 13-7] 인덱스를 설정한 후의 성능

thread 필드에 인덱스를 설정하고 난 후 결과는 위와 같이 3.80초의 시간이 걸렸습니다. 이는 인덱스를 설정하지 않았을 때와 비교하여 대략 2초의 시간이 줄어들었음을 알 수 있습니다.

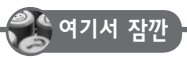
이와 같이 인덱스는 검색 시에 매우 효율적으로 결과를 정렬하거나 찾아주기 때문에 매우 중요한 성능 향상 요인이 됩니다. 그러나 인덱스가 장점만 있는 것은 아닙니다. MySQL의 경우 인덱스를 구현하기 위하여 B-Tree라는 구조를 사용하는데 단순히 말하자면 이름 그대로 나뭇가지가 뻗어나가듯이 데이터를 저장합니다. 그런데 이 알고리즘은 데이터를 삽입할 때 자기가 들어가야 할 자리를 찾아서 삽입해야 하고 또한 삭제하는 경우에는 빠져나간 빈자리를 채워주는 행위를 반드시 해야 합니다. 그렇지 않으면 나뭇가지들이 연결되지 않고 끊어지거나 나뭇가지가 한쪽으로 치우쳐버리는 현상이 생길 수 있습니다. B-Tree에서 B의 의미가 균형 있는 나무 구조(Balanced Tree)라는 뜻이기 때문에 나뭇가지가 한쪽으로 치우치지 않고 양쪽으로 균형이 잡혀야 합니다. 그래서 B-Tree 구조를 사용하면 삽입과 삭제에 평상시보다 더 많은 시간이 소요됩니다. 하지만 일반적인 게시판에서 글이 등록되거나 삭제되는 횟수보다는 글을 조회하는 경우가 훨씬 많기 때문에 인덱스로 사용합니다.

그렇다면 실제로 목록을 구현하기 위해서 사용되는 쿼리를 적용하여 얼마나 시간 차이가 나는지 확

인해 보도록 하겠습니다.

```
SELECT * FROM threadboard_index ORDER BY thread DESC LIMIT 500000,10;
10 rows in set (3.70 sec)

SELECT * FROM threadboard ORDER BY thread DESC LIMIT 500000,10;
10 rows in set (4.67 sec)
```

위와 같이 인덱스를 설정한 경우가 0.55초 가량 더 빠른 것을 알 수 있습니다.



인덱스를 통해서 글의 검색 속도가 향상되는 것을 확인할 수 있었습니다. 그런데 그 결과를 잘 살펴보면 이상한 점을 발견할 수 있는데, 바로 인덱스를 설정하지 않았을 때 쿼리의 변화로 인해서 줄어든 속도 변화입니다.

① SELECT name FROM threadboard ORDER BY thread DESC LIMIT 500000,10;
② SELECT * FROM threadboard ORDER BY thread DESC LIMIT 500000,10;

①의 처리 속도는 5.83초였고 ②의 처리 속도는 4.6초였습니다. 단순히 쿼리를 변경하였을 뿐인데 속도가 빨라졌습니다. 그 이유가 무엇일까요?

일반적으로 레코드를 검색할 때 필요한 필드만을 쓰는 것이 성능적으로 더 우수하다고 합니다. 메모리 측면이나 속도 면에서 더 우수하다고 말하는데 여기서는 조금 다릅니다.

①의 쿼리는 thread로 정렬하였고 그 결과로 name 값을 출력했습니다. 그런데 ②의 쿼리는 모든 항목을 다 출력했습니다. 이러한 차이로 속도의 차이가 나는 것은 MySQL의 파일정렬(filesort) 알고리즘 때문입니다. MySQL의 파일정렬 알고리즘은 두 가지가 있는데 하나는 일반적으로 사용하는 ORDER BY 절에 사용된 필드를 기준으로 정렬하는 방법이고 다른 하나는 ORDER BY 절에 사용된 필드 이외에 쿼리에서 사용된 모든 필드를 참조하여 정렬하는 방법입니다. MySQL은 두 알고리즘 중에서 결과가 우수할 것으로 예상되는 쪽을 선택하여 파일정렬을 시행합니다.

①의 경우는 ORDER BY 절에 사용된 thread 필드에 인덱스가 설정되지 않았고 쿼리에서 사용된 다른 필드(name)에도 인덱스가 설정되어 있지 않습니다. 그래서 인덱스의 효과를 보지 못하고 순수하게 퀵정렬(qsort)를 시행합니다.

그러나 ②의 경우는 ORDER BY 절에는 인덱스가 설정되지 않은 필드가 사용되지만 SELECT 다음에 별표(*)가 사용되면서 모든 필드를 사용하게 되고 그 필드 중에서 기본키로 설정된 id 값을 통해서 인덱싱을 하게 됩니다. id 값은 인덱스가 설정되어 있기 때문에 이 인덱스의 도움을 받아 더 빠르게 정렬을 한 것입니다.

아주 간단하게 인덱스를 하나 설정하는 것만으로 게시판의 성능을 향상시킬 수 있었습니다. 그런데 잘 생각해보면 실제로 스키마에서 id 값이 불필요하다는 것을 깨달을 수 있습니다. 내부적으로 글

번호를 처리하는 것은 thread 값이고 thread 값은 중복되지 않는 값이므로 글을 구분 짓는 기본키(Primary Key)로 사용될 수 있습니다. 따라서 thread 값을 글 외부적으로도 사용하게 될 경우 단지 글 목록에서 1000씩 증가하는 매우 큰 글 번호가 출력된다는 것 이외에는 차이가 없습니다. 그러나 이 문제는 글이 몇 번째 글인지 계산하여 순번을 출력해주는 방법으로 해결할 수 있습니다.

thread를 글의 기본키로 설정하고 id 필드를 제거하면 인덱스를 id와 thread 두 개를 설정할 필요가 없어지기 때문에 공간적 낭비를 줄일 수 있습니다. 인덱스 하나당 100만 건에 대략 10메가 정도의 공간이 소모되기 때문에 이러한 이득을 얻게 되고 또한 레코드 하나 당 4바이트의 공간이 절약됩니다.

즉, 기존 게시판의 스키마에 thread와 depth를 추가하는 것이 아니라 depth만 추가하고 기존의 id를 thread라는 이름으로 변경하거나 thread를 id라는 이름으로 변경했다고 생각하고 id 값을 사용하는 방법으로 간단히 처리할 수 있습니다. 단, id 필드는 자동으로 1씩 증가하는 AUTO_INCREMENT가 설정되어 있기 때문에 이를 제거해야 합니다. 왜냐하면 thread 값은 1씩 증가하는 것이 아니라 1000씩 증가하거나 답변 글은 계산된 결과값이 thread 값이 되기 때문입니다.

이 부분에서는 일단 기존의 스키마를 그대로 사용하여 불필요하지만 id 값과 thread 값을 공존시키게 합니다. 왜냐하면 앞서 언급하였듯이 글 번호를 글 목록에 표시할 때에 너무 큰 수가 적히는 것을 처리하려면 글의 번호를 계산하는 부분을 추가해야 하기 때문입니다. 이 부분은 다음에 다시 다루기로 합니다.

ORDER BY thread DESC는 느리다

앞서 인덱스를 통해서 게시판의 성능을 향상시켰지만 여전히 속도가 불만족스럽습니다. 속도를 조금 더 높여보도록 합시다.

게시판에서 글 목록을 구현할 때 글을 순서에 맞게 출력하기 위해 글을 정렬하고 있습니다. 일반 게시판에서는 ORDER BY id DESC를 사용했었고 계층형 게시판에서는 ORDER BY thread DESC를 사용했습니다.

그런데 id는 새 글이 등록될 때마다 1씩 증가하고 thread는 일반 글이면 1000의 배수로 증가하는 값이다 보니 최근에 게시된 글일수록 id와 thread 값이 커지게 됩니다. 이런 상황에서 최신의 글을 우선적으로 보여주어야 하는 글 목록을 위해 정렬할 때 부득이하게 ORDER BY thread DESC라는 것을 쓸 수밖에 없었습니다.

정렬에서 DESC는 큰 수부터 작은 수로 정렬하는 즉, 내림차순 정렬입니다. 일반적으로 ASC로 정렬하는 것보다 DESC로 정렬하는 것이 느리기 때문에 정렬할 때 되도록이면 ASC 정렬로 구현하는 것이 좋습니다.

'최근 글이 상단에 오게 한다'라는 생각에서 만들어진 역순 정렬을 이용한 리스트 알고리즘은 직관적이어서 생각해내기 쉽고 구현하는 것도 어려움이 없었지만 최적의 성능을 나타내지는 못합니다. 그래서 성능을 더 높이기 위해 때로는 직관적인 알고리즘이 아닌 그에 역행하는 알고리즘을 선택해야 하는 경우가 종종 있습니다.

ASC형 게시판을 만들려면 가장 최근에 등록된 일반 글이 가장 작은 값을 가지고 있어야 합니다. 그래야 점점 커지는 순차정렬을 사용할 수 있을 테니까요. 그래서 새로 등록되는 글이 점점 작아지는 값을 갖게 하기 위한 방법을 생각해야 합니다.

일반적으로 ASC형 게시판을 구현하기 위해 다음과 같은 두 가지 방법을 사용합니다.

① 글 번호를 음수로 하여 1000씩 빼가면서 등록

② 글 번호를 9999999000과 같은 큰 값에서 1000씩 빼가면서 등록

두 가지 방법 모두 기존 게시판에서 약간 수정하는 것만으로 변환이 가능합니다. 이 중에서 우리는 음수를 이용하여 ASC형 게시판을 구현해 보도록 하겠습니다.

본격적으로 음수를 사용하는 게시판을 만들기에 앞서 기존의 100만 개 레코드를 음수로 변경하도록 합니다.

```
UPDATE threadboard SET thread = -1*thread;
```

콘솔창에 위와 같이 쿼리를 던지면 꽤 오랜 시간이 소요됩니다. 100만 개의 레코드를 모두 수정해야 할 테니 시간이 걸리는 것이 당연합니다. 컴퓨터의 성능에 따라 짧게는 몇십 초에서 몇 분 걸릴 터이니 막간을 이용하여 눈 마사지를 해보는 것은 어떨까요? 장시간 가까이에 있는 물체를 보면 근시가 될 우려가 있습니다. 50분마다 10분 정도 멀리 바라보는 등의 방법으로 눈을 쉬게 해줍시다.

글 번호를 모두 수정하면 다음과 같이 글 번호가 등록되어 있을 것입니다.

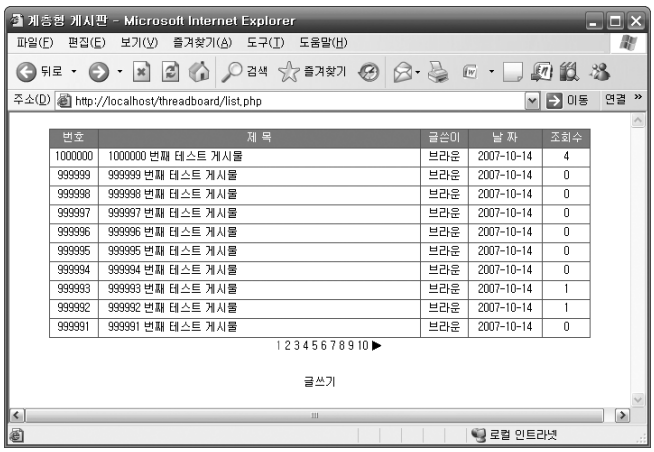
글 번호	깊이	제목	종류
-3000	0	세 번째 글	새 글
-2000	0	두 번째 글	새 글
-1000	0	첫 번째 글	새 글

[표 13-1] 음수의 thread 값을 가지는 계층형 게시판

기존 글 번호를 음수로 바꾸어 줌으로써 가장 작은 값을 가지는 글이 가장 늦게 등록된 글로 변경되었습니다. 따라서 다음과 같이 오름차순으로 정렬하여 글 목록을 구현할 수 있습니다.

```
SELECT * FROM $board ORDER BY thread LIMIT $no, $page_size;
```

수정된 쿼리를 list.php 파일에 적용해보면 다음과 같이 올바르게 정렬되는 것을 확인할 수 있습니다.



[그림 13-8] ORDER BY thread ASC로 구현된 게시판

이제 기존의 레코드는 모두 변환했으니 새로운 글을 하나 추가해 보도록 합시다. 새롭게 글을 등록하려면 우선 바뀐 알고리즘대로 새 글 쓰기 부분을 수정해야 합니다. 일단 기존의 새 글 쓰기 알고리즘부터 살펴 봅시다.

① 글 번호의 최대값을 구한다.

② 구해진 최대값을 이용하여 일반 글의 번호를 구한다.

③ 일반 글의 번호에 1000을 더한다.

위와 같은 알고리즘으로 새 글을 등록하였으나 위쪽이 큰 값을 갖던 것이 아래쪽이 큰 값을 갖도록 변경되었기 때문에 알고리즘을 수정해야만 합니다.

① 글 번호의 최소값을 구한다.

② 구해진 최소값을 이용하여 일반 글의 번호를 구한다.

③ 일반 글의 번호에 1000을 뺀다.

수정된 알고리즘은 위와 같습니다. 앞서 기존 알고리즘과 비교해 보면 마치 반대말하기를 하는 것 같습니다. 글 번호의 최소값을 구하는 이유는 새 글은 항상 가장 작은 값을 가져야 하기 때문입니다. 그래서 가장 작은 값에 1000을 빼면 새 글의 번호가 가장 작은 값을 갖게 됩니다.

따라서 새 글 쓰기의 소스 코드는 다음과 같이 변경되어야 합니다.

```
$max_thread_result = mysql_query("SELECT min(thread) FROM
$board",$conn);
$max_thread_fetch = mysql_fetch_row($max_thread_result);
$max_thread = floor($max_thread_fetch[0]/1000)*1000-1000;
```

여기서 주의할 점은 기존에는 일반 글을 구하고자 올림(ceil 함수를 사용)을 했었는데 반해 내림

(floor 함수를 사용)을 한다는 것입니다.

글 번호	깊이	제목	종류
-2999	1	└세 번째 글의 답변	답변 글
-2000	0	두 번째 글	새 글
-1000	0	첫 번째 글	새 글

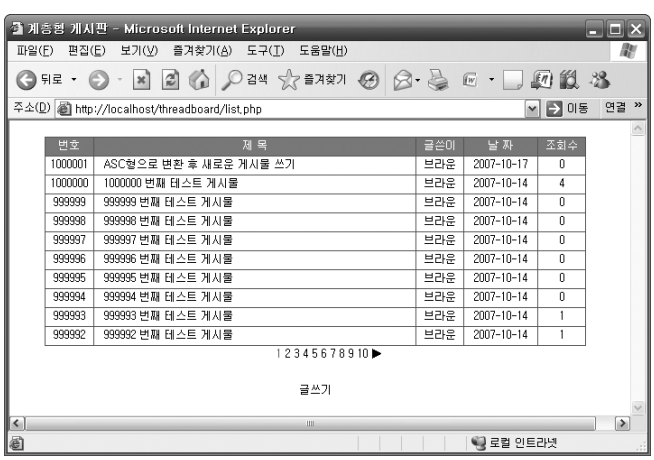
[표 13-2] 다음 글 번호를 구하기 위한 예제 상황

3000번의 글이 삭제된 경우 글 번호의 최소값은 -2999가 됩니다. -2999로부터 -3000을 얻어야 할 텐데 만약 기존과 같이 올림을 해버린다면 -2.999의 올림값 -2가 됩니다. -2에 1000을 곱하면 -2000이 되니 엉뚱한 글을 자신의 부모 글이라고 생각하게 됩니다. 그래서 올림 대신에 내림을 하면 -2.999는 -3이 되어 이를 통해 -3000을 얻을 수 있습니다. 얻어진 -3000에 1000을 빼서 결국 다음과 같이 -4000번에 새 글을 등록하게 됩니다.

글 번호	깊이	제목	종류
-4000	0	네 번째 글	새 글
-2999	1	└세 번째 글의 답변	답변 글
-2000	0	두 번째 글	새 글
-1000	0	첫 번째 글	새 글

[표 13-3] 새로 추가된 네 번째 글

수정된 새 글 쓰기를 실제로 적용하면 다음과 같이 제대로 등록되고 있음을 확인할 수 있습니다.



[그림 13-9] ASC형 게시판

여기서 잠깐

위의 표를 보면 한 가지 이상한 것을 발견할 수 있습니다. 그것은 –2999번의 글이 마치 –4000번 글의 답변처럼 보인다는 것입니다. 이를 해결하기 위해서 많은 게시판이 답변 글이 있는 경우 삭제를 못하도록 금지하는 정책을 쓰고 있습니다. 이 부분은 뒤에서 다시 다룰 것입니다.

이제 답변을 새롭게 추가해 봅시다. 답변 글이 제대로 등록되려면 다음과 같이 등록되어야 합니다.

글 번호	깊이	제목	종류
–3000	0	세 번째 글	새 글
–2000	0	두 번째 글	새 글
–1999	1	└두 번째 글의 답변	답변 글
–1000	0	첫 번째 글	새 글

[표 13–4] ASC형 게시판에서 답글 달기

우선 기존의 답변 달기 알고리즘을 살펴보겠습니다.

① 부모 글의 번호로부터 일반 글의 번호를 찾는다.

② 일반 글에 1000을 빼서 바로 아래의 일반 글 번호를 계산한다.

③ 부모 글과 아래의 일반 글 사이에 존재하는 답변 글의 번호에 1씩 뺀다.

④ 비워진 자리에 답변 글을 등록한다.

그러나 새 글 쓰기와 마찬가지로 알고리즘을 변경해야 합니다.

① 부모 글의 번호로부터 일반 글의 번호를 찾는다.

② 일반 글에 1000을 더해서 바로 아래의 일반 글 번호를 계산한다.

③ 부모 글과 아래의 일반 글 사이에 존재하는 답변 글의 번호에 1씩 더한다.

④ 비워진 자리에 답변 글을 등록한다.

새 글 쓰기와 마찬가지로 빼는 것을 모두 더해줍니다. 하나씩 차례차례 알아가 봅시다.

```
$prev_parent_thread = floor($_POST[parent_thread]/1000)*1000 +1000;
```

앞서 새 글 쓰기에서 언급했듯이 내림을 이용하여 일반 글의 번호를 찾고 1000을 더해서 아래의 일반 글 번호를 계산합니다.

```
$update_query = "UPDATE $board SET thread=thread+1 WHERE thread <
$prev_parent_thread and thread > $_POST[parent_thread]";
```

```
$update_query .= " ORDER BY thread DESC";
$update_thread = mysql_query($update_query, $conn);
```

부모 글과 아래의 일반 글 사이에 존재하는 답변 글의 번호에 1씩을 더합니다. 그런데 여기서 아주 특이한 것이 있습니다. 바로 ORDER BY thread DESC인데요, SELECT 문도 아니고 UPDATE 문에 정렬하는 키워드가 들어가 있습니다.

다음과 같은 경우를 생각해 봅시다.

글 번호	변경될 글 번호	제목	종류
–3000	–3000	세 번째 글	새 글
–2000	–2000	두 번째 글	새 글
–1999	–1999	└두 번째 글의 답변	답변 글
새로 추가	–1998	└두 번째 글의 답변의 답변	답변 글
–1998	–1997	└두 번째 글의 답변	답변 글
–1997	–1996	└두 번째 글의 답변	답변 글
–1000	–1000	첫 번째 글	새 글

[표 13–5] 새로운 답변 글로 인한 글 번호 변경

이미 두 번째 글에는 많은 답변 글이 달렸습니다. 이때 –1999번 글에 답변을 새로 등록하려고 한다면 기존의 –1998, –1997번은 각각 –1997, –1996이 되어야 합니다. 그래야 –1998번이 빈자리가 되어 새 답변 글이 등록될 것이기 때문입니다. 그런데 MySQL이 UPDATE 문을 실행하면서 작은 값부터 먼저 수정하기 시작합니다. –1998이 가장 작은 값이니 –1998을 –1997로 바꾸려고 시도합니다. 그런데 –1997이 이미 존재하기 때문에 등록할 수가 없습니다. 그래서 가장 작은 값부터 업데이트하는 것이 아니라 가장 큰 값부터 업데이트를 시키고자 ORDER BY thread DESC를 넣은 것입니다. 이렇게 되면 우선 –1997번을 –1996으로 변경하고 다시 –1998을 –1997로 변경하므로 이상 없이 빈자리를 만들 수 있습니다.

여기서 잠깐

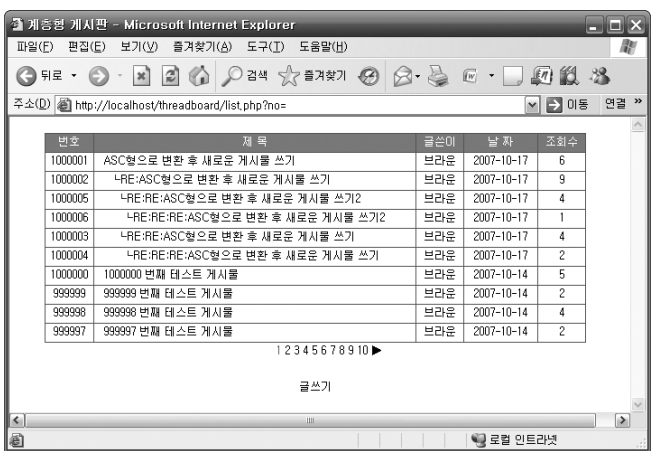
앞서 글의 번호를 양수로 사용할 때는 업데이트할 때 정렬을 한 적이 없었습니다. 그렇다면 왜 양수를 사용했을 때는 이와 같은 상황이 일어나지 않았을까요? 글의 번호가 양수일 때는 답변 글을 등록하기 위해 1을 빼서 빈자리를 만들어 주었습니다. 이때 새로운 답변이 들어갈 위치를 기준으로 아래에 존재하는 답변 글들은 모두 내림차순으로 정렬되어 있습니다. 그래서 그 중 가장 작은 값부터 1씩 빼 나가면 중복되는 값 없이 모두 잘 등록됩니다.

이제 새로운 답변 글이 들어갈 빈자리를 만들었으니 빈자리에 답변 글을 끼워넣어 봅시다.

```
$query = "INSERT INTO $board (thread, depth, name, pass, email, title,
         view, wdate, ip, content)";
$query .= " VALUES ('" . ($_POST[parent_thread]+1) . "'";
$query .= ",'" . ($parent_depth+1) ."','$name','$pass','$email',
          '$title',0";
$query .= ", UNIX_TIMESTAMP(),'$REMOTE_ADDR','$content')";
$result=mysql_query($query, $conn);
```

새로운 답변 글은 부모 글의 바로 아래에 위치합니다. 따라서 부모 글 번호의 바로 아래 번호 즉, 오름차순이니 1을 더한 값이 새로운 답변 글의 번호가 됩니다.

완성된 답변 글 쓰기를 테스트해보면 다음과 같습니다.



[그림 13-10] 완성된 ASC형 게시판 목록 보기

새 글 쓰기와 답변 글 쓰기가 모두 수정되었으니 마지막으로 글 읽기 페이지를 수정해 봅시다. 글 읽기의 경우 윗글 아랫글을 수정해야 하고 하단에 위치한 관련 글 목록을 새로운 알고리즘에 맞게 수정해야 합니다.

우선 윗글 아랫글은 쿼리를 서로 바꾸어 주기만 하면 됩니다. 아랫글이 번호가 더 커지기 때문에 아랫글을 찾으려면 기존의 윗글을 찾듯이 글 번호가 큰 것 중에서 가장 작은 값을 갖는 글을 찾으면 됩니다. 이와 마찬가지로 윗글을 찾으려면 글 번호가 작은 것 중에서 가장 큰 값을 갖는 글을 찾으면 됩니다.

```
$query=mysql_query("SELECT id, name, title FROM $board WHERE thread <
$row[thread] and depth=0 ORDER BY thread DESC LIMIT 1", $conn);
$up_id=mysql_fetch_array($query);

$query=mysql_query("SELECT id, name, title FROM $board WHERE thread >
$row[thread] and depth=0 LIMIT 1", $conn);
$down_id=mysql_fetch_array($query);
```

이와 더불어 관련 글 목록 보기를 수정해주면 다음과 같습니다.

```
$thread_start = floor($row[thread]/1000)*1000 + 1000;
$thread_end = floor($row[thread]/1000)*1000;
$query = "SELECT * FROM $board WHERE thread >= $thread_end and thread
< $thread_start ORDER BY thread";
$result = mysql_query($query, $conn);
```

여기서 잠깐

큰 값에서 1000씩을 빼어 나가는 형식은 다음과 같이 등록됩니다.

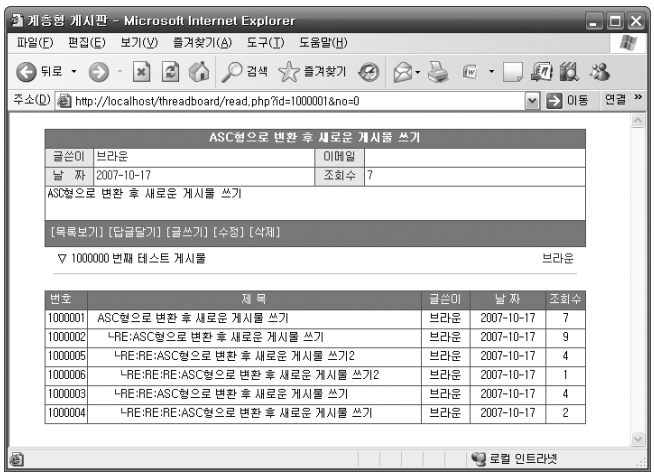
글 번호	깊이	제목	종류
999997000	0	세 번째 글	새 글
999998000	0	두 번째 글	새 글
999999000	0	첫 번째 글	새 글

여기에 답변 글을 달면 다음과 같이 등록됩니다.

글 번호	깊이	제목	종류
999997000	0	세 번째 글	새 글
999998000	0	두 번째 글	새 글
999998001	1	ㄴ두 번째 글의 답변	답변 글
999999000	0	첫 번째 글	새 글

이 방법은 앞서 음수로 변경하는 방법과 마찬가지로 약간의 수정으로 게시판을 ASC형으로 변경할 수 있습니다. 그러면 음수형과 큰 수에서 빼는 방법 중에 어느 것이 더 좋은 방법일까요? 성능은 두 방법 모두 엇비슷해서 차이가 없습니다. 그러나 일반적으로 음수형은 구현할 때 크기 비교를 거꾸로 생각해야 한다는 것 때문에 많은 실수를 범하기 쉽습니다. 부등호의 방향이라든지 더해야 할지 빼야 할지 혼란스럽습니다. 그도 그럴 것이 우리는 대부분 양수를 계산하고 이에 익숙해 있기 때문에 어쩔 수 없으리라 생각됩니다. 그리고 음수형의 경우 단점으로 지적하고 싶은 것은 공간 낭비입니다. 양수형인 경우에는 글 번호를 부호 없는 정수(Unsigned Integer)로 설정하여 0부터 4294967295까지의 공간을 거의 대부분 사용할 수 있지만 음수형은 –2147483648부터 0까지밖에 사용하지 못하기 때문에 절반의 공간을 허비하게 됩니다. 그래도 200만 건 이상 등록이 가능하니 실질적으로 사용하는 데는 큰 문제가 없긴 합니다.

완성된 글 읽기 페이지는 다음과 같습니다.



[그림 13-11] 완성된 ASC형 연관 글 목록 보기

뒷 페이지로 갈수록 느려진다

앞에서는 DESC와 ASC 정렬 방식으로 인한 성능 개선에 대해 알아보았습니다. 그러나 아직 문제가 많이 남아 있습니다. 100만 개의 글을 등록하고 테스트를 해보면 글 전체 목록 중 앞부분 즉, 최근에 등록된 글은 빠르게 리스트가 보이지만 마지막 페이지처럼 초기에 등록된 글의 페이지를 보고자 할 때는 10여 초씩 걸리게 됩니다. 물론 기존 버전과 달리 버전 5.x 대의 MySQL은 목록 페이지의 뒷부분에서도 무한정 시간이 오래 걸리지 않도록 바뀌었습니다. 그러나 여전히 글 목록을 보려면 3, 4초 이상을 기다려야 합니다. 3, 4초는 짧은 순간 같지만 실제로 게시판을 사용해보면 굉장히 답답한 느낌이 드는 것을 감출 수 없습니다.

일반적으로 검색의 속도가 느리면 먼저 인덱스를 설정하는 것을 생각합니다. 이미 인덱스를 설정하기도 했지만 워낙 레코드의 수가 많기 때문에 100만 개에 해당하는 인덱스만 10MB의 용량을 차지할 정도로 큽니다. 그래서 인덱스를 검색하는 것도 큰 부담이 되어버린 것입니다. 따라서 인덱스만으로 이것을 극복하기는 무리입니다.

어떻게 하면 좀 더 빠르게 글을 검색할 수 있을까요?

이 문제는 쿼리 하나를 최적화하려고 해서는 답을 얻기가 힘듭니다. 일반적으로 쿼리 하나로 처리하는 것이 두 개로 처리하는 것보다 훨씬 효율적이고 빠르다고 생각합니다. 그래서 간혹 보면 몇 번에 걸쳐 쿼리를 시도하는 것이 훨씬 편하고 빠름에도 하나의 쿼리를 작성하기 위해서 매우 길고 복잡한 쿼리를 작성하는 사람들이 있습니다. 이는 데이터베이스 수업의 시험에서 "이렇게 결과가 출력되도록 하나의 쿼리로 작성하시오."라는 문제가 아닌 이상 굳이 이렇게 작성할 필요가 없습니다.

그래서 글 목록을 출력하는 쿼리를 두 개의 쿼리로 나누고자 합니다. 쿼리는 다음과 같은 과정으로 수행됩니다.

① 100만 개 레코드 중에서 50만 번째 레코드를 찾는다.

② 찾은 레코드 번호부터 검색하여 10개의 글 목록을 가져온다.

이를 쿼리로 작성해 보면 다음과 같습니다.

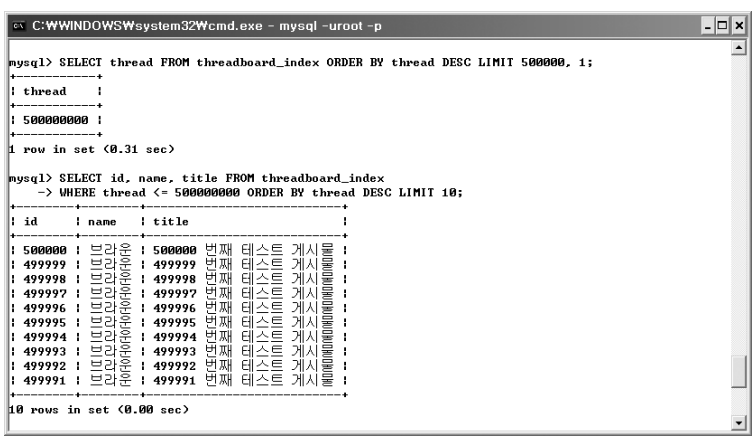
```
SELECT thread FROM threadboard ORDER BY thread DESC LIMIT 500000,1;
```

이 쿼리를 통해서 얻은 thread 값을 $thread 변수에 저장했다고 한다면 다음과 같은 쿼리를 통해서 50만 번째부터 10개의 글 목록을 얻을 수 있습니다.

```
SELECT * FROM threadboard WHERE thread <= $thread
ORDER BY thread DESC LIMIT 10;
```

역순으로 정렬되기 때문에 10개의 글은 점차 작아지는 thread 값을 갖습니다. 따라서 앞서 구해진 thread 값보다 작은 thread 값을 갖는 것 중에 역순 정렬을 하여 그 중 10개만 선택하면 글 목록 페이지를 구현할 수 있습니다.

다음은 나누어진 쿼리를 각각 실험한 결과입니다.



[그림 13-12] 쿼리의 실행결과

두 결과를 합쳐도 1초가 채 되지 않습니다. 놀랍지 않습니까?

첫 번째 쿼리는 단지 검색을 thread 필드만 사용하도록 한 것이 기존 쿼리와의 차이입니다. 그런데 너무나 확연히 차이가 납니다. 그 이유는 검색 항목이 단지 thread밖에 없기 때문에 직접 테이블을 검색한 것이 아니라 인덱스만을 검색했기 때문입니다. 이렇게 빠르게 검색한 thread 값을 통해서 글 목록을 찾으면 놀랍게도 0초가 걸립니다. 이는 잘못 출력된 것이 아니라 0.00x 초와 같이 매우

짧은 시간에 처리되므로 0초로 출력되는 것입니다. 따라서 기존 list.php 파일의 글 목록을 구현하는 부분을 수정해 봅시다.

기존에는 다음과 같이 하나의 쿼리로 작성되어 있었습니다.

```
$query = "SELECT * FROM $board ORDER BY thread DESC
          LIMIT $no,$page_size";
$result = mysql_query($query, $conn);
```

이를 다음과 같은 두 개의 쿼리문으로 바꾸어 줍니다.

```
//1. 글 목록의 첫 번째 글 찾기
$query = "SELECT thread FROM $board ORDER BY thread DESC LIMIT $no, 1";
$result = mysql_query($query, $conn);
$row = mysql_fetch_row($result);
$start_thread = $row[0];

//2. 찾은 thread 값부터 10개의 글을 가져옴
$query = "SELECT * FROM $board WHERE thread <= '$start_thread' ORDER
         BY thread DESC LIMIT 10";
$result = mysql_query($query, $conn);
```

실제로 게시판에 적용하면 처리 시간의 차이가 있을 수 있습니다. 왜냐하면 MySQL 콘솔에서는 PHP와 MySQL 간의 통신이 필요 없기 때문입니다. 그래서 실제로 게시판에 적용한 후 속도 차이가 얼마나 나는지 확인해 보도록 하겠습니다.

이에 앞서 PHP에서 처리 시간 측정을 위한 코드를 먼저 알아봅시다.

```
$st = explode(" ", microtime());

..... 시간을 측정할 코드 부분 .....

$et = explode(" ", microtime());
echo $et[1]-$st[1]+$et[0]-$st[0];
```

시간 함수인 microtime() 함수와 문자열 처리 함수인 explode() 함수를 사용하여 소스 코드의 처리 시간을 측정할 수 있습니다. 알고리즘은 다음과 같이 매우 단순합니다.

① 소스 코드 시작 전 시간을 저장한다.

② 소스 코드를 실행하고 처리한다.

③ 소스 코드의 처리가 완료되면 완료 시간을 저장한다.

④ 완료된 시간과 시작 시간의 차이를 구한다.

microtime() 함수는 이미 앞에서 배웠듯이 현재 시간을 UNIX_TIMESTAMP 값으로 출력하되 마

이크로 단위의 초까지 출력하는 함수입니다. 그래서 다음과 같이 띄어쓰기를 기준으로 시간이 분리되어 있습니다.

```
0.42191500 1192524101
```

앞의 수는 마이크로 단위의 시간이고 뒤의 수는 1970년 1월 1일부터 누적된 초 시간입니다. 이처럼 microtime() 함수가 문자열로 구성되어 있기 때문에 처리를 하기 위해서는 띄어쓰기를 기준으로 분리해야 합니다. 그래서 explode() 함수를 사용했으며 explode() 함수의 문자열 분리 기능을 이용해 띄어쓰기를 기준으로 두 개로 분리해서 배열에 저장합니다.

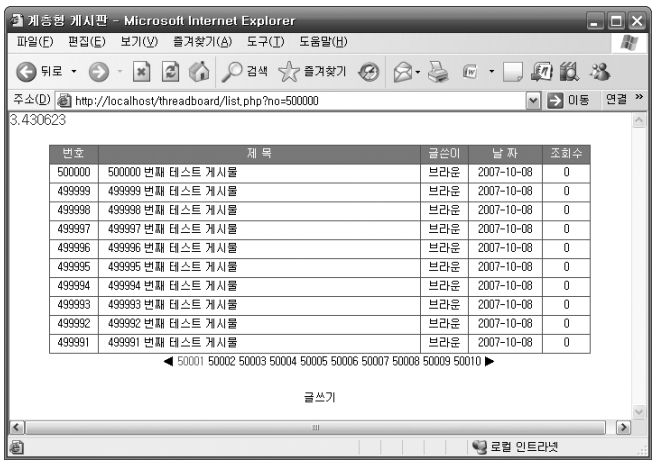
처리 시간은 "완료 시간 – 처리 시간"입니다.

당연한 것이 완료 시간이 시간적으로 더 뒤에 일어났기 때문에 시작 시간보다 값이 더 클 수밖에 없습니다. 따라서 큰 값에서 작은 값을 빼서 양수로 된 시간 차이를 구할 수 있습니다. 그런데 앞서 유닉스 타임 스탬프 값과 마이크로 단위의 초로 분리되어 있기 때문에 각각 차이를 구해야 합니다. 그래서 유닉스 타임 스탬프는 유닉스 타임 스탬프끼리, 마이크로초는 마이크로초끼리 차이를 구합니다. 마이크로초는 1초를 기준으로 소수점으로 표시되기 때문에 계산된 결과를 더해주면 우리가 원하는 형태의 시간을 구할 수 있습니다.

앞서 작성한 새로운 쿼리 전 후에 처리 시간 측정 소스를 다음과 같이 끼워 넣습니다.

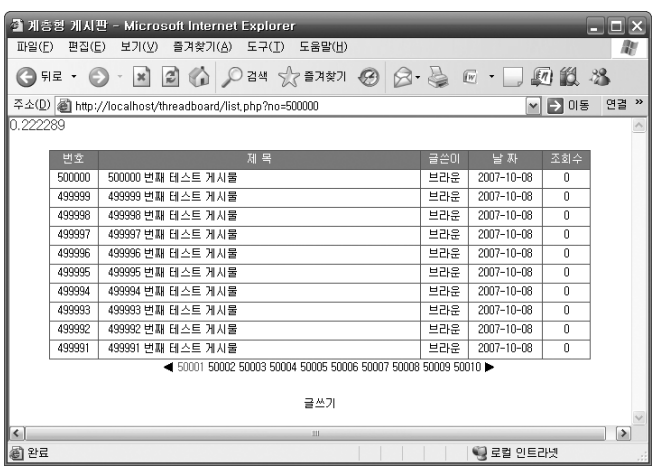
```
$st =  explode(" ", microtime());
//1. 글 목록의 첫 번째 글 찾기
$query = "SELECT thread FROM $board ORDER BY thread DESC LIMIT $no, 1";
$result = mysql_query($query, $conn);
$row = mysql_fetch_row($result);
$start_thread = $row[0];

//2. 찾은 thread 값부터 $page_size만큼의 글을 가져옴
$query = "SELECT * FROM $board WHERE thread <= '$start_thread' ORDER
BY thread DESC LIMIT $page_size";
$result = mysql_query($query, $conn);

$et =  explode(" ",microtime());
echo ($et[1]-$st[1] + $et[0]-$st[0]);
```

이제 새로 작성한 쿼리가 게시판에 적용되었을 때 얼마 만큼의 성능을 내는지 확인해 보겠습니다.



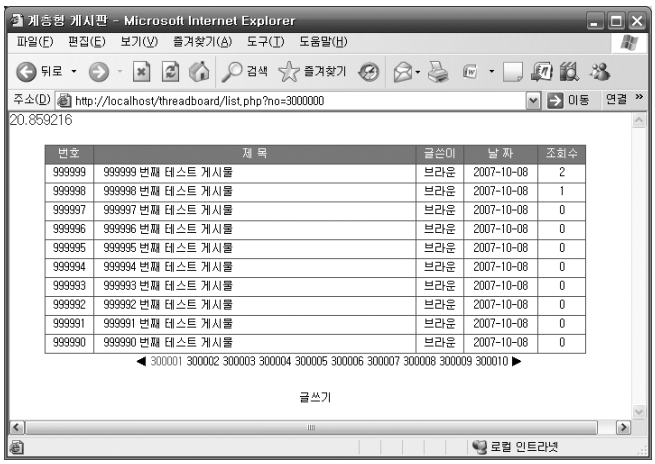
[그림 13-13] 쿼리를 분할하지 않은 상태의 성능

기존에 쿼리를 나누지 않은 상태에서 테스트한 결과 50만 번째 글의 목록을 보는 데 3.4초가 걸렸습니다. 쿼리를 두 개로 나누어 구현했더니 다음과 같은 결과를 얻을 수 있었습니다.



[그림 13-14] 쿼리를 분할한 후의 성능

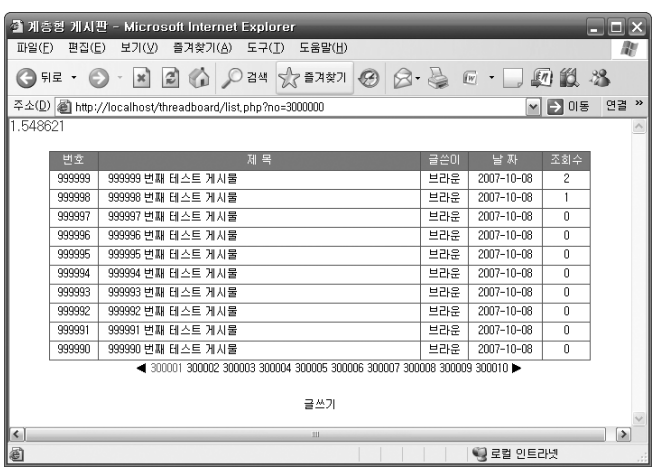
글 목록의 첫 번째 글을 찾아서 출력하는 방법을 사용한 결과 50만 번째 글 목록을 표시하는데 고작 0.22초밖에 걸리지 않았습니다. 레코드의 수에 따른 시간 변화를 알아보려고 임시로 300만 개의 레코드를 더 추가해 보았습니다. 총 400만 개의 레코드일 때 기존 방식으로 구현한 게시판의 속도를 측정하면 다음과 같습니다.



[그림 13-15] 400만 개의 글에서 분할하지 않은 쿼리의 성능

100만 건에서 3.45초가 걸린 것에 반해 400만 건에서 300만 번째 글 목록을 검색한 경우 20.85초가 걸렸습니다. 레코드의 수가 4배로 늘어난 반면에 처리 시간은 무려 6배나 느려졌습니다. 이처럼 레코드가 1000만 개 정도로 늘어나면 글 목록을 출력하는데 30초 이상이 걸리게 되어 30초 실행 시간 초과로 에러가 나게 됩니다.

그에 반해 새로운 방식을 적용한 경우의 결과는 다음과 같습니다.



[그림 13-16] 400만 개의 글에서 쿼리를 분할하였을 때의 성능

글 목록의 첫 번째 글을 찾는 방법의 경우 1.54초로 기존의 20초에 비해 13배나 빠른 것을 알 수 있습니다. 이처럼 이 방식을 쓰는 경우 게시물이 많이 늘어나더라도 시간 변화가 크지 않을 것으로 예상할 수 있습니다.

여기서 잠깐

MySQL 4.1 버전부터는 서브 쿼리(subquery)를 지원합니다. 서브 쿼리는 말 그대로 쿼리 안에 또 쿼리를 사용하는 것으로 경우에 따라 매우 유용하게 사용할 수 있습니다. 물론 쿼리 안에 들어가는 서브 쿼리에는 약간의 제약이 있어서 생각하는 대로 마음껏 사용할 수는 없습니다. 하지만 간단한 쿼리는 대부분 사용이 가능하므로 서브 쿼리를 잘 익혀두면 보다 간단하게 쿼리를 작성할 수 있습니다.

우리는 글 목록 출력의 성능 향상을 위해서 두 개의 쿼리를 사용했습니다. 그런데 서브 쿼리를 사용하면 이 두 개의 쿼리를 하나로 합칠 수 있습니다.

그럼 무엇을 기본 쿼리로 선택하고 무엇을 서브 쿼리로 선택해야 할까요?

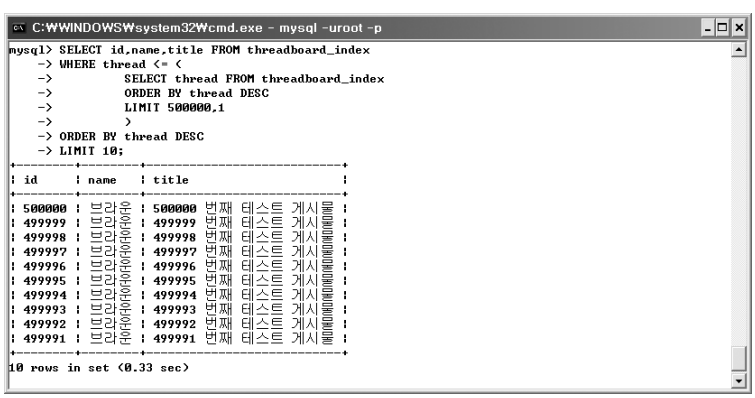
```
SELECT thread FROM $board ORDER BY thread DESC LIMIT $no, 1
SELECT * FROM $board WHERE thread <= '$start_thread' ORDER BY thread
  DESC LIMIT $page_size
```

두 개의 쿼리를 잘 살펴보면 첫 번째 쿼리는 단지 $start_thread 값을 알아내기 위한 목적임을 알 수 있습니다. 단지 두 번째 쿼리의 제약 조건을 지정하는 데 사용되는 쿼리인 것입니다. 이제 어떤 것이 서브 쿼리가 되고 어떤 것이 메인 쿼리가 될 것인지 아시겠습니까?

두 번째 쿼리를 메인으로 하고 첫 번째 쿼리를 두 번째 쿼리 속에 집어넣을 것입니다. 아주 쉬우니까 어떻게 합치는 지 잘 살펴보기 바랍니다.

```
SELECT * FROM $board
WHERE thread <= ( SELECT thread FROM $board ORDER BY thread DESC LIMIT
                  $no, 1 ) ORDER BY thread DESC
LIMIT $page_size;
```

생각보다 무척 쉽죠? 그저 변수 자리에 쿼리를 그대로 집어넣었을 뿐입니다. 이 쿼리가 제대로 동작하는지 콘솔에서 테스트를 해보겠습니다.



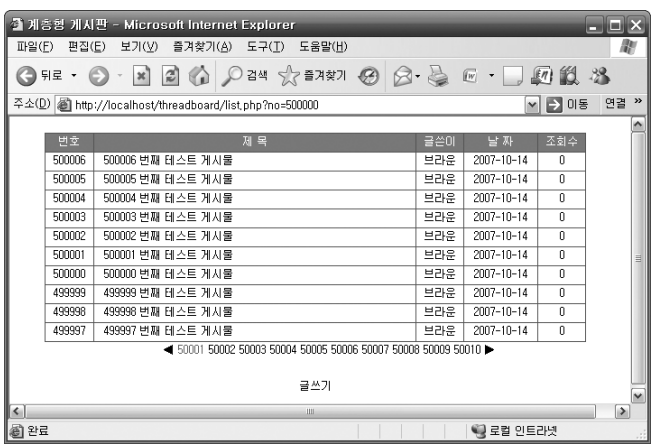
[그림 13-17] 서브 쿼리를 이용한 결과

두 개의 쿼리를 사용할 때와 비교해 보면 성능은 그다지 차이 나지 않는 것을 알 수 있습니다. 그러나 쿼리를 하나만 사용하면 두 번에 나누어 데이터베이스에 쿼리를 전송하고 그 결과를 수신받는 통신 부하를 줄일 수 있기 때문에 더 나은 방법이라 생각됩니다.

실제로 게시판에 코드를 적용하기 위해서는 글 목록을 가져오는 부분을 다음과 같이 수정하면 됩니다.

```
$query = "
  SELECT * FROM $board
  WHERE thread >= (
       SELECT thread FROM $board
       ORDER BY thread
       LIMIT $no,1
       )
  ORDER BY thread
  LIMIT $page_size;
  ";
$result = mysql_query($query, $conn);
```

수정된 소스를 적용한 게시판을 테스트한 결과는 다음과 같습니다.



[그림 13-18] 수정된 계층형 게시판

몇 가지 속도 개선을 통해서 100만 건 정도까지는 무난하게 사용할 수 있을만한 게시판을 만들었습니다. 첫 페이지부터 마지막 페이지까지 그다지 느리다는 느낌 없이 사용할 수 있을 만큼 속도가 빨라졌습니다. 이 책에서는 더 이상의 속도 개선을 다루지 않습니다. 그 이유는 지면을 많이 할당하여도 개선되는 사항은 미약하기 때문입니다. 하지만 제가 생각지 못한 부분에서 충분히 개선 사항이 있을 수 있습니다. 그러니 만약 이 게시판을 사용하면서 느리다고 느껴지는 부분이 있다면 왜 느려지는지 원인을 파악하여 새로운 알고리즘을 적용하고 더 나은 게시판을 만들어 보기 바랍니다.

여기서 잠깐

이 부분을 집필하면서 MySQL의 성능이 놀랍도록 많이 좋아진 것을 체감할 수 있었습니다. 캐시 기법이 적용되어 유사한 쿼리를 반복해서 던지는 경우 매우 빠르게 응답하는 것에 놀라움을 감출 수 없었습니다. 놀라움과는 반대로 매번 달라지는 처리 시간 때문에 정리하는 것을 괴롭게 만들기도 했습니다. 가볍고 빠른 데이터베이스로 시작한 MySQL이 어느덧 안정기인 4.x 버전을 지나서 이제 어엿한 데이터베이스로 모습을 갖춰가고 있는 모습이 어떻게 보면 대견스러워 보이기까지 합니다. 보다 멋진 데이터베이스로 변모해가길 바라면서 MySQL 파이팅입니다욧~!

Section 02

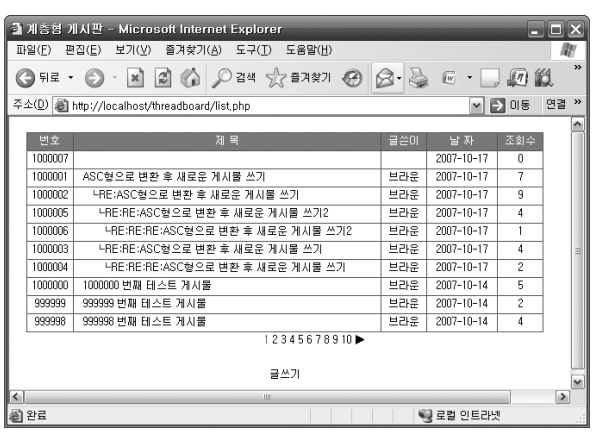
게시판 테러 방지

이제 게시판의 속도는 비교적 쓸만한 수준으로 업그레이드 되었으니 게시판의 잠재적인 문제점을 하나씩 고쳐나가기로 합시다. 앞서 누누이 언급하였듯이 지금까지 만들어온 게시판은 기본적인 게시판의 기능에 중점을 두었습니다. 그래서 많은 문제점을 잠재적으로 내재하고 있습니다. 때문에 앞서 게시판을 사용하지 말라고 말했던 것입니다.

이 장에서는 게시판을 사용하면서 자주 일어나는 문제점과 고의적으로 게시판을 초토화시키는 여러 가지 테러 행위를 막는 법을 다루도록 하겠습니다.

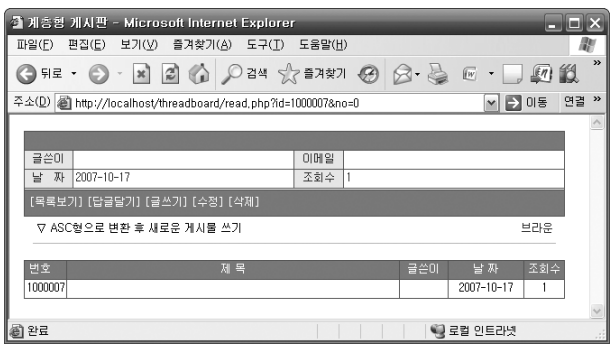
글 입력 시 필수 항목 확인

지금까지 만들어온 계층형 게시판의 가장 큰 문제점은 바로 이것입니다.



[그림 13-19] 빈 제목의 글 문제

1000007번 게시물을 보면 제목도 글쓴이도 모두 빈칸입니다. 실제로 글을 읽어보려면 제목이 없어서 클릭도 되지 않습니다. 다행히 글 번호를 통해서 글 읽기가 가능한데 막상 글을 읽어보면 다음과 같습니다.



[그림 13-20] 내용이 없는 글

현재 게시판에는 이처럼 글 쓰기를 할 때 아무것도 입력하지 않고도 글을 등록할 수 있는 문제가 있습니다. 이렇게 빈칸으로 글이 등록되는 것을 방지하고자 현존하는 대부분의 게시판이 글을 등록할 때 필수 항목을 기입하지 않으면 글이 등록되지 않게끔 처리를 하고 있습니다. 어떻게 필수 항목을 확인할 수 있을까요?

일반적으로 글을 쓸 때 필수 항목을 확인하는 방법은 두 가지입니다.

① 자바스크립트를 이용한 확인

② PHP 코드를 이용한 확인

각각의 방법에 대해 자세히 알아봅시다.

자바스크립트를 이용한 필수 항목 확인

자바스크립트를 이용해서 필수 항목을 확인하는 방법은 글 입력 페이지에서 바로바로 확인할 수 있다는 장점이 있습니다. 심지어는 어느 항목이 어떻게 잘못되었는지도 곧바로 알 수 있습니다. 페이지를 옮기지 않고 현재 페이지에서 확인할 수 있다는 것은 매우 큰 장점이 아닐 수 없습니다. 어떻게 확인할 수 있는지 알아볼까요?

웹 프로그래머에 입문하는 분이라면 HTML과 약간의 자바스크립트 지식이 있을 것으로 생각합니다. 그렇다고 자바스크립트를 전혀 모른다고 해서 이 부분을 이해하지 못한다거나 하는 것은 아니니 너무 걱정하지 마시기 바랍니다. 자바스크립트에 대한 지루한 소개는 건너뛰기로 하고 바로 설명으로 넘어가겠습니다.

자바스크립트에는 폼 정보를 읽어 올 수 있는 객체가 존재합니다.

```
document.forms
```

간단히 말씀드리면 현재 웹 페이지 문서 정보를 갖는 document 객체에 존재하는 여러 가지 하위 객체 중 하나입니다. 배열로 되어 있어서 한 페이지 내에 여러 개의 폼이 존재하는 경우 다음과 같이 배열로 접근할 수 있습니다.

```
document.forms[0]
document.forms[1]
document.forms[2]
```

만약 폼에 이름을 등록해두었다면 인덱스 대신에 폼의 이름으로도 접근할 수 있습니다.

```
document.forms['form1']
document.forms['form2']
document.forms['form3']
```

폼에 이름이 등록된 경우 위와 같은 방식 이외에 아래와 같이 간략하게 접근할 수도 있습니다.

```
document.form1
document.form2
document.form3
```

더 간단하게는 document를 사용하지 않고 직접 폼 이름을 통해서 접근할 수 있습니다.

```
form1
form2
form3
```

또한 폼 안에는 텍스트 상자, 목록 상자, 라디오 단추, 입력 영역 등과 같은 여러 가지 폼 컨트롤이 있는데 이들 각각의 폼 컨트롤이 다시 폼 객체의 하위 객체를 이룹니다. 이 하위 객체들도 위와 마찬가지로 다음과 같이 접근할 수 있습니다.

```
폼이름.폼컨트롤이름
```

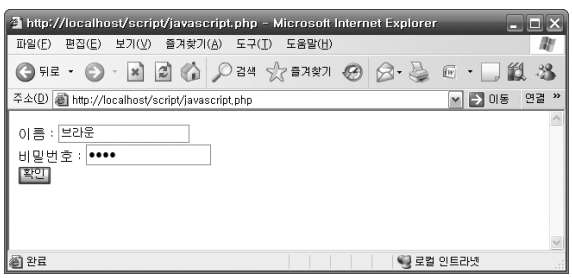
바로 이것을 이용하여 사용자가 글을 입력했는지 안 했는지를 확인할 수 있습니다.

예를 하나 들어 보겠습니다.

```
<form name="test">
  이름 : <input name="name" type="text">
  <BR>
  비밀번호 : <input name="pass" type="password">
```

```
  <BR>
  <input type=submit value="확인">
</form>

<script>
  test.name.value="브라운";
  test.pass.value="1234";
</script>
```

위의 소스를 HTML 문서로 저장한 다음 웹 브라우저로 실행해보면 다음과 같습니다.



[그림 13-21] 자바스크립트를 이용한 폼 값 설정

분명히 아무런 값을 입력하지 않았는데 자동으로 이름과 비밀번호가 입력되어 있습니다. 이것은 위 소스 코드 중 아래의 자바스크립트 부분 때문입니다. 자바스크립트 부분을 자세히 살펴보겠습니다.

```
<script>
  test.name.value="브라운";
  test.pass.value="1234";
</script>
```

test는 test라고 이름 지어진 폼 객체를 의미하는 것이 분명하고 name과 pass는 텍스트 상자와 비밀번호 상자의 이름이 분명합니다. 그래서 name.value는 name이라는 이름의 폼 컨트롤의 입력된 값을 의미하게 됩니다. 그런데 자바스크립트를 이용하여 폼 컨트롤의 입력된 값에 '브라운'이라고 입력을 해두었으니 웹 페이지가 뜨자마자 자동으로 폼 컨트롤에 글자가 뜨게 되는 것입니다.

우리는 게시판에서 값이 입력되어 있는지를 확인해야 하니 이를 이용해서 바꾸어보면 다음과 같습니다.

```
<form name="test" action="a.html">
  이름 : <input name="name" type="text">
  <BR>
  비밀번호 : <input name="pass" type="password">
  <BR>
  <input type=submit value="확인">
</form>
```

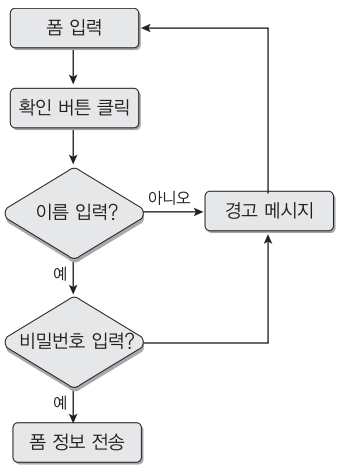
```
<script>
  if (!test.name.value)
  {
    alert("이름을 입력하세요.");
  }
  if (!test.pass.value)
  {
    alert("비밀번호를 입력하세요.");
  }
</script>
```

이름과 비밀번호가 입력되지 않으면 경고 메시지를 출력하게 합니다. 여기에 사용된 alert 함수는 자바스크립트에서 경고 메시지를 출력하는 함수입니다. 파라미터로 넘어온 값을 경고창으로 보여줍니다.

[그림 13-22] 폼을 입력하지 않은 경우의 경고 메시지

위 소스를 HTML 문서로 저장하여 실행하면 위와 같은 경고 메시지를 보게 될 것입니다. 이와 더불어 비밀번호를 입력하라는 메시지도 같이 볼 수 있습니다. 그런데 웹 페이지를 불러오자마자 이렇게 에러 메시지를 띄우니 이름과 비밀번호를 확인하는 의미가 없습니다.

그래서 다음의 순서도와 같이 확인 버튼을 클릭하였을 때 이름과 비밀번호를 입력하였는지 확인하도록 바꾸어봅시다.



[그림 13-23] 폼의 입력 값 확인 흐름

```
<form name="test" onsubmit="return FormCheck()" action="a.html">
  이름 : <input name="name" type="text">
  <BR>
  비밀번호 : <input name="pass" type="password">
  <BR>
  <input type="submit" value="확인" >
</form>

<script>

  function FormCheck() {
    if (!test.name.value)
    {
      alert("이름을 입력하세요.");
      return false;
    }
    if (!test.pass.value)
    {
      alert("비밀번호를 입력하세요.");
      return false;
    }
  }

</script>
```

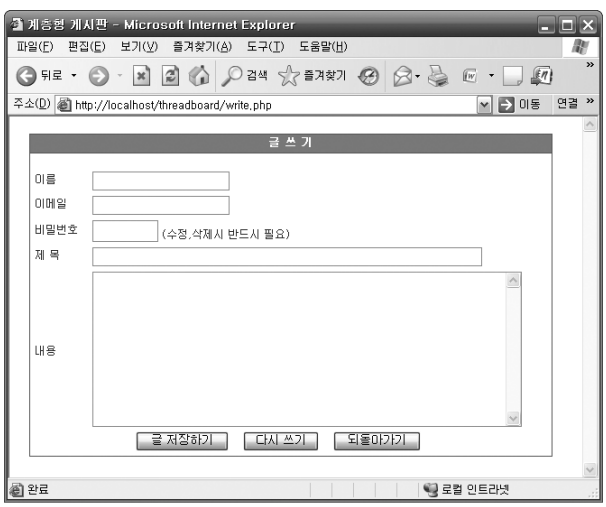
바뀐 부분을 살펴보면 우선 기존의 자바스크립트 부분을 함수로 바꾸었습니다. 함수 내부에는 에러가 발생하면 false를 리턴하도록 지정했습니다. 자바스크립트에서 함수를 만드는 방법이 PHP와 유사하기 때문에 쉽게 이해할 것으로 생각합니다.

그리고 원래 폼 태그에 onsubmit이라는 부분이 추가되었습니다. 이는 이벤트 핸들러로서 submit

이벤트가 발생하면 지정된 함수를 호출하도록 해주는 기능을 합니다. 그래서 확인 버튼을 누르면 자동으로 이벤트 핸들러가 FormCheck 함수를 호출합니다. 그런데 onsubmit="FormCheck()"처럼 호출하지 않고 onsubmit="return FormCheck()"처럼 호출하는 이유는 전자의 경우 에러 메시지의 여부와는 상관 없이 무조건 submit이 이루어지기 때문입니다. 그래서 우선 FormCheck() 함수를 호출하여 true 또는 false를 되돌려받아서 onsubmit="return true" 혹은 onsubmit="return false"가 되게 합니다. 그래서 만약 return true면 action에 지정된 주소로 폼 정보를 전송하고 return false면 submit을 취소합니다.

이제 기본 기능을 하는 스크립트를 작성했습니다. 이를 이용하여 실제로 게시판에서 사용될 글 쓰기 폼의 필수 항목을 확인하는 스크립트를 작성해 보도록 하겠습니다.

우선 글 쓰기를 할 때 어떤 항목을 작성하는지 살펴봅시다.



[그림 13-24] 글 쓰기에서 작성해야 할 항목들

이름, 이메일, 비밀번호, 제목 그리고 글의 내용 이렇게 다섯 항목을 입력받습니다. 작성하는 입력값 중에서 이메일을 제외한 나머지 네 가지 항목은 모두 필수적인 항목입니다. 이를 정리해보면 다음과 같습니다.

입력 항목	이름	필수 여부
이름	name	필수
이메일	email	옵션
비밀번호	pass	필수
제목	title	필수
내용	content	필수

[표 13-6] 글 쓰기에서 항목의 필수 여부

이처럼 4가지 항목에 대해서 필수 항목 검사를 시행한 후 입력된 값이 없을 때 "OOO을 입력하세요."라는 경고 메시지를 보여주면 됩니다.

우선 폼에 이름을 부여하고 submit 이벤트 핸들러를 등록하도록 합니다.

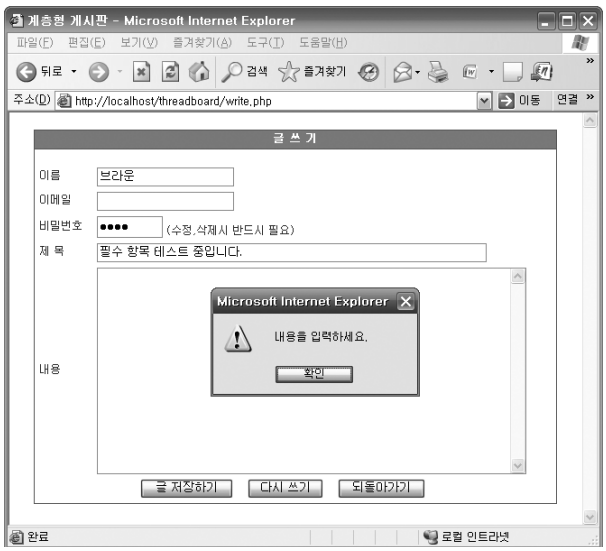
```
<form action=insert.php method=post name=fm onsubmit="return
FormCheck()">
```

이제 필수 항목 체크 함수를 완성시킵니다.

```
<script>

  function FormCheck() {
    if (!fm.name.value)
    {
      alert("이름을 입력하세요.");
      return false;
    }
    if (!fm.pass.value)
    {
      alert("비밀번호를 입력하세요.");
      return false;
    }
    if (!fm.title.value)
    {
      alert("제목을 입력하세요.");
      return false;
    }
    if (!fm.content.value)
    {
      alert("내용을 입력하세요.");
      return false;
    }
  }

</script>
```

단순히 항목을 더 추가하기만 하면 완성됩니다. 이 스크립트를 write.php 파일에 삽입합니다. 삽입하는 위치는 어느 곳이나 가능합니다. 하지만 추후 수정하게 될지도 모르기 때문에 보기 좋게 〈HEAD〉 태그 사이에 넣어둡니다.

수정된 글 쓰기 폼을 테스트 해봅시다.



[그림 13-25] 수정된 글 쓰기 폼 페이지

위의 테스트 결과와 같이 필수 항목을 입력하지 않으면 경고 메시지를 출력하고 제대로 기입할 때까지 글 등록이 되지 않습니다. 그런데 사용하면서 조금 불편한 게 느껴집니다. 이름을 입력하지 않았다면 마우스로 이름 입력 텍스트 상자를 클릭해야 하고 비밀번호를 입력하지 않았다면 비밀번호 입력 텍스트 상자를 클릭한 후 입력해야 한다는 것입니다. 이름을 입력하지 않았다면 이름 칸에 커서가 깜빡이고 있으면 정말 편할텐 데 말입니다.

그래서 필수 항목을 체크하고 만약 필수 항목이 비어있는 경우에는 해당 항목으로 커서를 옮겨주는 소스를 추가해 보도록 합시다.

자바스크립트에서 커서를 옮기는 방법은 focus()라는 함수를 사용하는 것입니다. 다음과 같은 방법으로 폼 컨트롤에 포커스를 줄 수 있습니다.

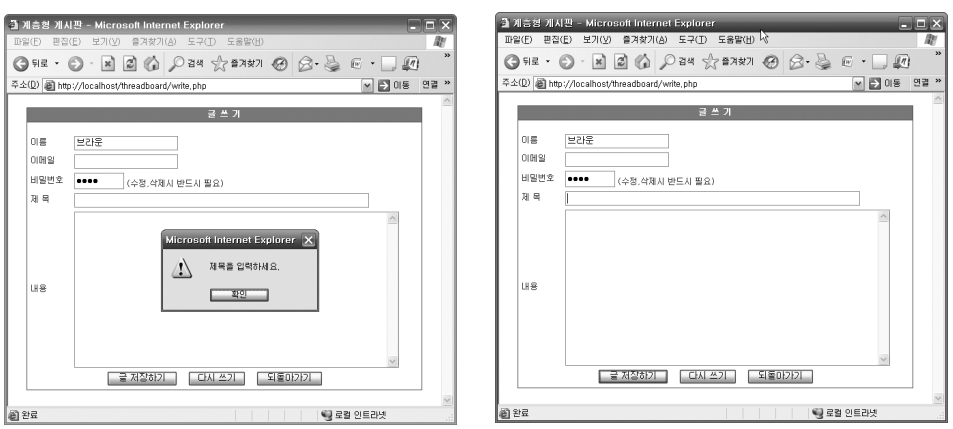
```
폼이름.폼컨트롤이름.focus();
```

따라서 이것을 이용해서 모든 필수 항목에 적용하면 다음과 같은 자바스크립트를 얻게 됩니다.

```
<script>

  function FormCheck() {
    if (!fm.name.value)
    {
      alert("이름을 입력하세요.");
      fm.name.focus();
      return false;
```

```
    }
    if (!fm.pass.value)
    {
      alert("비밀번호를 입력하세요.");
      fm.pass.focus();
      return false;
    }
    if (!fm.title.value)
    {
      alert("제목을 입력하세요.");
      fm.title.focus();
  return false;
    }
    if (!fm.content.value)
    {
      alert("내용을 입력하세요.");
      fm.content.focus();
  return false;
    }
  }

</script>
```

실제로 확인해보면 다음과 같습니다.



[그림 13-26] 입력하지 않은 항목으로의 커서 이동

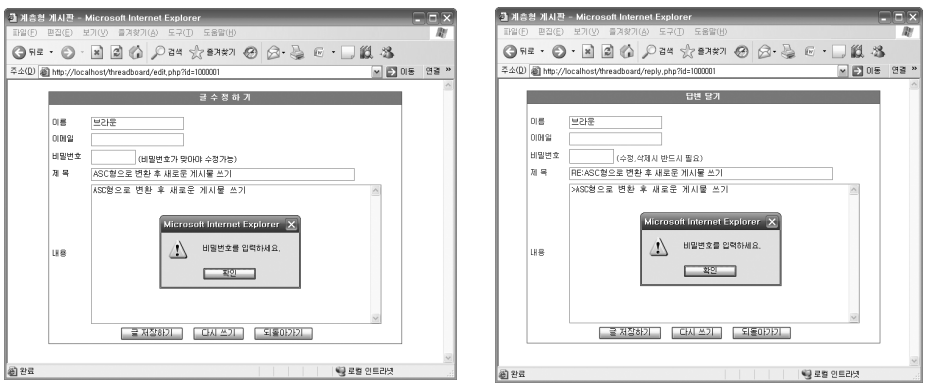
이처럼 제목을 입력하지 않으면 경고 메시지가 뜨면서 제목 칸에 커서가 옮겨진 것을 확인할 수 있습니다. 이제 클릭 없이 바로 글을 작성할 수 있어서 매우 편리해졌습니다.

이와 마찬가지로 답변 달기와 글 수정 페이지를 수정해 줍니다. 수정하는 절차를 정리해보면 다음과 같습니다.

① 폼에 fm이라는 이름을 등록하고 submit 이벤트 핸들러를 등록한다.

② 필수 항목 체크 자바스크립트를 추가한다.

위와 같이 폼 이름을 fm으로 동일하게 부여하면 위에서 작성한 자바스크립트를 수정 없이 그대로 사용할 수 있습니다.



[그림 13-27] 재사용을 통한 폼 입력 검증 추가

글 수정하기와 답변 달기 모두 제대로 동작하는 것을 확인할 수 있습니다. 마지막으로 글을 삭제할 때는 비밀번호 폼만 존재하므로 FormCheck() 함수에서 비밀번호를 제외한 나머지 부분을 지우고 predel.php 파일을 위와 같은 방법으로 수정합니다.

[그림 13-28] 글 삭제에서의 비밀번호 검증

모든 필수 항목 체크가 끝났습니다. 이제는 값을 입력하지 않고는 글을 제대로 입력할 수 없게 되었습니다. 그러나 자바스크립트를 이용하여 필수 항목을 체크하는 방법은 한 가지 문제가 있습니다. 글 작성 페이지를 거치지 않고 직접 글 저장 페이지로 접근하는 경우 확인이 불가능하기 때문입니다. 그래서 결국은 PHP를 이용하여 다시 검증할 수밖에 없습니다.

PHP를 이용한 필수 항목 확인

PHP를 이용하여 필수 항목을 체크해 보겠습니다. 앞서 자바스크립트를 이용해서 필수 항목을 체크한 경우에는 악의적인 사용자에 의해서 쉽게 무력화될 수 있습니다.

지금은 회사의 내부사정으로 사라진 오르지오(www.orgio.net)라는 웹 메일 사이트가 있었습니다. 그 당시 다른 메일 서비스에 비해서 많은 공간을 제공하고 거기다 SMTP와 POP3 서비스를 제공하고 있어서 아웃룩과 같은 프로그램에서 메일을 주고 받을 수 있는 장점이 있었습니다. 이러한 장점 때문에 저도 메일에 가입하려고 했습니다. 그런데 아이디를 반드시 4자 이상으로 등록해야 한다는 조건이 있었습니다. 특이한 메일 주소를 갖고 싶었던 저는 매우 아쉬울 수밖에 없었습니다. 그래서 가입 폼 페이지의 HTML 소스 코드를 살펴보았더니 자바스크립트로 아이디의 길이를 제한하고 있는 것을 쉽게 알 수 있었습니다. 그래서 혹시나 하는 마음에 HTML 소스를 본떠서 가입 폼을 새롭게 만들었습니다. 물론 자바스크립트 검증을 하지 않는 순수한 가입 폼 페이지입니다. 한 자짜리 아이디를 넣고 가입하기를 눌렀더니 아이디가 그냥 등록되어 버렸습니다. 지금은 사용하지 않지만 그때 만들어진 메일 주소가 바로 _@orgio.net 입니다. 사람들에게 메일 주소를 알려주면 누구나 한 번쯤 되물어 보는 특이한 메일 주소였습니다.

이처럼 자바스크립트를 너무 맹신하면 안 됩니다. 자바스크립트는 단지 일차적인 검증 수단에 불과합니다. 웹 브라우저의 설정을 약간 바꾸면 자바스크립트를 실행할 것인지 아니면 무시할 것인지 매번 확인할 수도 있습니다. 이를 이용하여 가입 폼을 작성할 때 자바스크립트를 사용하지 않음으로 설정해 둔다면 아무리 확인 버튼을 누르더라도 필수 항목 체크를 하지 않을 것입니다.

하지만 PHP로 필수 항목을 검증하면 매번 검증 페이지로 이동하여 확인을 하고 에러가 있으면 다시 페이지로 돌아오게 되는 단점이 있습니다. 이 때문에 대부분은 편의를 위해서 자바스크립트로 1차 검증을 하고 PHP로 다시 검증을 하는 것입니다.

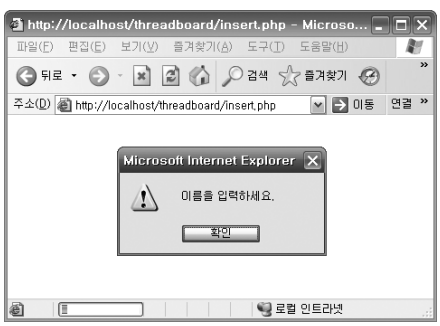
그럼 글 저장 페이지에 다음과 같은 필수 항목 확인 코드를 삽입합니다.

```
//입력값 검증
if (!$_POST[name])
{
  echo "<script> alert('이름을 입력하세요.'); history.back(); </script>";
  exit;
}
if (!$_POST[pass])
{
  echo "<script> alert('비밀번호를 입력하세요.'); history.back(); </script>";
  exit;
}
if (!$_POST[title])
{
  echo "<script> alert('제목을 입력하세요.'); history.back(); </script>";
```

```
    exit;
  }
  if (!$_POST[content])
  {
    echo "<script> alert('내용을 입력하세요.'); history.back(); </script>";
    exit;
  }
```

자바스크립트를 통해서 경고 메시지를 출력하고 이전 페이지로 되돌아가게 합니다. 그런데 여기서 주의할 것은 앞에서도 잠깐 언급한 적이 있지만 경고 메시지를 출력한 후에 반드시 exit를 통해서 PHP스크립트를 종료해주어야 한다는 것입니다. 일반적으로 자바스크립트를 실행하고 history.back()을 만나면 페이지가 이동하기 때문에 그 뒤의 PHP 소스 코드가 처리되지 않을 거라고 생각하지만 그렇지 않습니다. 따라서 반드시 PHP 코드의 실행을 멈추게 해야 합니다. 그렇지 않으면 뒷부분의 PHP 코드가 실행되어 글이 등록됩니다.

이를 실제 글 쓰기에 적용해보면 다음과 같습니다.



[그림 13-29] PHP 소스를 이용한 폼 입력 값 검증

이 소스를 다시 글 수정 시 저장하는 페이지와 답변을 저장하는 페이지에 동일하게 삽입합니다. 또한 비밀번호를 확인하는 부분만 떼어내어 글을 삭제하는 페이지에도 삽입합니다. 소스에 삽입하는 위치는 페이지의 제일 상단인 데이터베이스 연결하는 부분 앞에 삽입합니다. 입력 값이 제대로 넘어오지 않았으면 굳이 데이터베이스에 연결할 이유가 없기 때문입니다.

이제 자바스크립트를 비롯하여 PHP로도 필수 항목의 값을 확인했기 때문에 보다 안심하고 게시판을 운영할 수 있게 되었습니다.

에러 메시지 처리

프로그래머는 항상 프로그래머 자신의 의지대로 프로그램을 개발합니다. 그래서 대부분의 프로그래머는 자신이 의도한 올바른 행위만을 하기 때문에 에러를 일으킬 가능성이 매우 적습니다. 그러

나 일반 사용자가 사용하는 경우에는 다릅니다. 일반 사용자는 프로그램의 원리나 구조에는 관심이 없기 때문에 자신이 생각하는 대로 프로그램을 사용합니다. 그래서 프로그래머가 의도하지 않은 행동을 함으로써 여러 가지 에러를 일으킬 수 있습니다. 대표적인 예가 바로 필수 항목을 채우지 않는 것입니다. 프로그래머는 당연히 값을 입력해야 한다는 것을 알고 있기 때문에 에러가 일어나지 않지만 사용자는 게시판 사용이 서툴러 제목이나 이름을 입력하지 않고 글을 저장해서 에러를 일으킵니다. 또한 시스템의 일시적인 오동작이나 MySQL 서버의 종료 등 시스템 문제로 인한 에러가 발생할 수도 있습니다.

이러한 여러 가지 에러에 대비해 사용자에게 알맞은 메시지를 보여주고 올바르게 프로그램을 사용할 수 있게끔 유도하는 것을 에러 처리(Error Handling)라고 합니다.

우리가 만들어온 게시판은 다음과 같은 에러 발생 요인이 있습니다.

① 사용자의 잘못된 입력으로 인한 에러

② 데이터베이스 관련 에러

첫 번째의 경우 사용자가 필수 항목을 입력하지 않았다든지 사용자가 글을 수정하거나 삭제하기 위해서 입력한 비밀번호가 틀리다든지 하는 이유로 발생합니다. 두 번째의 경우는 데이터베이스에 연결을 시도하는데 데이터베이스가 종료되어 연결이 불가능하다거나 데이터베이스에 쿼리를 전송했는데 쿼리에 문제가 있거나 해서 에러가 발생합니다.

이러한 에러를 처리하기 위해서 대부분 경고 메시지를 출력하고 이전 페이지로 돌아가서 다시 올바르게 시도하게끔 하는 방법을 많이 사용합니다. 그래서 동일한 유형의 에러 처리 구문들이 자주 반복해서 사용됩니다. 이에 매번 에러 처리 구문을 작성하는 번거로움을 없애기 위해서 여러 가지 경우에 모두 사용할 수 있는 공용 메시지 처리 함수를 만들어 보겠습니다.

에러 메시지를 처리하는 방식은 두 가지입니다. 한 가지는 에러 메시지를 출력하고 이전 페이지로 돌아가는 것이고 나머지 한 가지는 에러 메시지만 출력하고 PHP 스크립트의 실행을 종료하는 방법입니다. 첫 번째 경우는 글을 저장하거나 수정하거나 하는 페이지에서 발생하고 두 번째 경우는 글을 입력하는 폼 페이지나 이전으로 돌아갈 페이지가 없는 글 목록 페이지와 같은 곳에서 발생합니다. 그러나 글 입력 폼에서 발생하는 메시지는 그때그때 폼 정보에 따라 수정되어야 하고 함수로 만든다고 더 편리해지지 않으므로 제외하기로 합니다.

이 두 가지 경우를 모두 해결할 수 있는 하나의 함수를 만들어 봅시다.

```
<?
  function ErrorMessage($message, $type = "on")
  {
    echo "<script> alert('$message'); ";
    if ($type == "on") echo " history.back(); ";
```

```
        echo "</script>";
        exit;
    }
?>
```

기본 인수를 사용하여 메시지만 입력되었을 때는 메시지를 출력하고 이전 페이지로 돌아가며, 메시지와 또 하나의 "on"이 아닌 다른 값이 인수로 들어오면 메시지만 출력하고 PHP 스크립트를 종료하게 합니다.

앞으로 이 함수와 더불어 여러 가지 공용 함수들을 여럿 만들게 되므로 미리 library.php 파일에 이 함수를 저장해두고 사용하기로 합니다.

사용자의 잘못된 입력으로 인한 에러 처리

사용자의 잘못된 입력으로 인한 에러는 앞서 말씀드렸듯이 다음과 같이 두 가지입니다.

① 필수 항목 미입력 에러

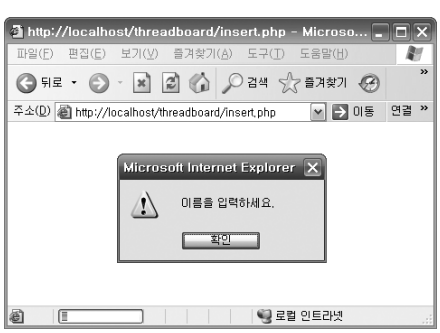
② 비밀번호 오류 에러

우선 필수 항목 미입력 에러 부분을 처리해 보기로 합니다. 이미 앞서 필수 항목 확인 부분은 에러 처리를 해두었기 때문에 앞서 만든 공용 함수를 이용하여 수정하면 됩니다.

글 저장하기를 비롯한 답변 저장 등의 페이지를 아래와 같이 수정합니다.

```
include "library.php";

//입력값 검증
if (!$_POST[name]) ErrorMessage('이름을 입력하세요.');
if (!$_POST[pass]) ErrorMessage('비밀번호를 입력하세요.');
if (!$_POST[title]) ErrorMessage('제목을 입력하세요.');
if (!$_POST[content]) ErrorMessage('내용을 입력하세요.');
```

수정된 부분을 확인해보면 다음과 같습니다.



[그림 13-30] 공용 에러 메시지 함수를 이용한 폼 검증

이제 비밀번호 오류에 따른 에러 처리를 해봅시다.

del.php 파일과 update.php 파일에는 글을 삭제하거나 수정하기 위한 권한을 알아보고자 비밀번호를 확인하는 부분이 있습니다. 만약 비밀번호가 저장된 비밀번호와 다르다면 다음의 소스 코드 때문에 에러 메시지를 출력합니다.

```
echo ("
<script>
  alert('비밀번호가 틀립니다.');
  history.go(-1);
  </script>
");
exit;
```

이 부분을 공용 함수로 바꾸어보면 다음과 같이 한 줄로 정리가 됩니다.

```
ErrorMessage('비밀번호가 틀립니다.');
```

이처럼 del.php와 update.php 파일을 수정해 줍니다. 실제로 ErrorMessage 함수를 사용하려면 반드시 library.php 파일을 인클루드해야 합니다. 그러나 이미 del.php와 update.php에 필수 항목을 체크하기 위해서 이미 인클루드시켰기 때문에 바로 ErrorMessage 함수를 사용할 수 있습니다.

데이터베이스 관련 에러 처리

데이터베이스를 사용하다 보면 데이터베이스가 이상 동작으로 종료되거나 웹 서버의 재부팅으로 데이터베이스 데몬이 실행되지 않는 등의 이유로 접속이 안 될 경우가 있습니다. 이러한 데이터베이스 관련 에러를 어떻게 처리하는지 알아봅시다.

우선 데이터베이스 관련 에러는 다음과 같이 두 가지가 존재합니다.

① 데이터베이스 연결 에러
② 데이터베이스 쿼리 에러

데이터베이스의 이상으로 접속이 안 되는 경우는 자바스크립트를 이용해서 경고 메시지를 출력하지 않고 텍스트로 경고 메시지를 출력하고 PHP 스크립트의 실행을 마치게 합니다. 그 이유는 데이터베이스에 연결이 안 되는 경우에는 게시판 전체의 사용이 불가능하기 때문입니다. 이전 페이지로 돌아가더라도 게시판을 사용할 수 없는 것은 마찬가지니까 경고 문구를 출력하는 것으로 끝내도록 합니다.

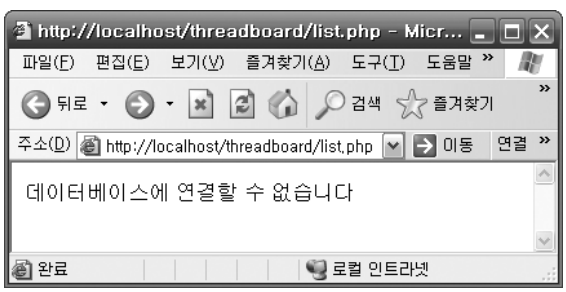
```
$conn=@mysql_connect("localhost","아이디","비밀번호")
  or die('데이터베이스에 연결할 수 없습니다');
@mysql_select_db("mydb_euckr", $conn)
  or die('선택한 데이터베이스를 사용할 수 없습니다.');
```

위의 소스 코드와 같이 우선 mysql_connect 함수와 mysql_select_db 함수에 골뱅이(@) 표시를 추가합니다. 이는 앞서 언급하였듯이 에러 메시지를 출력하지 않게끔 하는 것입니다. 만약 데이터베이스의 문제로 에러가 발생하면 PHP 차원에서 에러 메시지를 출력하지 말고 우리가 지시한 에러 메시지를 출력하게끔 하기 위해서입니다.

```
or die(메시지);
```

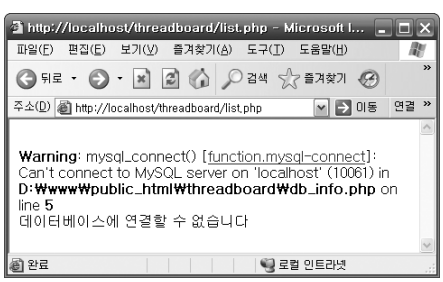
위는 함수가 실행되고 에러가 발생하였을 때 실행되는 구문입니다. die는 실제로 exit와 동일합니다. 단지 관습적으로 데이터베이스에서 에러가 발생하여 프로그램의 실행을 중단하면서 메시지를 출력할 때 die라고 많이 사용합니다. 아마도 die라는 단어적 의미 때문이 아닌가 싶습니다.

실제로 데이터베이스를 종료시킨 후 글 목록 페이지에 접속해보면 다음과 같은 경고 메시지가 출력됩니다.



[그림 13-31] @를 이용한 에러 메시지 처리

만약 골뱅이를 추가하지 않았으면 다음과 같이 자세한 경고 문구가 나오게 되지만 사용자의 입장에서는 알 수 없는 어지러운 메시지일 뿐이므로 골뱅이를 사용하여 경고 문구를 무시하는 방법을 많이 사용합니다.



[그림 13-32] @가 없을 때의 에러 메시지

이제 쿼리를 전송했을 때 발생할 수 있는 에러를 처리해 보도록 합시다.

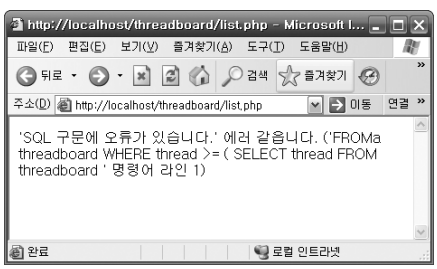
mysql_query 함수를 사용하는 모든 곳에 에러 처리 함수를 사용하여 메시지를 출력해주면 됩니다.

우선 글 목록 페이지를 수정해 봅시다.

글 목록 페이지에는 mysql_query 함수가 두 번 쓰이게 됩니다. 한 번은 데이터베이스에서 글 목록을 가져오는 부분이며 다른 한 번은 글이 총 몇 개가 있는지 카운트하는 데 쓰입니다. 그런데 글 목록 페이지는 에러가 발생해도 글 목록 페이지에 그대로 있어야 합니다. 그래서 이 부분도 데이터베이스 연결과 마찬가지로 경고 메시지를 출력하고 PHP의 실행을 종료하기로 합시다.

```
$result = mysql_query($query, $conn) or die(mysql_error());
$result_count=mysql_query("SELECT count(*) FROM $board",$conn) or
die(mysql_error());
```

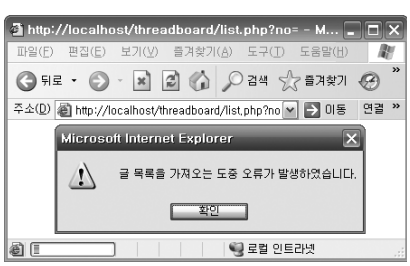
mysql_error() 함수를 이용하여 데이터베이스 에러를 출력합니다. 쿼리를 일부러 문법에 맞지 않게 작성하고 테스트를 해보았습니다.



[그림 13-33] mysql_error() 함수를 이용한 에러 메시지 출력

그러나 이는 프로그래머 측면에서 에러 메시지를 출력한 것으로 실제 게시판을 제작해서 사용할 때는 다음과 같이 변경하여 처리합니다.

```
$result = mysql_query($query, $conn)
or ErrorMessage('글 목록을 가져오는 도중 오류가 발생하였습니다.', false);

$result_count=mysql_query("SELECT count(*) FROM $board",$conn)
or ErrorMessage(' 글 목록을 가져오는 도중 오류가 발생하였습니다.', false);
```



[그림 13-34] 쿼리에 대한 에러 메시지 처리

글 목록을 모두 처리하였으니 글 저장하기 등을 처리해 봅시다.

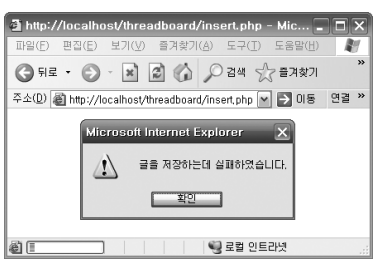
```
$result=mysql_query($query, $conn)
or ErrorMessage(에러 메시지);
```

대부분의 페이지는 mysql_query 함수를 사용하는 모든 곳에 위와 같이 ErrorMessage 함수를 사용하는 것으로 에러 메시지를 처리할 수 있습니다. 하지만 각 페이지마다 사용자에게 알려주어야 하는 말이 다르니 이것을 고려해야 합니다.

페이지	파일 이름	에러 메시지
글 저장	insert.php	글을 저장하는 데 실패하였습니다.
답변 글 저장	insert_reply.php	답변 글을 저장하는 데 실패하였습니다.
수정 글 저장	update.php	글을 수정하는 데 실패하였습니다.
글 삭제	del.php	글을 삭제하는 데 실패하였습니다.

[표 13-7] 각 상황에 대한 에러 메시지

위와 같이 에러 메시지를 고려하여 각 파일을 수정해 줍니다.



[그림 13-35] 글 쓰기 반영 도중의 에러 메시지

이처럼 상황을 알리는 메시지를 보여주어 사용자가 어떤 일이 일어났는지를 알게끔 하는 것이 중요합니다. 기존의 경우 에러가 발생하더라도 에러 처리 부분이 존재하지 않아 화면에는 글을 저장하였다는 메시지만 출력될 뿐 글이 제대로 등록되지 않는 경우가 있었습니다. 이처럼 사용자로 하여금 게시판에 어떤 문제가 있다는 것을 알게 해서 몇 번 시도해보다가 계속 동일한 문제가 발생한다면 관리자에게 알려서 문제를 수정하게끔 유도하는 것이 중요합니다.

그러나 개발 도중이나 테스트 중이라면 이런 메시지보다는 개발자에게 이로운 자세한 에러 메시지가 훨씬 이롭습니다. 그래서 앞서도 사용한 mysql_error() 함수를 이용하여 자세한 에러 메시지를 출력해서 오류를 수정할 수 있도록 편의를 제공해주는 것이 더 좋습니다.

게시판 모양 깨짐 방지

홈페이지를 만들어서 게시판을 올리고 사용하다 보면 특히 많은 사람이 오가는 홈페이지일수록 짓

궂은 악동들이 많이 있습니다. 자신의 존재를 알리기 위해서나 아니면 그냥 재미로 게시판의 모양을 일그러뜨리는 사람들인데요. 며칠 밤낮을 고심하며 예쁘게 디자인한 게시판을 깨트려버리면 만든 사람은 얼마나 가슴이 아프겠습니까? 그래서 게시판의 모양을 깨트리는 일반적인 기법 몇 가지를 알아보고 게시판 디자인을 사수해 보겠습니다.

테이블 길이 늘이기 방지

대부분의 홈페이지가 디자인의 이유로 게시판의 가로 크기가 고정되어 있습니다. 이때 고의에서건 고의가 아니건 종종 게시판의 가로 길이를 길게 만들어 버리는 경우가 있습니다. 이런 경우 대부분의 홈페이지에선 전체 디자인까지 엉망이 되어 버립니다.

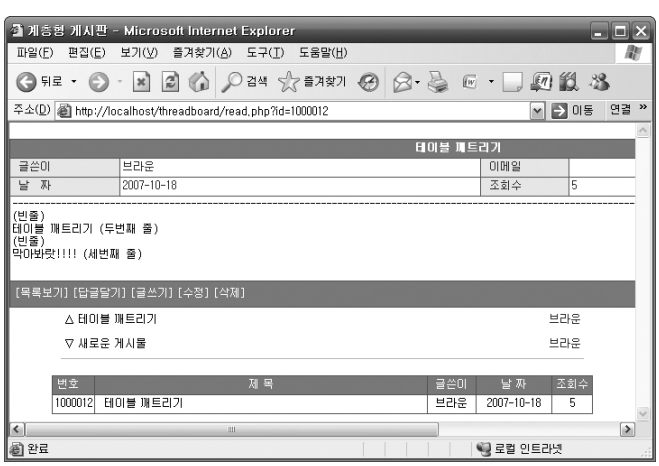
제가 운영하는 ezphp.net 사이트도 오래전 일이지만 한때 이런 일이 있었습니다. 방문자의 소소한 장난에 방구석에서 벽을 보며 끙끙 앓았답니다.

저를 벽과 친구로 만든 범인은 바로 이 녀석입니다.

```
"----------------------------------"
```

범인은 하이픈입니다. 사실 이 녀석만이 범인은 아닙니다. 띄어쓰기를 하지 않는 알파벳도 모두 다 해당됩니다.

테이블은 띄어쓰기가 되어 있으면 띄어쓰기를 기준으로 잘 나누어 자동으로 줄 바꿈을 해줍니다. 그런데 띄어쓰기 없이 길게 연결된 문자는 테이블이 어디서 줄 바꿈을 해야 할지 알지 못하여 테이블의 길이를 벗어나서도 계속 길어지게 됩니다.



[그림 13-36] 하이픈으로 인한 테이블 깨트리기

이처럼 게시판의 가로 길이가 길어지면서 보기 싫게 변해 버립니다. 이를 방지하기 위해서 테이블

(〈TABLE〉)에는 특별한 기능이 있습니다. 바로 테이블 길이보다 긴 문장이 오면 자동으로 다음 줄로 줄 바꿈을 해주는 기능입니다.

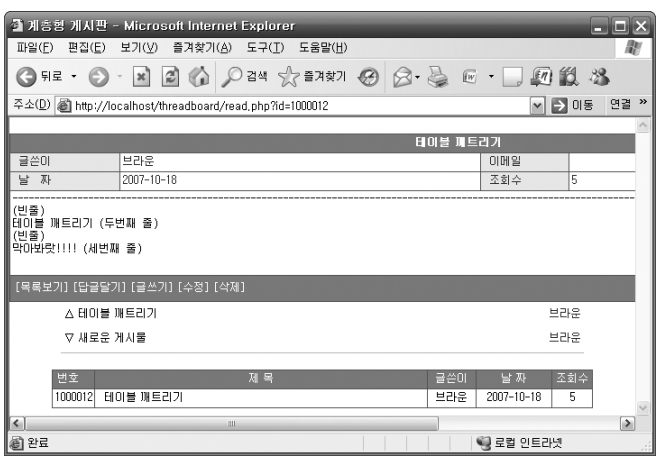
```
<td style="word-break:break-all;">
```

이 스타일을 쓰면 테이블의 크기는 변하지 않고 다음 줄로 자연스레 넘겨주는 소위 자동 줄 바꿈 기능이 적용됩니다. 사용자가 길게 입력할 수 있는 곳에 이 문장을 삽입하면 테이블의 길이가 길어지는 것을 막을 수 있습니다.

대표적으로 글 읽기 페이지에 삽입해 보겠습니다. 제목은 길이의 제한이 있어서 테이블을 깨트리는 경우가 없으므로 넘어가기로 하고 본문 내용만 이 스타일을 적용해 봅시다.

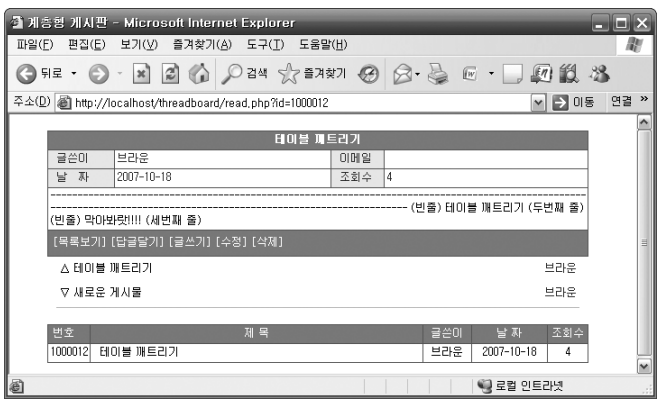
```
<td bgcolor=white colspan=4 style="word-break:break-all;">
<font color=black>
<pre><?=strip_tags($row[content]);?></pre>
</font>
</td>
```

소스를 수정하고 확인해보면 다음과 같습니다.



[그림 13-37] 테이블 스타일을 이용한 결과

이런, 전혀 수정이 안 되었습니다. 분명히 스타일을 적용하였는데 안 되는 것을 보니 제가 거짓말쟁이인 것 같습니다. 하지만 사실은 〈PRE〉〈/PRE〉 태그 때문에 스타일이 적용되지 않은 것입니다. 그 말이 사실인지 확인하기 위해서 〈PRE〉〈/PRE〉 태그를 삭제해보면 다음과 같이 제대로 스타일이 적용되는 것을 알 수 있습니다.

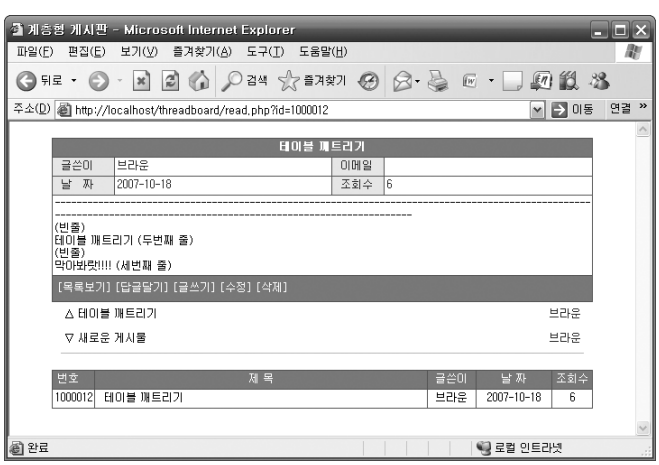


[그림 13-38] 〈PRE〉 태그를 제거한 후의 결과

테이블의 길이를 벗어나지 않게 줄 바꿈이 된 것을 알 수 있습니다. 그런데 문제는 여러 줄로 입력한 글이 한 줄로 나와 버렸습니다. 〈PRE〉 태그에는 입력 시 줄 바꿈이 된 것을 그대로 표현했는데 〈PRE〉 태그를 사용하지 않게 되면서 줄 바꿈이 이루어지지 않은 것입니다. 그래서 이를 별도로 처리해야 합니다.

PHP 함수 중에는 nl2br (New Line To BR의 약자)이라는 함수가 있습니다. nl2br 함수는 줄 바꿈을 〈BR〉 태그로 바꾸는 기능을 합니다. 따라서 다음과 같이 코드를 수정하면 줄 바꿈이 바로 될 것입니다.

```
<td bgcolor=white colspan=4 style="word-break:break-all;">
<font color=black>
<?=nl2br(strip_tags($row[content]));?>
</font>
</td>
```



[그림 13-39] nl2br 함수를 이용한 〈PRE〉 태그 대체 결과

그러나 여기에도 문제가 있습니다. 다음과 같이 글을 등록하는 경우

[그림 13-40] 띄어쓰기가 된 글의 입력

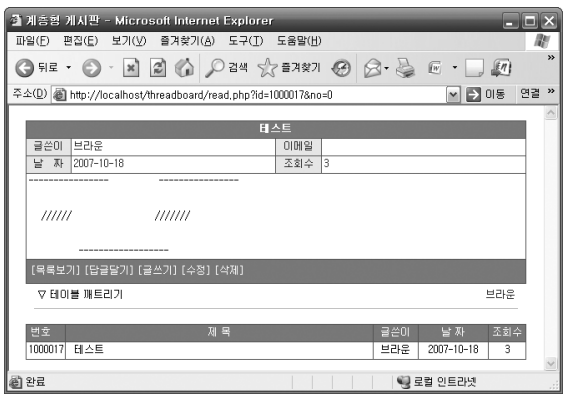
글 읽기를 해보면 입력할 때와 다르게 나타남을 알 수 있습니다.

[그림 13-41] 띄어쓰기가 무시되어 버린 결과

이는 〈PRE〉 태그가 줄 바꿈뿐만이 아니라 띄어쓰기까지 해결했던 것을 〈PRE〉 태그를 없애면서 처리해주지 않았기 때문입니다. 즉, 띄어쓰기 공간을 전부 되살려야 한다는 뜻입니다. 띄어쓰기 공간은 라는 특수기호로 표시할 수 있습니다. 그러면 모든 띄어쓰기 공간을 로 바꾸어 주면 되니까 치환 함수인 str_replace 함수를 사용하도록 합니다. str_replace 함수를 통해서 기존 코드를 수정하면 다음과 같습니다.

```
<td bgcolor=white colspan=4 style="word-break:break-all;">
<font color=black>
<?=nl2br(str_replace(" ", " ", strip_tags($row[content])));?>
</font>
</td>
```

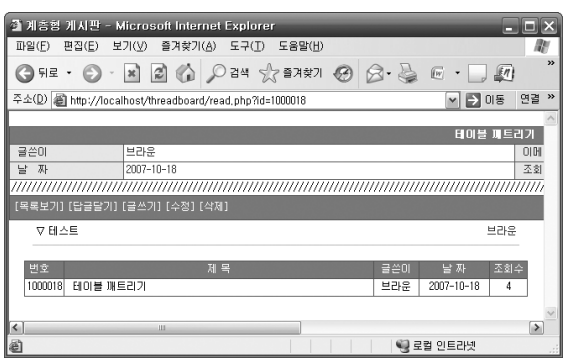
수정된 소스 코드를 확인해보면 다음과 같습니다.



[그림 13-42] str_replace 함수를 이용한 띄어쓰기 반영

띄어쓰기가 제대로 되는 것을 확인할 수 있습니다. 그런데 입력할 때와 약간 모양이 차이가 나는 것은 Stylesheet가 적용되어 글씨 크기와 띄어쓰기 기준 등이 변경되었기 때문입니다.

모든 것이 완료되었다고 기쁨을 만끽하고 싶은 이때, 귀여운 수준이었던 하이픈과 밑줄 친구들과는 달리 테이블 길이 늘이기의 최강자가 나타났으니 테이블 세상은 또 한 번 큰 혼란을 겪게 됩니다.



[그림 13-43] 테이블 스타일이 적용되지 않는 문자열

분명히 스타일을 적용했으나 전혀 반영되지 않고 계속해서 길어져만 가는 저 테이블은 테이블 길이 늘이기의 최강자인 슬래시와 친구들(쉼표, 마침표) 앞에서 너무 무력해졌습니다. 그러나 더욱 강력한 창이 있다면 더욱 강력한 방패도 있는 법.

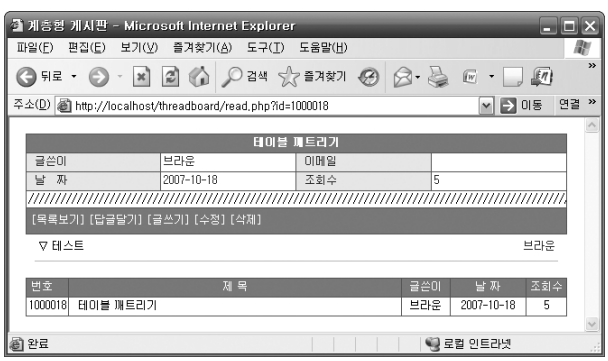
```
<table width=600 border=0 style="table-layout:fixed;">
```

테이블의 길이를 절대 늘어나지 못하게 고정해버리는 스타일을 쓰게 됩니다. 그러나 이 방법은 부득이하게도 줄 바꿈을 해주는 것이 아니라 테이블 크기 이상의 문자에 대해서는 무시해버리는 기능입니다.

글 읽기 페이지의 테이블에 위와 같이 스타일을 적용하면 다음과 같습니다.

```
<table width=600 border=0 cellpadding=2 cellspacing=1 bgcolor=#777777
style="table-layout:fixed;">
```

그 결과는 어떻게 되었을까요?



[그림 13-44] 테이블 사이즈 고정을 통한 테이블 보호

앞서 말씀드린 것처럼 슬래시가 더 많이 있음에도 불구하고 잘려서 출력됩니다. 이 스타일로 인하여 선의의 피해를 입을 수도 있지만 대부분의 경우에 띄어쓰기 없이 길게 늘어지는 경우가 없으니 특별히 걱정하지 않으셔도 됩니다. 또한 앞서 언급한 두 가지 스타일을 테이블에 적용하면 더 이상 테이블 깨트리기는 불가능하니 마음껏 디자인을 해도 될 듯합니다.

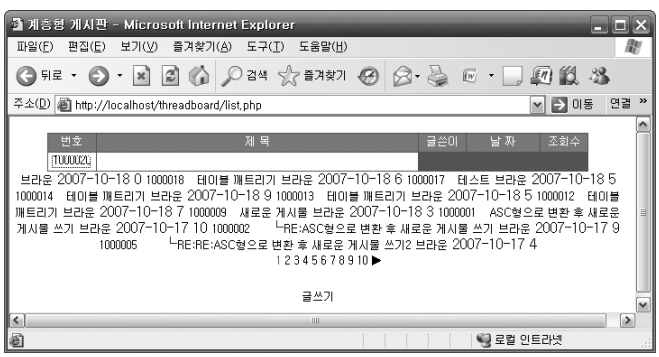
HTML을 이용한 공격 방지

지금으로부터 한 8년 전만 해도 대부분의 게시판은 마음껏 HTML을 사용할 수 있었습니다. 저와 나이대가 비슷한 분이시라면 공감할 수 있을 거라 생각되는데 그 당시만 하더라도 채팅이 매우 발달해 있었습니다. 그래서 채팅을 하다 보면 많은 사람 사이에서 자신의 존재를 알리고자 HTML 태그를 사용하여 글씨를 크게 한다거나 색깔을 넣는다거나 좌우로 글씨가 흐르게 한다거나 하는 방법으로 눈길을 끌고는 했습니다. 저와 같이 컴퓨터 관련 전공자이거나 웹 프로그래밍에 관심이 있던 사람이라면 조금 더 심해져서 채팅창의 배경 그림을 바꾼다든지 팝업창을 띄운다든지 배경음악을 재생한다든지 하는 기술을 화려하게 구사하고는 했습니다.

이처럼 게시판에서도 자유롭게 HTML을 허용하는 경우 악동들로부터 게시판을 다른 사람들이 전혀 사용할 수 없게끔 만들어 버리는 공격을 받기도 합니다.

그래서 어느 순간부터 게시판에서 HTML을 사용할 수 없게 하거나 부분적으로만 허용하는 경향이 많아졌습니다. 게시판 관리자만 HTML을 모두 사용할 수 있게 하고 일반 사용자에게는 부분적이거나 전혀 사용할 수 없게 만들어 버렸습니다.

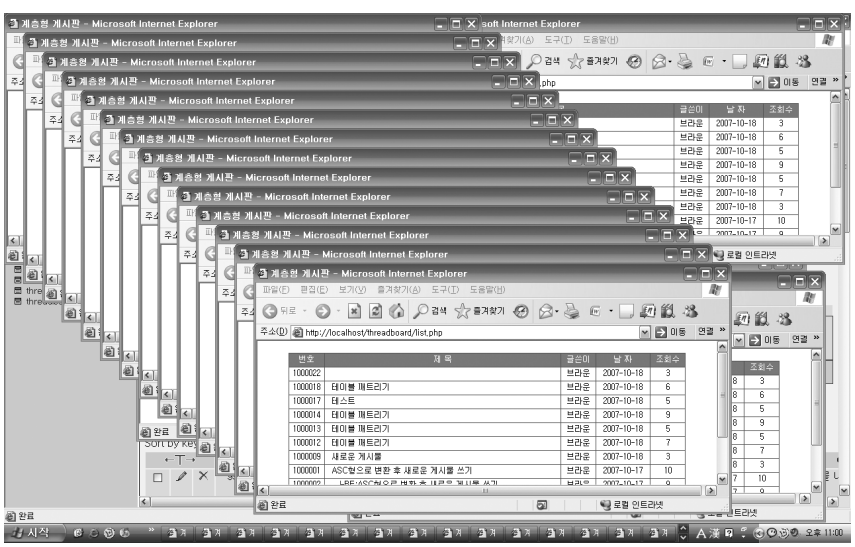
왜 그렇게까지 해야 했느냐 하면 다음과 같은 장난이 가능하기 때문입니다.



[그림 13-45] 테이블 모양을 파괴하는 태그 공격

앞서 테이블의 길이를 늘이는 귀여운 행동과는 비교할 수 없게 아예 게시판의 모양 자체를 파괴해 버리는 장난이 가능하기 때문입니다.

사실 이런 식의 테러는 온건파 집단의 테러라고 생각할 수 있습니다. 자바스크립트를 이용한 테러는 정말 몸서리치게 무시무시하기 때문입니다.



[그림 13-46] 자바스크립트를 이용한 무한 팝업창 공격

이처럼 무한히 반복되는 팝업창을 통한 테러도 가능해집니다. 심지어는 이를 이용하여 사용자의 컴퓨터를 다운시킬 수도 있습니다. 호환 마마보다도 무서운 자바스크립트 테러는 반드시 처리해야 할 최우선 과제입니다.

그러면 HTML과 자바스크립트를 이용한 테러를 방지하는 방법을 알아보겠습니다. 일반적으로 두 가지 방법을 많이 사용합니다.

① HTML과 자바스크립트를 무시하도록 한다.

② HTML과 자바스크립트를 입력한 값 그대로 문자로 보여준다.

첫 번째는 HTML이나 자바스크립트를 모두 무효화시킵니다. 따라서 아무리 태그를 입력한다 하더라도 HTML은 실행되지 않습니다. 즉, 애초부터 HTML을 입력하지 않은 것과 동일한 효과를 갖습니다. 이 방법의 장점은 아주 간단하게 그것도 완벽하게 테러를 방지할 수 있다는 것입니다.

PHP 내장 함수 중에는 strip_tags라는 함수가 존재합니다. 이 함수는 문자열에서 태그를 제거하는 기능을 하는 함수로 기본적으로는 모든 태그를 제거합니다.

```
$string = strip_tags("<B>PHP</B>");
```

위와 같이 strip_tags 함수를 사용하면 $string 변수에는 태그가 모두 사라지고 "PHP"라는 세 글자만 남게 됩니다.

그러나 상황에 따라 인수로 허용할 태그를 기입할 수 있어서 만약 부분적인 태그를 허용하는 경우 매우 유용하게 사용할 수 있습니다.

```
$string = strip_tags("<font color=red><B>PHP</B></font>", "<B>");
```

위와 같이 strip_tags 함수를 사용하면 $string 변수에는 "〈B〉PHP〈/B〉"가 저장됩니다. 인수로 〈B〉태그를 사용할 수 있게 허용했기 때문입니다. 만약 여러 개의 태그를 허용할 경우에는 다음과 같이 사용하면 됩니다.

```
$string = strip_tags("<font color=red><B><I>PHP</I></B></font>",
"<B><I>");
```

이처럼 두 번째 인수에 허용할 태그를 계속해서 덧붙이면 해당하는 태그만 허용되고 나머지 모든 태그는 제거됩니다. 또한 두 번째 인수에서 대소문자를 구별하지 않기 때문에 태그를 대문자와 소문자로 나누어서 등록할 필요가 없습니다.

이러한 매우 편리한 함수를 이용하여 간단하게 테러를 방지할 수 있습니다. 또한 부분적으로 HTML을 사용하게 하는 경우에도 매우 훌륭하게 동작합니다. 그러나 이 방법을 사용하면 태그가 원초적으로 사라져서 HTML 소스를 공개한다든가 하는, 글 내용 자체에 태그를 보여주어야 하는 경우에는 태그가 모두 사라지게 되어 제대로 출력되지 않는 단점이 있습니다.

두 번째 방법은 사용자가 입력한 값을 그대로 보여주는 것입니다. 그 말은 HTML 태그를 사용하였다면 태그가 적용된 상태로 보여지는 것이 아니라 태그 문자 자체를 출력한다는 말입니다. 예를 들어 〈font size=5〉PHP〈/font〉 라고 입력하였을 때 글씨가 5사이즈인 "PHP"라는 문자를 표시하는 것이 아니라 입력한 그대로 〈font size=5〉PHP〈/font〉라고 보여주는 것입니다. 즉, HTML이든 자바스크립트든 입력해봐야 입력한 값 그대로 보여주겠다는 것입니다.

이 방법의 장점은 웹 프로그래머 커뮤니티와 같이 게시물 내용에 HTML이나 자바스크립트의 내용이 포함되는 경우에 태그가 사라지는 것을 막을 수 있는 장점이 있습니다.

HTML 및 자바스크립트를 무효화시키는 방법은 HTML 태그와 자바스크립트에서 사용하는 꺽쇠를 치환하는 것입니다. HTML 태그와 자바스크립트는 모두 "〈"으로 시작하여 "〉"으로 끝납니다. 이 기호를 치환하는 문자는 다음과 같습니다.

기호	변환 기호	의미
〈	<	less than
〉	>	greater than

[표 13-8] 태그 시작 및 끝 기호의 치환

<와 >는 태그를 웹 브라우저에서 보이게 하기 위한 특별한 문자입니다. 웹 브라우저에서 〈BR〉는 줄 바꿈으로 보이지만
는 〈BR〉이라는 문자로 보입니다. 따라서 문자열 치환을 통해서 모든 입력 값을 입력한 그대로 보여주는 것이 가능해집니다.

```
$string = str_replace("<", "&lt;", "<B>PHP</B>");
$string = str_replace(">", "&gt;", $string);
```

위와 같이 str_replace 함수를 사용하면 $string 변수에는 결과적으로 <B>PHP<B>가 남게 되고 웹 브라우저에는 〈B〉PHP〈/B〉가 보입니다. 실제로는 왼쪽 꺽쇠만 바꾸더라도 태그는 항상 왼쪽 꺽쇠(〈)와 오른쪽 꺽쇠(〉)가 쌍을 이뤄야 하기 때문에 태그를 사용할 수 없게 됩니다.

첫 번째 방법이 간단하고 편리하지만 앞서 말한 바와 같이 HTML 소스를 그대로 출력해야 한다면 바람직하지 못합니다. 그래서 치환법을 사용하여 실제 게시판 코드에 적용해 보겠습니다.

우선 실제 적용하기에 앞서서 자주 이용할 기능이므로 함수로 공용 라이브러리에 등록하기로 합니다.

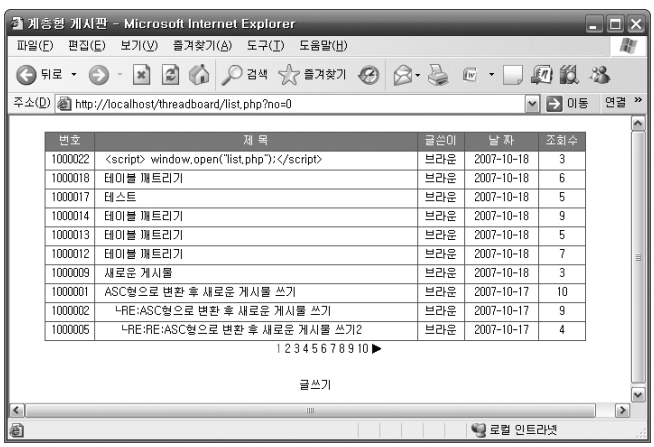
```
function remove_html($string)
{
  $string = str_replace("<", "&lt;", $string);
  $string = str_replace(">", "&gt;", $string);
  return $string;
}
```

library.php 파일에 위와 같이 새로운 함수를 추가합니다. 이제 HTML 코드를 제거하려면 library.php 파일을 인클루드한 다음에 remove_html 함수를 사용하면 됩니다.

먼저 가장 문제가 되는 글 목록 페이지를 수정해 봅시다. 글 목록에서는 제목을 통한 테러가 많이 발생합니다. 그래서 제목을 보여줄 때 remove_html 함수를 이용하여 HTML을 제거합니다.

```
<?=remove_html($row[title]);?>
```

$row[title] 변수를 출력하는 부분을 위와 같이 수정하면 됩니다. 그 결과 앞에서 무한 반복되던 팝업창은 다음과 같이 출력되었습니다.



[그림 13-47] 자바스크립트를 그대로 보여주는 방법

이제 글 읽기 페이지에도 이처럼 적용해 봅시다. 글 읽기 페이지에는 제목과 본문 내용이 출력되는 부분을 모두 수정해야 합니다. 해당 글에 대한 제목과 내용뿐만이 아니라 아랫글, 윗글의 제목도 처리해야 하고 하단의 관련 글 목록 부분의 제목까지 모두 다섯 군데를 수정해야 합니다.

① 해당 글의 제목<?=remove_html($row[title]);?>

② 해당 글의 본문

```
<?=nl2br(str_replace(" ", " ", remove_html($row[content])));?>
```

③ 윗글의 제목

```
echo "<a href=read.php?id=$up_id[id]>△ " . remove_html($up_id[title]) .
"</a></td><td align=right>$up_id[name]</td></tr>";
```

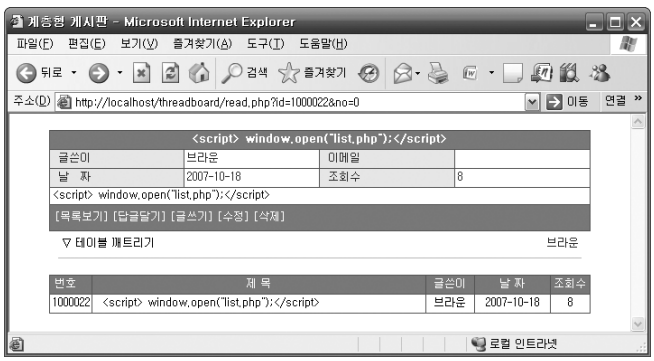
④ 아랫글의 제목

```
echo "<a href=read.php?id=$down_id[id]>▽ " . remove_html($down_
id[title]) . "</a></td><td align=right>$down_id[name]</td></tr>";
```

⑤ 관련 글 목록의 제목

```
<a  href=read.php?id=<?=$row[id]?>&no=<?=$no?>><?=remove_
html($row[title]);?></a>
```

이렇게 모두 다섯 군데를 수정하면 다음과 같이 HTML이 제거된 상태의 글을 읽을 수 있습니다.



[그림 13-48] 태그 기호의 치환을 이용한 자바스크립트 공격 방지

그러나 이것으로 모두 끝나는 것이 아닙니다. 여기서는 제목과 본문 내용만을 다루었지만 실제로는 이름이나 이메일 등을 통해서도 여전히 공격받을 수 있습니다. 그렇다면 모든 출력 부분에 대해서 HTML을 제거해야 할까요? 물론 그렇게 해도 무방합니다. 그러나 출력할 때마다 매번 신경 써서 HTML을 제거해주는 것은 무리가 있습니다.

이름과 이메일, 제목 그리고 본문 내용 이렇게 네 가지 항목을 통해 게시판 공격이 가능합니다. 그런데 이 중에서 제목과 본문 내용은 문구에 HTML이 실제로 들어갈 수 있습니다. 그러나 이름과 이메일은 HTML 태그가 들어갈 아무런 이유가 없습니다. 우리나라를 비롯하여 어떤 나라의 이름에도 HTML 태그와 같은 이름은 존재하지 않으며 이메일은 올바른 이메일 주소가 되기 위한 규칙이 존재합니다. 따라서 이름과 이메일은 굳이 HTML을 입력할 필요가 없으므로 글을 입력할 때 strip_tags 함수를 이용하여 애초에 제거하기로 합니다.

```
$_POST[name] = trim(strip_tags($_POST[name]));
$_POST[email] = trim(strip_tags($_POST[email]));
```

insert.php와 insert_reply.php 그리고 update.php 파일에 이 코드를 삽입합니다. 코드를 삽입할 때 필수 항목 검증 부분 앞에 넣도록 합니다. 그 이유는 이름값에 HTML 태그만을 입력한 경우 태그가 제거되면서 빈 값이 되어버리기 때문입니다. 만약 필수 항목 검증 이후에 코드를 삽입하게 되면 빈 값이 그대로 이름 항목에 입력됩니다. 따라서 반드시 필수 항목 검증을 하기 전에 삽입하도록 합니다.

이제 HTML이나 자바스크립트를 이용한 게시판 공격이 불가능해졌습니다. 이 책에서는 원천적으로 HTML과 자바스크립트를 사용하지 못하도록 막는 방법을 다루고 있습니다. 그러나 실제로 부분

적으로 HTML을 허용한다면 매우 다양하게 공격받을 수 있습니다. 실로 놀라울 정도로 많은 방법으로 공격하는데 특히 스패머(스팸발송자)나 성인 사이트 광고 글에 이러한 최신(?) 기법들이 등장합니다. 이를 모두 막기 위해서는 정규식을 이용하여 모든 방법에 대처를 해야 합니다. 일반적으로 script라는 단어를 변경하고 onXXXX와 같은 이벤트 핸들러를 전부 무효화시킵니다.

스팸글 등록 방지

요즘 어디에서나 쉽게 스패머를 볼 수 있을 만큼 스팸 게시물이 넘쳐나고 있습니다. 이 대부분의 스팸성 글들은 글을 자동으로 등록하는 봇(bot)에 의해 등록되며 무차별적이기 때문에 홈페이지 주인들의 가슴에 불덩이(?)를 심어주곤 합니다. 필자가 운영하는 사이트에도 강좌용 게시판이 초토화될 정도로 스팸성 글이 많이 등록됩니다.

[그림 13-49] 스팸성 글로 점령된 게시판

우선 스팸성 글을 막으려면 어떻게 등록되는지부터 알아야 합니다.

게시판의 글 쓰는 방법은 어떤 게시판이든지 거의 비슷합니다. 예를 들면 입력하는 값은 이름, 비밀번호, 제목, 내용이 빠지지 않고 나옵니다. 또한 이 폼의 이름 또한 대부분 유사합니다. 이러한 유사성을 기반으로 글 입력 폼을 분석하여 자동으로 글을 등록하는 것을 봇(Bot)이라고 합니다.

봇의 행동을 살펴보면 다음과 같습니다. 먼저 유명한 사이트나 개인이 운영하는 홈페이지 등의 게시판 주소를 얻어와서 저장해둡니다. 이러한 주소들은 검색엔진을 통해서 쉽게 구할 수 있습니다. 봇이 본격적으로 동작하기 시작하면 등록된 주소로 접근하여 웹 페이지를 불러온 후 폼 정보를 분석하여 이름, 비밀번호, 제목, 본문 등의 정보를 파악합니다.

여러 종류의 봇이 존재하는데 실제로 사람이 등록하는 것과 동일하게 입력 폼 페이지부터 시작하는

봇이 있고 폼 페이지를 분석한 후 바로 글 저장 페이지로 값을 전달하는 봇이 있습니다.

윈도우 프로그래밍을 해본 사람이라면 그리 어렵지 않게 만들 수 있으니 봇이 넘쳐나는 것 같습니다. 그러나 요즘 봇이 판을 치다 보니 웹 프로그래머들도 여러 가지 방법으로 방지하고 있어서 최근 점점 더 봇이 똑똑해지는 추세입니다.

그렇다면 광고 글을 어떻게 막을 수 있을까요?

광고 글을 원초적으로 막기는 힘듭니다. 지금 홈페이지에 접속한 것이 봇인지를 알아낼 수 있다면 쉽게 막을 수 있겠지만 그렇다고 봇이 "나 봇이오~"라고 알려주면서 접속할 리가 없습니다. 그래서 어쩔 수 없이 많은 대안 책을 생각해내게 되었습니다.

① 회원제 게시판으로 변경
② IP 차단
③ HTTP_REFERER 확인
④ 글 등록 시간 제한
⑤ 광고 문구 차단
⑥ 인증 문자를 입력

물론 이것 이외에도 많은 방법이 존재합니다. 그러나 대표적으로 6가지를 설명하고자 합니다.

우선 첫 번째로 회원제 게시판으로 변경합니다. 게시판에 글을 쓰려면 반드시 회원 가입을 해야 한다는 단점이 있지만 이러한 점 때문에 스팸성 글들이 등록될 가능성이 매우 낮습니다. 어쩌면 가장 강력한 방법이면서 최후의 보루인 방법입니다. 그러나 완벽하게 스팸성 글에서 벗어날 수 있는 것은 아닙니다. 봇이 글을 쓰듯이 회원 가입을 하면 되기 때문입니다. 그래도 필자의 생각으로는 회원제를 선택하는 순간 90% 이상의 스팸성 글은 더 이상 올라오지 않으리라 생각합니다. 또한 만약 등록된다 하더라도 비회원제였을 때보다 손쉽게 스패머를 차단할 수 있습니다.

두 번째 방법은 스패머의 IP를 차단하는 방법입니다. 스팸성 글을 계속해서 등록하는 사람의 IP를 알아내서 원천적으로 접속을 차단하는 방법입니다. 데이터베이스에 차단 IP를 저장해두고 접속자의 IP와 비교하여 등록된 IP면 접속을 차단하는 방법을 사용하면 됩니다. 그러나 이 방법은 이미 스팸 공격에 당하고 나서 뒤처리 성향이 강하고 또한 봇은 대부분 유동 IP를 사용하거나 프록시 서버 등을 통해서 IP를 변경하여 접속하기 때문에 이 방법은 실효성이 그리 크지 않습니다.

세 번째 방법은 $HTTP_REFERER 변수를 이용한 방지법입니다. 대부분의 봇이 글 쓰기 페이지나 직접 글 저장 페이지로 접근합니다. $HTTP_REFERER 변수는 현재 페이지로 이동하기 전에 어떤 페이지에 있었는지 정보를 갖고 있습니다. 만약 직접 글 저장 페이지로 직접 들어왔다면 $HTTP_REFERER 변수에는 write.php 파일이 아닌 다른 값이 들어 있을 것입니다. 정상적인 경우라면 반드시 글 쓰기 페이지부터 넘어 왔을 텐데 말이죠. 글 쓰기 페이지부터 시작하는 봇이라도 글 쓰기

페이지에서 $HTTP_REFERER를 검사하면 봇을 구별할 수 있습니다. 일반적으로 글 쓰기를 할 때는 글 목록이나 글 읽기 페이지에서 넘어올 텐데 다른 사이트나 $HTTP_REFERER 값이 없는 경우는 의심할만하기 때문입니다. 그러나 이 방법은 봇이 $HTTP_REFERER 정보를 조작할 수 있기 때문에 실질적인 효용성이 떨어집니다.

네 번째 방법은 글을 등록하는데 소요되는 시간을 확인하는 방법입니다. 봇은 프로그램이기 때문에 사람처럼 글을 쓰는데 생각을 하거나 머뭇거리지 않습니다. 또한 저장할 내용이 이미 정해져 있기 때문에 봇은 글 쓰는 시간이 매우 짧습니다. 그래서 글 쓰는 페이지에 들어왔을 때 시간 정보를 저장해 두었다가 글이 입력될 때까지의 시간을 비교하여 일정 시간 이하인 경우에 봇으로 판단하여 등록을 거부하는 방법입니다. 예를 들어 글 쓰기부터 저장까지 1초 미만이라면 봇으로 충분히 의심할만합니다. 왜냐하면 장난으로 아무렇게나 작성한 글이나 자동으로 등록하는 봇이 아니고서야 1초만에 글을 쓰는 것은 불가능하기 때문입니다. 그러나 이 방법은 봇이 글을 저장하는 시간을 늦춰버리면 무용지물이 됩니다. 그러나 봇의 목적이 짧은 시간 안에 최대한 많은 양의 게시물을 등록하는 것이므로 하나의 글을 등록하는데 오랜 시간이 소요된다면 봇 입장에서는 글 하나 등록하는 시간에 다른 곳에서 여러 개의 글을 등록할 수 있기 때문에 꺼려지는 방법이라 할 수 있습니다. 하지만 쿠키를 이용하여 작성 시간 정보를 계산한다면 이 또한 봇이 쿠키를 생성하는 방법으로 회피할 수 있습니다.

다섯 번째 방법은 광고 글에서 자주 등장하는 문구나 단어를 금지하도록 하는 방법입니다. 광고 글에는 대개 비슷한 유형의 단어가 많이 등장합니다. 예를 들면 "포*노", “몰카”, “야동”, "무료"와 같은 단어가 자주 등장합니다. 일반적으로 평범한 글을 쓰는 사람은 잘 입력하지 않지만 광고성 글의 특성상 자주 등장하는 단어가 글 내용에 포함되면 광고 글로 판단하고 차단하는 방법입니다. 이 방법은 실제로 가장 많이 사용되고 있습니다. 그러나 이 또한 금지 단어 사전에 해당하지 않는 글들이 올라오면 그때그때 금지 단어를 업데이트해야 하고 워낙 많이 사용되는 방법이다 보니 봇 또한 금지어에 속하지 않게 “몰.카”, “야#동”, “$무$료$” 등과 같이 단어를 변형하여 등록하기도 하는 등 어려움이 많습니다. 또한 선의의 피해자가 생길 수 있기 때문에 금지어를 등록하는데도 어려움이 존재합니다. 예를 들면 “오늘은 너무 무료하다”와 같은 문장을 등록하고 싶은데 무료라는 말 때문에 글이 등록되지 않을 수 있기 때문입니다.

마지막으로 여섯 번째 방법은 인증을 위한 특수한 문자를 입력하게 하는 방법입니다. 다음이나 네이버 같은 대규모 포털 사이트에서 회원 가입에 사용하는 방법으로 이미지로 글자를 보여줘서 그 글자를 사람으로 하여금 입력하게 하는 방법입니다. 초기에는 대규모 사이트에만 적용되다가 근래에는 개인 홈페이지의 게시판까지 많은 곳에서 사용되고 있습니다. 실제로 이미지에서 글자를 인식하는 프로그램이 존재하지만 아직 봇에 글자 인식 엔진이 탑재된 경우는 없는 것 같습니다. 또한 일부러 글씨를 특이하게 출력하여 사람은 쉽게 알아볼 수 있으나 프로그램은 알아볼 수 없게 만들어서 글자 인식 엔진이 탑재되어도 어려울 것으로 예상합니다. 따라서 이 방법을 사용하면 스팸 방지

효과를 확실하게 얻을 수 있을 것으로 생각합니다. 그러나 글을 쓸 때 매번 인증 문자를 입력해야 하므로 사용자가 매우 불편해하는 단점이 있습니다.

이외에 필자가 자주 이용하는 방법 중의 하나로 게시판 주소를 가끔씩 변경하는 방법도 있습니다. 봇이 주소를 기록했다가 저장된 주소로 직접 접속하므로 가끔 주소를 변경해주면 봇이 주소를 업데이트할 때까지 스팸성 글이 등록되지 않게 할 수 있습니다. 하지만 이 방법 역시 일시적인 미봉책에 불과합니다.

스팸 방지 기법에 대해서 알아보면서 느꼈듯이 스팸을 원초적으로 방지하는 것은 불가능에 가깝습니다. 단지 여러 가지 기법으로 봇을 괴롭게 하여 등록이 쉽지 않게 만드는 것이 대부분입니다. 그래서 여기에 언급한 기법을 섞어서 사용하면 보다 스팸성 글에서 자유로운 게시판을 구현할 수 있습니다.

Section 03 게시판 기능 추가

점점 게시판이 그럴듯한 모습을 갖춰가고 있습니다. 게시판의 처리 속도도 실제로 사용할 수 있을 만큼 양호해졌고 여러 가지 공격도 대처할 수 있게 되었습니다. 기본적인 틀이 어느 정도 완성되었으니 기능을 몇 가지 더 추가해 보고자 합니다.

추가하고자 하는 기능은 다음과 같습니다.

① 간단한 덧글 기능
② 파일 업로드 기능

우선 첫 번째로 간단한 덧글 기능은 최근 게시판의 추세로 점차 답변 글을 대체하고 있습니다. 현재 답변 글과 간단한 덧글이 공존하고 있지만 점차 답변 글 기능이 사장되고 간단한 덧글이 이를 대체할 것으로 생각합니다.

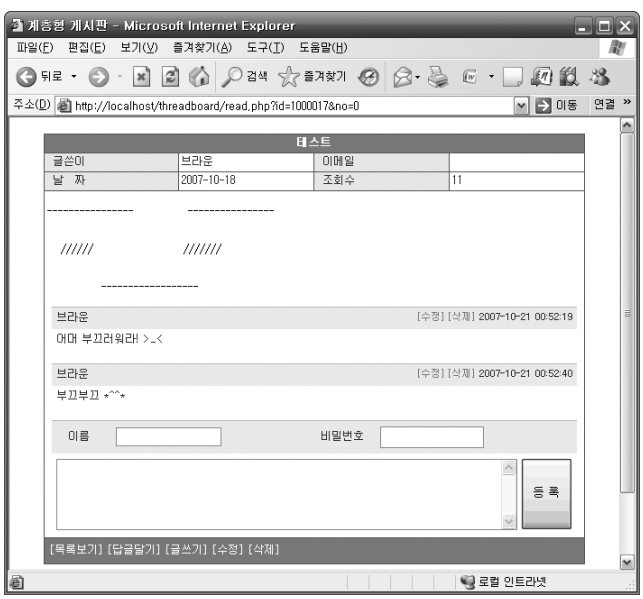
두 번째로 파일 업로드는 자료실을 만들고자 많이 구현되었으나 최근에는 게시물에 이미지나 문서 등을 업로드하는 경우가 많아져서 대부분의 게시판에서 파일 업로드를 지원하고 있습니다. 그러나 파일 업로드는 매우 주의해서 구현해야 합니다. 왜냐하면 파일 업로드를 이용한 해킹 사례가 매우 많이 일어나기 때문입니다. 따라서 보안을 항상 염두해두고 파일 업로드를 구현해야 합니다.

간단한 덧글

덧글은 답글과 달리 답변적인 성향보다는 글쓴이의 생각이나 글 본문의 내용에 대한 의견을 덧붙인다는 의미에서 덧글이라고 말합니다. 그렇다고 답변적인 성향이 전혀 없는 것은 아닙니다. 그러나 대개 꽤 긴 응답 글을 쓸 때는 답글을 이용하고 짧은 답변이나 내용에 코멘트를 달 때는 덧글을 사용하고 있습니다.

덧글은 답글과 달리 읽어보기 위해 글을 찾거나 따로 클릭할 필요 없이 원문 하단에 바로 표시되기 때문에 방문자가 본문의 내용을 보고 난 후 덧글을 보는 경우가 많다는 이유로 답변 기능을 사용하지 않고 덧글에 답변을 등록하는 경우가 많아지게 되었습니다. 최근 제로보드를 비롯하여 답변 기능을 지원하지 않고 덧글만을 허용하는 게시판이 많아지고 있습니다. 또한 덧글이 한 단계 진화하여 덧글에 다시 덧글을 등록할 수 있게 덧글을 계층형으로 만드는 게시판도 눈에 띄게 증가하고 있습니다.

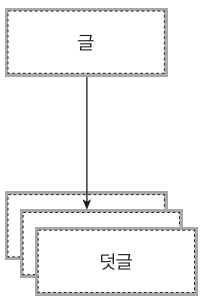
너도나도 덧글이 대세라고 하니 우리도 빠지지 않고 대세에 동참해 봅시다. 우리가 만들어 볼 덧글 기능은 다음과 같습니다.



[그림 13-50] 덧글 기능이 추가된 글 읽기 페이지

덧글 스키마 작성

덧글을 구현하기 위해서 일반 글과 덧글과의 관계를 생각해 봅시다.



[그림 13-51] 글과 덧글의 1:N 관계

일반 글과 덧글의 관계는 위의 그림처럼 일반 글 하나에 여러 개의 덧글이 등록되는 1대 다(1:N) 관계입니다. 이 관계를 표현하려면 새로운 테이블을 생성해야 합니다. 물론 재귀적으로 하나의 테이블에 구현할 수도 있습니다. 그러나 일반 글에 비해 덧글의 내용이 매우 짧고 덧글을 구현할 때 불필요한 필드를 포함하므로 공간 사용 측면에서 동일한 테이블을 사용하는 것은 낭비라고 할 수 있습니다.

그래서 새로운 테이블을 만들어 보겠습니다.

우선 덧글에 필요한 필드를 생각해 봅시다. 우선 덧글을 구분 지을 수 있는 키가 되는 덧글 id가 있을 것이고 일반 글과 마찬가지로 글쓴이, 비밀번호, 제목 그리고 덧글 내용이 있을 것입니다. 또한 등록된 시간과 날짜를 표시하기 위해서 날짜도 필요합니다. 그리고 공격적인 악플러를 막기 위해서 IP를 기록하는 것도 좋은 생각인 것 같습니다.

이렇게 덧글을 위해서 필요한 것들을 정리해보면 다음과 같습니다.

항 목	영어 항목	변수형	크기	비 고
덧글 번호	id	int	11	기본키
글쓴이	name	varchar	15	
비밀번호	pass	varchar	15	
내용	comment	text		65,535개 가변 문자열
날짜	wdate	timestamp		날짜 및 시간
IP 주소	ip	varchar	15	

[표 13-9] 덧글 테이블의 스키마

일단 threadboard의 스키마와 비교해보면 제목과 이메일 그리고 조회 수가 보이질 않습니다. 덧글은 본문의 내용에 덧붙이는 말이기 때문에 글의 주제는 언제나 본문의 내용에 바탕을 둡니다. 그래서 일반적으로 덧글에는 제목을 붙이지 않습니다. 또한 그다지 불필요한 이메일과 같은 정보를 없애서 덧글 등록을 쉽게 하고 저장 공간도 적게 소모하게 만듭니다. 마지막으로 조회 수는 덧글의 내용이 모두 공개되어 조회 수라는 것의 의미가 없기 때문에 제거합니다.

이외에 계층을 나타내는 thread나 depth가 존재하지 않습니다. 이는 일반형 게시판과 같이 덧글에는 답변 덧글을 달 수 없도록 했기 때문에 그렇습니다. 최근의 추세대로 덧글에 다시 덧글을 덧붙일 수 있게 하려면 comment 테이블에도 thread와 depth를 추가해야 합니다.

이제 스키마를 대충 작성하고 테이블 생성문을 만들어 볼까 하였더니 뭔가 찝찝한 느낌을 감출 수 없습니다. 그래서 한 번 더 고민을 해 보았더니 아차~ 제일 중요한 것을 빠뜨렸군요. 무엇을 빠뜨렸을까요?

앞에서도 언급하였듯이 일반 글과 덧글은 1대 다의 관계를 가집니다. 그렇다면 하나의 글에 여러 개의 덧글이 등록된다는 뜻이고 글을 읽었을 때 그와 관련된 덧글이 주르룩 출력된다는 뜻이 됩니다. 그렇다면 어떻게 관련이 있는지 알 수 있을까요? 한번 상상의 나래를 펼쳐봅시다.

> 어떤 마을에는 공동으로 운영하는 젖소 목장이 있습니다. 드넓고 푸른 언덕에 젖소들이 마음껏 뛰어놀 수 있도록 넓은 목장에 자유롭게 풀어두고 키우고 있습니다. 마을 사람들은 각자 다섯 마리에서 많게는 열 마리, 스무 마리씩 자기의 젖소를 이 목장에 풀어놓고 키우다가 젖을 짤 시간이 되면 목장에서 자기 젖소를 찾아다가 젖을 짭니다. 그런데 마을 사람들이 키우는 젖소가 총 200마리가 넘는다면 자신의 젖소를 어떻게 구별할 수 있을까요? 비슷비슷하게 생긴 젖소들 얼굴을 일일이 외워둘 수도 없고 말이죠. 그래서 마을 사람들은 자신의 이름이 적힌 이름표를 젖소에다 달아주었습니다. 이름표를 보고 자신의 이름이 적혀 있다면 자신의 젖소인 것이 분명하니 쉽게 찾을 수 있게 말입니다.

이처럼 덧글도 마찬가지로 일반 글의 이름표를 갖고 있어야 합니다. 이름표를 가지라고 하니 그러면 name 값을 가지면 될까요? 만약 같은 이름을 갖는 두 사람이 있다면 누구의 덧글인지 알 수 없으므로 안될 것이 분명합니다. 그래서 이름표는 언제나 그 사람을 확실히 구분할 수 있는 키(Primary Key)를 사용합니다.

threadboard 테이블에서 키를 찾아보니 두 개가 있습니다. 하나는 id이고 다른 하나는 thread입니다. 둘 다 중복되지 않고 게시물 하나를 정확하게 구별할 수 있는 값이기 때문에 둘 다 키가 맞습니다. 그래서 둘 중에 어떤 것을 이름표로 사용하더라도 상관이 없습니다. 하지만 일반적으로 키를 사용할 때는 기본키(Primary Key)를 사용하므로 id 값을 사용하도록 합니다.

일반 글의 번호를 의미하는 id를 추가하면 다음과 같습니다.

항 목	영어 항목	변수형	크기	비 고
덧글 번호	id	int	11	기본키
일반글 번호	bid	int	11	외래키

글쓴이	name	varchar	15	
비밀번호	pass	varchar	15	
내용	comment	text		65,535개 가변 문자열
날짜	wdate	timestamp		날짜 및 시간
IP 주소	ip	varchar	15	

[표 13-10] 관계가 표현된 덧글 스키마

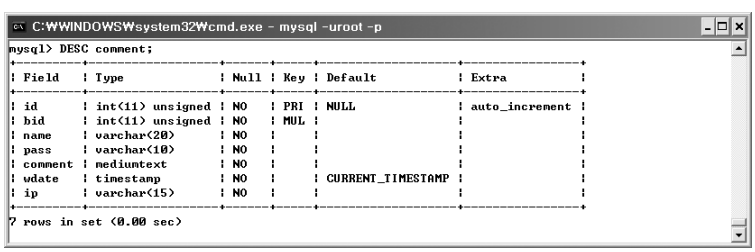
이처럼 테이블에 다른 테이블의 키를 삽입하는 경우 이 항목을 '외래키'라고 부릅니다. 그냥 말 그대로 외부에 있는 키를 의미합니다.

이제 테이블 스키마가 완성되었으니 테이블 생성문을 작성해 봅시다.

```
CREATE TABLE `comment` (
  id int(11) unsigned NOT NULL auto_increment,
  bid int(11) unsigned NOT NULL,
  name varchar(20) NOT NULL,
  pass varchar(10) NOT NULL,
  comment text NOT NULL,
  wdate timestamp NOT NULL default CURRENT_TIMESTAMP,
  ip varchar(15) NOT NULL,
  PRIMARY KEY(id),
  KEY idx (bid)
) ENGINE=MyISAM DEFAULT CHARSET=euckr;
```

최소의 입력 사항만 필드로 추가했기 때문에 모든 필드가 NULL 값을 거부합니다. 또한 wdate는 값을 입력하지 않으면 자동으로 데이터베이스에서 입력 시간 정보를 저장하도록 기본값을 CURRENT_TIMESTAMP와 같이 지정해 주었습니다. 마지막으로 id를 기본키로 등록하고 외래키인 bid를 키로 등록해 줍니다. 키로 등록하면 인덱스를 사용하게 되어 검색이 빠릅니다.

테이블 생성문을 실행하고 제대로 등록되었는지 확인을 해보면 다음과 같습니다.



[그림 13-52] 덧글 테이블의 정보

덧글 쓰기 폼

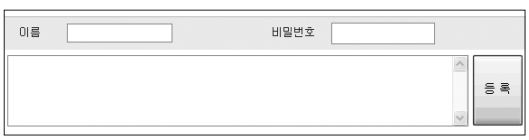
덧글을 구현하기 위해서는 앞서 배운 일반형 게시판을 떠올리면 됩니다. 덧글을 등록하고 목록을

보여주고 덧글을 삭제하는 것은 모두 일반형 게시판의 알고리즘을 바탕으로 합니다. 일반형 게시판과 덧글의 차이를 생각해보면 글 목록이 글 읽기와 글 쓰기 폼 페이지를 대신하고 페이징이 필요 없다는 것밖에 차이가 없습니다. 페이징이 없다는 것은 덧글의 목록을 출력하는 부분이 매우매우 간결해진다는 것을 의미하므로 일반형 게시판을 만드는 것보다 훨씬 쉬울 거라는 느낌이 팍팍 올거라 생각합니다. 그렇죠? ^^

우선 덧글을 등록할 수 있게 덧글 쓰기 폼부터 만들어 봅시다.

우리는 이미 글 쓰기 폼을 많이 만들어 보았기 때문에 여러분은 설명 없이도 충분히 만들 수 있을 것으로 생각합니다. 그래도 그냥 넘어갔다가는 날로 먹었다고 악성 덧글을 달지도 모르니 간단하게 설명드리겠습니다.

우선 웹 에디터나 메모장 등을 이용하여 다음과 같은 모양의 폼을 구성합니다.



[그림 13-53] 덧글 쓰기 폼

그림에서 알 수 있듯이 입력 항목은 매우 간략합니다.

① 이름 – name

② 비밀번호 – pass

③ 내용 – comment

이처럼 세 가지 항목만 기입하면 덧글을 등록할 수 있게 폼을 작성합니다. 단, 모든 항목이 필수 항목이니 필수 항목을 기입하였는지 꼭 확인하도록 합니다.

```
<script>
  function CommentFormCheck() {
    if (!comment_insert.name.value) {
      alert("이름을 입력하세요.");
      comment_insert.name.focus();
      return false;
    }
    if (!comment_insert.pass.value) {
      alert("비밀번호를 입력하세요.");
      comment_insert.pass.focus();
      return false;
    }
    if (!comment_insert.comment.value) {
      alert("내용을 입력하세요.");
```

```
        comment_insert.comment.focus();
        return false;
      }
    }
</script>

<form name=comment_insert method=post action='comment_insert.php'
onsubmit="return CommentFormCheck()">
이름 <input type=text name=name size=15>
비밀번호 <input type=password name=pass size=15>
<textarea name=comment rows=5 cols=70></textarea>
<input type=submit value=' 등록 '>
</form>
```

디자인을 제외하고 중요한 부분을 정리하면 위와 같습니다. 여러분도 이름이나 위치 등의 차이는 약간씩 있겠지만 위와 유사하게 작성했을 것으로 생각합니다. 그런데 여기에는 매우 중요한 정보가 빠져 있습니다. 바로 해당 글의 번호 정보입니다. 앞서 스키마를 작성하면서 언급하였듯이 덧글 테이블에는 일반 글의 번호를 저장하게끔 되어 있습니다. 그래서 아래와 같이 숨겨진 폼을 이용하여 해당 글의 번호를 전달하게끔 합니다.

```
<input type=hidden name=bid value='<?=$_GET[id]?>'>
```

디자인이 적용된 글 쓰기 폼 페이지는 다음과 같습니다.

Comment.php

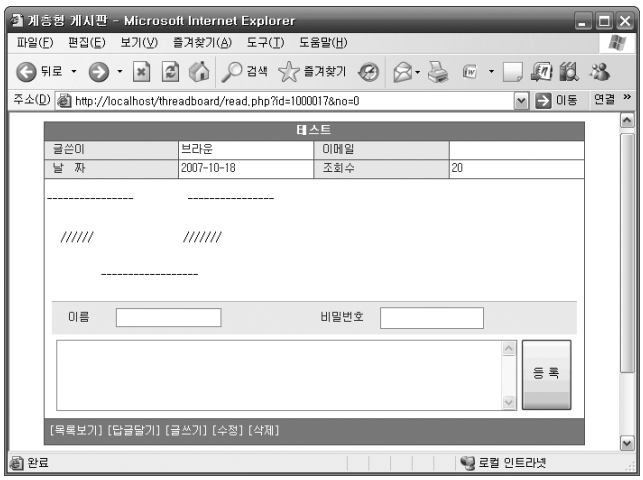
```
1  <script>
2    function CommentFormCheck() {
3      if (!comment_insert.name.value) {
4        alert("이름을 입력하세요.");
5        comment_insert.name.focus();
6        return false;
7      }
8      if (!comment_insert.pass.value) {
9        alert("비밀번호를 입력하세요.");
10       comment_insert.pass.focus();
11       return false;
12     }
13     if (!comment_insert.comment.value) {
14       alert("내용을 입력하세요.");
15       comment_insert.comment.focus();
16       return false;
17     }
18   }
19 </script>
```

```
20
21 <table width=98% align=center border=0 cellpadding=5 cellspacing=0>
22 <form name=comment_insert method=post action='comment_insert.php'
23 onsubmit="return CommentFormCheck()">
24 <input type=hidden name=bid value='<?=$_GET[id]?>'>
25 <tr bgcolor=#CCCCCC><td colspan=4></td></tr>
26 <tr bgcolor=#F0F0F0>
27   <td width=50 align=center>이름</td>
28   <td width=100><input type=text name=name size=15></td>
29   <td width=50 align=center>비밀번호</td>
30   <td><input type=password name=pass size=15></td>
31 </tr>
32 <tr>
33   <td valign=absmiddle colspan=4><textarea name=comment rows=5
34   cols=70></textarea>
35   <input type=submit value=' 등 록 ' style="height:80;width:57">
36   </td>
37 </tr>
38 </form>
39 </table>
```

완성된 소스 코드는 comment.php라는 이름으로 저장하고 read.php의 본문 내용을 출력하는 부분 하단에 comment.php 파일을 인클루드시킵니다.

```
<tr>
<td bgcolor=white colspan=4 style="word-break:break-all;">
<BR>
<font color=black>
<?=nl2br(str_replace(" ", " ", remove_html($row[content])));?>
</font>
        <BR><BR>
        <? include "comment.php"; ?>

</td>
</tr>
```

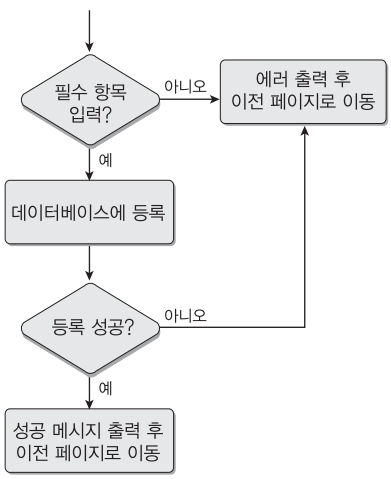
수정된 read.php 파일은 다음과 같이 덧글 등록 부분이 추가됩니다.



[그림 13-54] 덧글 쓰기 폼이 추가된 읽기 페이지

덧글 등록

앞에서 작성한 덧글 쓰기 폼을 이용하여 전달된 덧글 정보를 데이터베이스에 등록하는 comment_insert.php 파일을 만들어 보겠습니다.



[그림 13-55] 덧글 등록의 흐름

위의 순서도와 같이 덧글의 등록이 이루어집니다. 일반형 게시판의 insert.php 파일을 참조해서 comment_insert.php 파일을 만들면 됩니다.

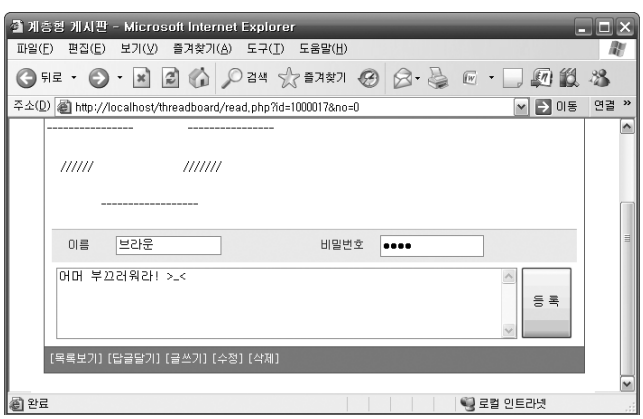
Comment_insert.php

```
1  <?
2     include_once "library.php";
```

```
3
4     $_POST[name] = trim(strip_tags($_POST[name]));
5     //입력값 검증
6
7     if (!$_POST[name]) ErrorMessage('이름을 입력하세요.');
8     if (!$_POST[pass]) ErrorMessage('암호를 입력하세요.');
9     if (!$_POST[comment]) ErrorMessage('내용을 입력하세요.');
10
11    include_once "db_info.php";
12
13    $query = " INSERT INTO comment (bid, name, pass, comment, ip) ";
14    $query .= " VALUES ('{$_POST[bid]}','{$_POST[name]}',
15    '{$_POST[pass]}','{$_POST[comment]}','$REMOTE_ADDR')";
16    $result = mysql_query($query, $conn)
17    or ErrorMessage('덧글을 저장하는 데 실패하였습니다.');
18 ?>
19 <meta http-equiv='Refresh' content='1; URL=read.php?id=<?=$_GET[id]?>'>
20 <center>덧글이 등록되었습니다.</center>
```

일반형 게시판의 insert.php 파일에 필수 항목 검증 부분과 데이터베이스 에러 메시지 처리 부분을 추가하면 위와 같은 코드를 얻을 수 있습니다. 또한 이름 값은 HTML 태그를 사용하지 못하도록 태그를 제거합니다.

실제로 덧글을 몇 개 등록해 봅시다.



[그림 13-56] 덧글의 등록

덧글 두 개를 위와 같이 등록했으나 아직 확인할 길이 없습니다. 데이터베이스에 직접 쿼리를 던져서 확인할 수도 있지만 덧글 목록을 만들어서 직접 확인해보는 것이 더 좋을 듯합니다.

덧글 목록 및 보기

덧글의 목록을 만들어 봅시다. 덧글의 목록은 앞서 언급한 대로 일반형 게시판의 기본 틀을 따르지만 페이징이 없고 방명록과 마찬가지로 글 목록이 글 읽기 페이지를 대신합니다. 따라서 단순하게 comment 테이블에서 관련 덧글을 가져와서 모두 보여주기만 하면 덧글 목록 부분이 완성됩니다.

우선 덧글 목록을 위한 쿼리를 작성해 봅시다. 덧글의 목록은 일반형 게시판에서 모든 글을 목록으로 만들어 보여주던 것과는 달리 해당 글에 연관된 덧글만을 목록으로 보여줘야 합니다. 그래서 덧글 테이블에는 그 관계를 표시하기 위해서 bid라는 필드를 추가했습니다. 바로 이 bid가 일반 글의 id가 됩니다. 따라서 다음과 같이 bid가 id와 같은 덧글만을 목록으로 출력합니다.

```
SELECT * FROM comment WHERE bid = '$id' ORDER BY id
```

덧글은 일반형 게시판에서와 마찬가지로 DESC형으로 등록됩니다. 즉, 최근에 등록된 글이 큰 번호를 가지게 됩니다. 그런데 게시판의 목록과는 다르게 정렬을 DESC로 하지 않고 ASC로 순차정렬을 하고 있습니다. 그 이유는 덧글의 경우 먼저 등록한 사람을 먼저 보여주는 관례 때문입니다. 이것으로 인해서 빨리 덧글을 등록하려고 순위 다툼을 하는 속칭 등수놀이라는 것이 생겨나게 되었습니다.

이제 쿼리가 완성되었으니 이 쿼리를 while 문으로 반복하면서 목록을 출력하면 됩니다. 단, 덧글은 방명록과 마찬가지로 리스트에 덧글의 내용(comment)을 출력하도록 합니다.

디자인을 제외하고 작성된 코드는 다음과 같습니다.

```
<?
    include_once "db_info.php";
    include_once "library.php";
    $id = $_GET[id];
    $sql_cmt = "SELECT * FROM comment WHERE bid = '$id' ORDER BY id";
    $result_cmt = mysql_query($sql_cmt, $conn);

    while($row_cmt=mysql_fetch_array($result_cmt)) {
        $comment = nl2br(str_replace(" "," ",
        remove_html($row_cmt[comment])));
?>

  <?=$row_cmt[name]?>
  <?=$row_cmt[wdate]?>
  <?=$comment?>

<?  } ?>
```

덧글의 내용은 계층형 게시판에서와 마찬가지로 HTML을 모두 태그 그대로 보여주고 줄 바꿈과 띄어쓰기를 복원시켜 줍니다. 이제 디자인을 입히고 적절한 위치에 이름, 등록된 날짜 그리고 덧글의 내용을 출력해주면 됩니다. 여기에 덧붙여 덧글의 수정 삭제를 위하여 다음과 같은 링크를 추가합

니다.

```
<a href="comment_edit.php?id=<?=$row_cmt[id]?>&bid=<?=$id?>">[수정]</a>
<a href="comment_predel.php?id=<?=$row_cmt[id]?>&bid=<?=$id?>">[삭제]</a>
```

여기서 주의할 점은 일반 글의 id와 덧글의 id를 혼동하지 말아야 한다는 것입니다. 덧글을 수정하기 위한 id는 수정의 대상이 되는 덧글의 id가 됩니다. 여기에 수정이나 삭제 후 다시 글 읽기 페이지로 되돌아오기 위해서 일반 글의 id를 bid라는 이름으로 전달하는 것입니다.

디자인을 추가하면 다음과 같습니다.

```
<?
    include_once "db_info.php";
    include_once "library.php";
    $id = $_GET[id];
    $sql_cmt = "SELECT * FROM comment WHERE bid = '$id' ORDER BY id";
    $result_cmt = mysql_query($sql_cmt, $conn);

    while($row_cmt=mysql_fetch_array($result_cmt)) {
        $comment = nl2br(str_replace(" "," " ,
        remove_html($row_cmt[comment])));
?>

<table width=98% border=0 align=center cellpadding=5 cellspacing=0>
  <tr bgcolor=#CCCCCC><td colspan=2></td></tr>
  <tr bgcolor=#F0F0F0>
    <td width=50%><?=$row_cmt[name]?></td>
    <td align=right style="font-size:8pt">

      <a href="comment_edit.php?id=<?=$row_cmt[id]?>"
      style="font-size:8pt;color:#999999">[수정]</a>
      <a href="comment_predel.php?id=<?=$row_cmt[id]?>"
      style="font-size:8pt;color:#999999">[삭제]</a>
      <?=$row_cmt[wdate]?>
    </td>
  </tr>
  <tr>
    <td valign=top colspan=2><?=$comment?><BR><BR></td>
  </tr>
</table>
<?  } ?>
```

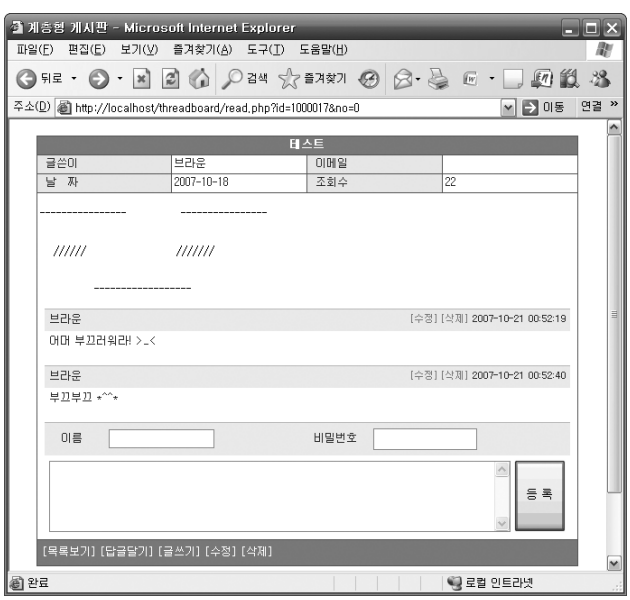
이 코드를 앞서 덧글 쓰기 폼에서 작성한 comment.php 파일의 제일 윗 부분에 추가하면 됩니다. 최종으로 완성된 comment.php 파일은 다음과 같습니다.

Comment.php

```
<?
include_once "db_info.php";
include_once "library.php";

$id = $_GET[id];
$sql_cmt = "SELECT * FROM comment WHERE bid = '$id' ORDER BY id";
$result_cmt = mysql_query($sql_cmt, $conn);

while($row_cmt=mysql_fetch_array($result_cmt)) {
  $comment = nl2br(str_replace(" "," " ,
  remove_html($row_cmt[comment])));
?>
<table width=98% border=0 align=center cellpadding=5 cellspacing=0>
<tr bgcolor=#CCCCCC><td colspan=2></td></tr>
<tr bgcolor=#F0F0F0>
  <td width=50%><?=$row_cmt[name]?></td>
  <td align=right style="font-size:8pt">
  <a href="comment_edit.php?id=<?=$row_cmt[id]?>&bid=<?=$id?>"
  style="font-size:8pt;color:#999999">[수정]</a>
  <a href="comment_predel.php?id=<?=$row_cmt[id]?>&bid=<?=$id?>"
  style="font-size:8pt;color:#999999">[삭제]</a>
  <?=$row_cmt[wdate]?>
  </td>
</tr>
<tr>
  <td valign=top colspan=2><?=$comment?><BR><BR></td>
</tr>
</table>
<? } ?>
<script>
function CommentFormCheck() {
  if (!comment_insert.name.value) {
    alert("이름을 입력하세요.");
    comment_insert.name.focus();
    return false;
  }
  if (!comment_insert.pass.value) {
    alert("비밀번호를 입력하세요.");
    comment_insert.pass.focus();
    return false;
  }
  if (!comment_insert.comment.value) {
    alert("내용을 입력하세요.");
    comment_insert.comment.focus();
    return false;
```

```
46    }
47 }
48 </script>
49 <table width=98% align=center border=0 cellpadding=5 cellspacing=0>
50 <form name=comment_insert method=post action='comment_insert.php'
51 onsubmit="return CommentFormCheck()">
52 <input type=hidden name=bid value='<?=$id?>'>
53 <tr bgcolor=#CCCCCC><td colspan=4></td></tr>
54 <tr bgcolor=#F0F0F0>
55   <td width=50 align=center>이름</td><td width=100>
56   <input type=text name=name size=15></td>
57   <td width=50 align=center>비밀번호</td>
58   <td><input type=password name=pass size=15></td>
59 </tr>
60 <tr>
61   <td valign=absmiddle colspan=4><textarea name=comment rows=5
62   cols=70></textarea>
63   <input type=submit value=' 등록 ' style="height:80;width:57">
64   </td>
65 </tr>
66 </form>
```

이제 앞서 등록한 덧글이 목록으로 잘 보이는지 확인해 봅시다.

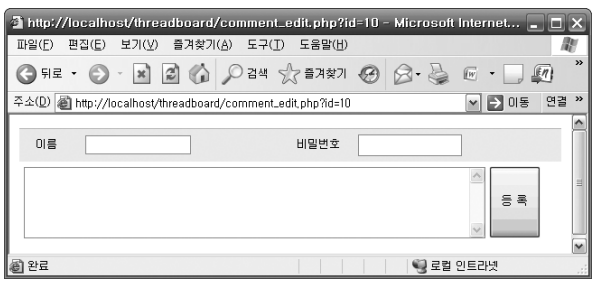


[그림 13-57] 덧글 목록 확인

덧글이 제대로 잘 등록되고 또한 목록도 잘 나오는군요. 기세를 몰아서 수정과 삭제도 서둘러 만들어 봅시다.

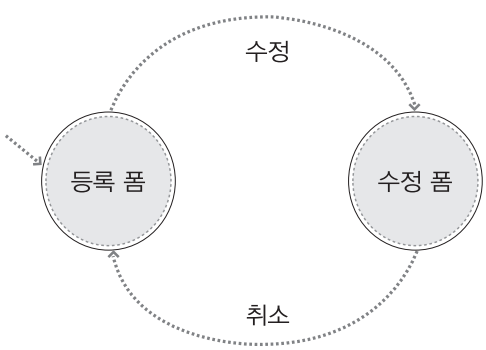
덧글 수정 폼

덧글을 수정해 봅시다. 덧글 수정은 일반형 게시판이나 계층형 게시판에서 edit.php 파일을 가져와서 수정해서 간단히 만들어 볼 수 있습니다. 그런데 다음과 같이 덩그러니 수정 폼을 쓰기에는 뭔가 허전합니다.



[그림 13-58] 덧글 쓰기 폼을 이용한 덧글의 수정 폼

그래서 만약 기존의 덧글 쓰기 폼을 그대로 이용할 수 있으면 어떨까? 하고 생각했습니다. 게시판은 따로 글 쓰기 폼 페이지가 있어서 그러지 못했지만 덧글은 글 쓰기 폼이 항상 노출되어 있으므로 이 글 쓰기 폼을 약간 수정하여 그대로 사용할 수 없을까 하는 생각이었습니다. 고민해보니 자바스크립트를 이용하여 간단히 수정할 수 있습니다. 이왕 이렇게 된 거 자바스크립트도 배워볼 겸해서 뚝딱뚝딱 만들어 보겠습니다.



[그림 13-59] 등록 폼과 수정 폼의 전환

등록 폼은 기본적으로 글 목록에 보입니다. 그러나 덧글의 수정 버튼을 클릭하면 등록 폼은 수정 폼으로 변경됩니다. 수정을 취소하고 싶을 때는 수정 폼의 취소 버튼을 클릭하면 다시 원래의 등록 폼

으로 변경됩니다.

이처럼 구현하려면 자바스크립트를 이용한 동적인 웹 페이지로 구성해야 합니다. 그래서 두 개의 자바스크립트 함수와 몇 군데 소스를 추가하여 동적인 웹 페이지로 변모시키고자 합니다.

① 수정 폼으로 변경하는 자바스크립트 함수

② 취소 버튼을 클릭한 경우 원래대로 되돌려주는 자바스크립트 함수

③ 기타 정보를 가져오거나 동적인 웹 페이지를 구성하기 위한 추가 코드

우선 수정 폼으로 변경하는 자바스크립트 함수를 작성하겠습니다. 자바스크립트 함수와 함께 기존 소스에 HTML과 PHP 소스를 약간 추가합니다. 일단 어떤 절차로 수정 폼을 변경하는지 알아보겠습니다.

가. 폼에 기존 덧글의 내용을 출력한다.

나. 등록 버튼을 수정 버튼으로 바꾼다.

다. 취소 버튼을 추가한다.

라. 폼의 action 값을 comment_update.php로 변경한다.

마. 덧글 내용 입력부로 포커스를 준다.

수정 버튼을 클릭하면 폼에 기존 덧글의 내용을 출력해야 합니다. 이때 기존의 게시판에서는 데이터베이스에서 글의 정보를 가져와서 출력했으나 덧글은 이미 목록에 필요한 정보가 모두 출력되어 있으니 이를 가져와서 수정 폼에 보여주기로 합니다.

이 방법을 사용하려면 웹 페이지에 출력된 값을 가져다가 수정 폼에 출력해주는 부분을 작성해야 합니다. 웹 페이지에 출력된 값을 가져오는 방법은 이름을 부여하는 방법을 사용합니다. 〈TABLE〉, 〈TD〉, 〈DIV〉, 〈SPAN〉 등과 같은 태그에는 이름(ID)을 지정할 수 있습니다.

```
<TD ID="tab1">뇌를 자극하는 PHP 프로그래밍</TD>
<DIV ID="title">PHP 세상에 오신 것을 환영합니다.</DIV>
```

위와 같은 방법으로 지정된 ID는 자바스크립트를 통해서 접근할 수 있습니다. 즉, 이름이 부여되지 않은 항목에 대해서는 구체적으로 특정 부분의 값에 접근하는 것이 불가능합니다.

```
document.all.tab1
document.all.title
```

위와 같이 접근할 수 있으며 직접적으로 tab1이나 title처럼 간략하게도 사용할 수 있습니다. 하지만 이름은 반드시 문서 전체에서 중복되지 않도록 지정해야 합니다. 만약 중복된 이름이 존재하면 구분을 못 하게 되어 사용이 불가능해질 수 있습니다.

이렇게 이름을 부여하여 웹 페이지의 특정 부분에 접근하면 됩니다. 그러면 이제 본격적으로 문자

열을 가져와 볼까요? 이와 관련된 자바스크립트 함수는 다음과 같습니다.

자바스크립트 함수	기능
innerText	시작 태그와 닫는 태그 사이의 내용을 단순 문자로 처리
innerHTML	시작 태그와 닫는 태그 사이의 내용을 HTML로 처리
outerText	시작 태그와 닫는 태그를 포함하여 내용을 단순 문자로 처리
outerHTML	시작 태그와 닫는 태그를 포함하여 내용을 HTML로 처리

[표 13-11] 관련 자바스크립트 함수

위의 네 가지 함수들은 모두 이름으로 지정된 부분의 내용을 가져오거나 반대로 내용을 지정하여 변경할 수 있습니다.

```
alert(title.innerText);
```

위와 같은 방법으로 사용하면 title에 지정된 문자열을 경고창으로 출력합니다. 반면에,

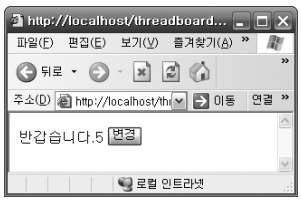
```
title.innerText = "PHP 프로그래밍";
```

위와 같이 사용하면 원래 문장이 "PHP 프로그래밍"으로 바뀝니다.

함수를 잘 살펴보면 inner, outer 부류와 Text, HTML 부류로 구분할 수 있는 것을 알 수 있습니다. inner 부류는 이름을 지정한 태그의 사이에 있는 내용을 가져오거나 변경할 수 있습니다. 반면에 outer 부류는 이름을 지정한 태그를 포함하여 그 사이에 있는 내용을 가져오거나 변경할 수 있습니다. 이 차이는 다음과 같습니다.

```
<SPAN ID="test">안녕하세요.</SPAN>
<input type="button" value="변경" onclick="change();">
<script>
  var i=0;
  function change() {
     i++;
     test.innerText = "반갑습니다." + i;
     test.outerText = "반갑습니다." + i;
  }
</script>
```

위와 같은 코드가 있을 때 변경 버튼을 누르면 자바스크립트의 change 함수가 호출됩니다. 우선 innerText의 주석을 제거하고 실행을 해보면 "안녕하세요"가 "반갑습니다.1"로 변경되었음을 알 수 있습니다. 한번 더 변경 버튼을 눌러 보면 "반갑습니다.2"라는 값을 보게 되고 계속 변경 버튼을 누르면 숫자는 계속해서 올라가는 것을 볼 수 있습니다.



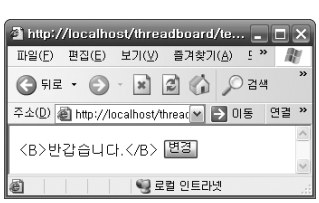
[그림 13-60] innerText의 실행결과

그러나 outerText의 주석을 제거하고 다시 innerText를 주석을 씌우면 처음에는 잘 변경되는 듯싶지만 계속 버튼을 눌렀을 때 자바스크립트 에러가 나타나고 더 이상 숫자가 증가하지 않는 것을 알 수 있습니다.

[그림 13-61] outerText의 실행결과

이러한 현상이 나타나는 이유는 inner 부류는 〈SPAN ID="test"〉안녕하세요.〈/SPAN〉 태그에서 〈SPAN〉태그 사이에 있는 값만을 변경했지만 outer 부류는 〈SPAN〉태그까지 포함해서 수정을 해버렸기 때문입니다. 그래서 두 번째 시도에서는 이미 〈SPAN ID="test"〉안녕하세요.〈/SPAN〉 전체가 "반갑습니다.1"로 변경되어 더 이상 〈SPAN〉 태그가 존재하지 않으므로 에러가 발생한 것입니다.

두 번째로 Text 부류와 HTML 부류는 이름에서도 알 수 있듯이 내용을 단순한 Text로 다룰 것인지 아니면 HTML로 다룰 것인지를 의미합니다. innerText로 내용을 가져오면 그 사이에 존재하는 모든 HTML은 다 무시되고 문자열만 가져옵니다. 반면에 innerHTML로 내용을 가져오면 시작과 닫는 태그 사이에 존재하는 모든 HTML 태그가 HTML 형태 그대로 가져옵니다. 만약 innerText를 이용하여 HTML 태그가 포함된 문자열로 변경하면 HTML 태그가 사라지는 것이 아니라 모든 태그가 태그 글자 그대로 출력됩니다. 마치 게시판에서 본문 내용을 보여주기 위해서 태그를 문자로 변환하는 것과 동일한 효과를 갖습니다. 그 반면에 innerHTML은 HTML 태그가 모두 적용된 상태로 변경됩니다.



[그림 13-62] innerText의 결과

[그림 13-63] innerHTML의 결과

따라서 제목과 내용을 출력하는 부분에 ID를 설정하고 innerText 함수를 이용하여 내용을 가져와서 수정 폼에 출력하면 됩니다. 그러면 일단 제목과 내용에 ID를 설정합니다.

```
<td width=50% id="n_<?=$row_cmt[id]?>"><?=$row_cmt[name]?></td>
<td valign=top colspan=2 id="c_<?=$row_cmt[id]?>"><?=$comment?><BR><B
R></td>
```

이처럼 이름을 설정할 때 덧글의 번호를 함께 부여해서 여러 개의 덧글 중에 어느 덧글의 내용을 가져올지 판단할 수 있게 합니다. 즉, n_1, c_1과 같은 이름으로 접근할 수 있게 됩니다.

```
<script>
function change_form(id) {
  name = eval("n_" + id).innerText;
  comment = eval("c_" + id).innerText;

  comment_insert.name.value = name;
  comment_insert.comment.value = comment;
  comment_insert.id.value = id;
  comment_insert.action = "comment_update.php";
}
</script>
```

수정 버튼을 클릭하면 change_form(덧글 번호)과 같은 방법으로 함수가 호출됩니다. 이때 넘어오는 덧글 번호를 이용하여 n_1, c_1과 같은 객체를 만들어야 합니다. PHP에서 가변변수와 같이 eval 함수를 이용하여 조합한 문자열을 해당 객체로 인식하게 할 수 있습니다. 이를 통해서 가져온 값은 글 쓰기 폼에 각각 대입합니다.

여기서 수정할 번호인 id 값을 전달해야 합니다. 그런데 글 쓰기 폼에는 id란 값이 필요 없었기 때문에 숨겨진 필드가 존재하지 않습니다. 따라서 다음과 같이 숨겨진 필드를 글 쓰기 폼에 추가해야 합니다.

```
<form name=comment_insert method=post action='comment_insert.php'
onsubmit="return CommentFormCheck()">
<input type=hidden name=id value=''>
<input type=hidden name=bid value='<?=$id?>'>
```

또한 change_form 함수에 다음과 같은 코드를 추가합니다.

```
comment_insert.comment.cols=61;
comment_insert.register.value="수 정";
str="<input type=button value=' 취 소 ' ";
str +="style='height:80;width:57' onclick='restore_form()'>";
cancel.innerHTML=str;
comment_insert.comment.focus();
```

아울러 덧글 등록하기에서 덧글 수정하기로 변경되었으므로 폼의 action을 comment_update.php 파일로 수정합니다.

등록 버튼을 수정 버튼으로 변경하기 위하여 다음과 같이 submit 버튼에 register라는 이름을 부여합니다.

```
<input name=register type=submit value=' 등 록 '
style="height:80;width:57">
```

register라는 이름으로 이 버튼의 값을 수정으로 변경합니다.

```
comment_insert.register.value="수 정";
```

취소 버튼을 추가로 보여주기 위하여 덧글 내용 입력부의 가로 길이를 줄여줍니다.

```
comment_insert.comment.cols=61;
```

줄어든 공간에 취소 버튼을 추가하려면 앞서 배운 innerHTML 함수를 사용해야 합니다. 그래서 cancel이라는 이름의 공간을 등록 버튼 옆에 만들어 줍니다.

```
<input name=register type=submit value='등록' style="height:80;width:57">
<span id=cancel></span>
```

이제 이 cancel 공간에 innerHTML을 이용해서 취소 버튼을 동적으로 추가할 수 있습니다.

```
str = "<input type=button value=' 취 소 '
str += "style='height:80width:57'onclick='restore_form()'>";
cancel.innerHTML = str;
```

마지막으로 내용 입력부에 포커스를 주어 쉽게 수정할 수 있도록 도와줍니다.

```
comment_insert.comment.focus();
```

만약 포커스를 주지 않으면 덧글이 수 십개 이상 달렸을 경우 첫 번째 덧글을 수정하기 위해서 수정 버튼을 누르고 한참을 아래로 스크롤해야만 합니다. 혹은 처음 사용해보는 사용자는 수정이 안 된다고 계속해서 수정 버튼을 누르고 있을지도 모릅니다. 그러나 이렇게 포커스를 주면 자동으로 스크롤이 되기 때문에 이러한 우려를 지울 수 있습니다.

완성된 자바스크립트 함수는 다음과 같습니다.

```
function change_form(id) {
  name = eval("name_" + id).innerText;
  comment = eval("comment_" + id).innerText;

  comment_insert.name.value = name;
  comment_insert.comment.value = comment;
  comment_insert.id.value = id;
  comment_insert.action = "comment_update.php";
  comment_insert.comment.cols=61;
  comment_insert.register.value="수 정";
  str="<input type=button value=' 취소 '";
  str +=style='height:80;width:57' onclick='restore_form()'>";
  cancel.innerHTML = str;
  comment_insert.comment.focus();
}
```

이제 수정을 취소하는 경우 폼을 원상태로 되돌리는 함수를 구현해 보겠습니다. 취소 버튼을 누르면 restore_form이라는 함수가 호출됩니다. 이 함수가 해야 할 일은 폼에 입력된 값을 모두 초기화시키고 폼을 원상복귀시키며 action 값을 원상태로 되돌려주는 것입니다. 앞서 폼을 변경하면서 다루었던 것이므로 보면 쉽게 이해할 수 있습니다.

```
function restore_form() {
  cancel.innerHTML = "";
  comment_insert.action = "comment_insert.php";
  comment_insert.id.value = "";
  comment_insert.reset();
  comment_insert.register.value="등 록";
  comment_insert.comment.cols=70;
}
```

최종 수정된 comment.php 파일의 소스는 다음과 같습니다.

Comment.php

```
1  <?
2  include_once "db_info.php";
3  include_once "library.php";
4
5  $id = $_GET[id];
6  $sql_cmt = "SELECT * FROM comment WHERE bid = '$id' ORDER BY id";
7  $result_cmt = mysql_query($sql_cmt, $conn);
8
9  while($row_cmt=mysql_fetch_array($result_cmt)) {
```

```
10    $comment = nl2br(str_replace(" "," " ,
11    remove_html($row_cmt[comment])));
12  ?>
13  <table width=98% border=0 align=center cellpadding=5 cellspacing=0>
14  <tr bgcolor=#CCCCCC><td colspan=2></td></tr>
15  <tr bgcolor=#F0F0F0>
16    <td width=50% id="name_<?=$row_cmt[id]?>"><?=$row_cmt[name]?></td>
17    <td align=right style="font-size:8pt">
18      <a href="javascript:change_form(<?=$row_cmt[id]?>);"
19      style="font-size:8pt;color:#999999">[수정]</a>
20      <a href="comment_predel.php?id=<?=$row_cmt[id]?>&bid=<?=$id?>"
21      style="font-size:8pt;color:#999999">[삭제]</a>
22      <?=$row_cmt[wdate]?>
23    </td>
24  </tr>
25  <tr>
26  <td valign=top colspan=2 id="comment_<?=$row_cmt[id]?>">
27  <?=$comment?><BR><BR></td>
28  </tr>
29  </table>
30  <? } ?>
31  <script>
32  function change_form(id) {
33    name = eval("name_" + id).innerText;
34    comment = eval("comment_" + id).innerText;
35    comment_insert.name.value = name;
36    comment_insert.comment.value = comment;
37    comment_insert.id.value = id;
38    comment_insert.action = "comment_update.php";
39    comment_insert.comment.cols=61;
40    comment_insert.register.value="수 정";
41    str = "<input type=button value=' 취 소 '";
42    str += "style='height:80;width:57' onclick='restore_form()'>";
43    cancel.innerHTML = str;
44    comment_insert.comment.focus();
45  }
46  function restore_form() {
47    cancel.innerHTML = "";
48    comment_insert.action = "comment_insert.php";
49    comment_insert.id.value = "";
50    comment_insert.reset();
51    comment_insert.register.value="등 록";
52    comment_insert.comment.cols=70;
53  }
54  function CommentFormCheck() {
55    if (!comment_insert.name.value) {
56      alert("이름을 입력하세요.");
```

```
57      comment_insert.name.focus();
58      return false;
59    }
60    if (!comment_insert.pass.value) {
61      alert("비밀번호를 입력하세요.");
62      comment_insert.pass.focus();
63      return false;
64    }
65    if (!comment_insert.comment.value) {
66      alert("내용을 입력하세요.");
67      comment_insert.comment.focus();
68      return false;
69    }
70 }
71 </script>
72 <table width=98% align=center border=0 cellpadding=5 cellspacing=0>
73 <form name=comment_insert method=post action='comment_insert.php'
74 onsubmit="return CommentFormCheck()">
75 <input type=hidden name=id value=''>
76 <input type=hidden name=bid value='<?=$id?>'>
77 <tr bgcolor=#CCCCCC><td colspan=4></td></tr>
78 <tr bgcolor=#F0F0F0>
79   <td width=50 align=center>이름</td><td width=100>
80   <input type=text name=name size=15></td>
81   <td width=50 align=center>비밀번호</td>
82   <td><input type=password name=pass size=15></td>
83 </tr>
84 <tr>
85   <td valign=absmiddle colspan=4><textarea name=comment rows=5
86   cols=70></textarea>
87   <input name=register type=submit value=' 등 록 '
88   style="height:80;width:57">
89   <span id=cancel></span>
90   </td>
91 </tr>
92 </form>
93 </table>
```

덧글 수정

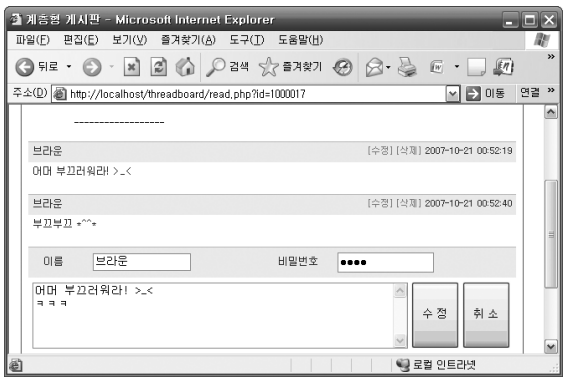
매우 어수선했지만 덧글 수정하기 폼을 모두 작성했으니 수정된 덧글을 저장하는 페이지를 만들어 보겠습니다. 수정하기는 이제 매우 익숙하니 쉽게 구현할 수 있습니다. 계층형 게시판의 update.php 파일을 바탕으로 약간만 수정하면 됩니다.

Comment_update.php

```
<?
include_once "library.php";

$_POST[name] = trim(strip_tags($_POST[name]));

//입력값 검증
if (!$_POST[name]) ErrorMessage('이름을 입력하세요.');
if (!$_POST[pass]) ErrorMessage('암호를 입력하세요.');
if (!$_POST[comment]) ErrorMessage('내용을 입력하세요.');

//데이터 베이스 연결하기
include_once "db_info.php";

// 글의 비밀번호를 가져온다.
$query = "SELECT pass FROM comment WHERE id='$_POST[id]'";
$result=mysql_query($query, $conn)
or ErrorMessage('덧글을 수정하는데 실패하였습니다.');
$row=mysql_fetch_array($result);

//입력된 값과 비교한다.
if ($_POST[pass]==$row[pass]) { //비밀번호가 일치하는 경우
  $query = "UPDATE comment SET name='$_POST[name]',";
  $query .= "comment='$_POST[comment]' WHERE id='$_POST[id]'";
  $result=mysql_query($query, $conn)
  or ErrorMessage('덧글을 수정하는데 실패하였습니다.');
}
else { // 비밀번호가 일치하지 않는 경우
  ErrorMessage('비밀번호가 틀립니다.');
}

//데이터베이스와의 연결 종료
mysql_close($conn);

//수정하기인 경우 수정된 글로..
echo ("<meta http-equiv='Refresh' content='0;
URL=read.php?id=$_POST[bid]'>");
?>
```

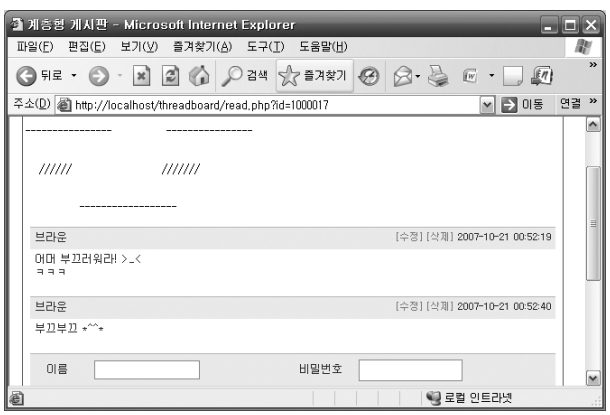
수정된 부분은 테이블 이름을 comment로 변경했고 덧글의 스키마에 맞게 수정 항목을 변경해 주었습니다.

제대로 동작하는지 확인을 해봅시다.



[그림 13-64] 덧글의 수정

첫 번째 덧글에서 수정하기 버튼을 눌러 위와 같이 수정한 후 수정 버튼을 클릭합니다.



[그림 13-65] 덧글이 수정된 결과

그 결과 위와 같이 수정된 사항이 변경된 것을 알 수 있습니다.

덧글 삭제

이제 덧글 기능의 마지막으로 삭제를 해봅시다. 삭제는 앞서 제작한 predel.php와 del.php 파일을 참조하여 만들어볼 생각입니다. 일단 비밀번호 입력부는 predel.php 파일을 그대로 가져와서 action만 수정하면 모두 끝납니다.

Comment_predel.php

```
1  <html>
2  <head>
3  <title>계층형 게시판</title>
```

```
<style>
<!--
td { font-size : 9pt; }
A:link { font : 9pt;color : black;text-decoration : none;
font-family: 굴림;font-size : 9pt; }
A:visited { text-decoration : none; color : black;
font-size: 9pt; }
A:hover { text-decoration : underline; color : black;
font-size: 9pt; }
-->
</style>
<script>
function FormCheck() {
  if (!fm.pass.value)
  {
    alert("암호를 입력하세요.");
    fm.pass.focus();
    return false;
  }
}
</script>
</head>

<body topmargin=0 leftmargin=0 text=#464646>
<center>
<BR>
<!-- 입력된 값을 다음 페이지로 넘기기 위해 FORM을 만든다. -->
<form action="comment_del.php?id=<?=$_GET[id]?>&bid=<?=$_GET[bid]?>"
method=post name=fm onsubmit="return FormCheck()">
<table width=300 border=0 cellpadding=2 cellspacing=1
bgcolor=#777777>
<tr>
  <td height=20 align=center bgcolor=#999999>
    <font color=white><B>비 밀 번 호 확 인</B></font>
  </td>
</tr>
<tr>
  <td align=center >
  <font color=white><B>비밀번호 : </b>
  <INPUT type=password name=pass size=8 maxlength=8>
  <INPUT type=submit value="확 인">
  <INPUT type=button value="취 소" onclick="history.back(-1)">
  </td>
</tr>
</table>
```

수정된 결과는 다음과 같습니다.

[그림 13-66] 덧글 삭제를 위한 비밀번호 확인

이제 여기서 입력받은 비밀번호를 확인하고 삭제하는 부분을 만들어 봅시다. 데이터베이스에서 삭제하는 부분은 del.php에서 테이블의 이름과 되돌아가는 페이지를 약간 수정하면 됩니다.

Comment_del.php

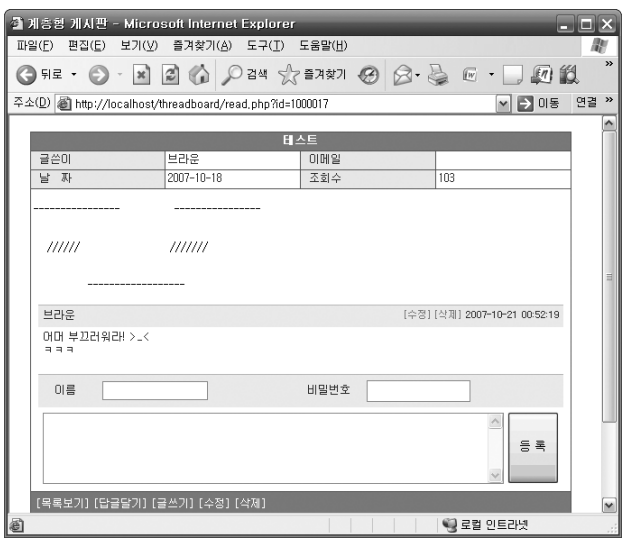
```
<?
include_once "library.php";

//입력값 검증
if (!$_POST[pass]) ErrorMessage('암호를 입력하세요.');

//데이터 베이스 연결하기
include_once "db_info.php";

$result=mysql_query("SELECT pass FROM comment WHERE id=$_GET[id]",
$conn) or ErrorMessage('덧글을 삭제하는데 실패하였습니다.');;
$row=mysql_fetch_array($result);

if ($_POST[pass]==$row[pass] )
{
  $conndel = "DELETE FROM comment WHERE id=$_GET[id] ";
  $result=mysql_query($conndel, $conn)
    or ErrorMessage('덧글을 삭제하는데 실패하였습니다.');;
}
else
{
  ErrorMessage('비밀번호가 틀립니다.');
}
?>
<meta http-equiv='Refresh' content='0; URL=read.php?id=<?=$_GET[bid]?>'>
```

기존의 소스를 그대로 이용하므로 삭제하기는 매우 쉽게 구현할 수 있습니다. 수정된 소스가 제대

로 동작하는지 확인해 보겠습니다.

[그림 13-67] 덧글 삭제 테스트

비밀번호를 올바르게 입력하면 다음과 같이 덧글이 삭제되었음을 확인할 수 있습니다.



[그림 13-68] 삭제된 덧글

덧글의 개수 출력

덧글의 모든 기능을 구현했습니다. 덧글의 등록부터 수정, 삭제까지 기존의 일반형 게시판 소스를 이용하여 쉽게 구현할 수 있었습니다. 단, 수정 부분은 자바스크립트를 이용했기에 조금 생소했을 것 같습니다.

이제 게시판의 글 목록에서 해당 글에 덧글이 몇 개 등록되어 있는지를 표시하여 봅시다. 최근에는 이런 덧글이 몇 개 등록되어 있느냐에 따라서 조회 수가 급격히 차이나는 경우가 많습니다.

덧글의 수를 알아내기 위해서는 다음과 같은 쿼리를 사용하면 됩니다.

```
SELECT COUNT(*) FROM comment WHERE bid=$id
```

그러나 글 목록을 출력하기 위해서 while 문이 반복하는 곳에서 다시 새로운 쿼리를 던져 덧글의 수를 가져오는 것은 번거로울 뿐만 아니라 처리 속도도 느려지게 됩니다. 그래서 서브 쿼리를 이용하여 이 쿼리까지 하나로 묶어보려고 합니다.

일단 기존의 목록을 위한 쿼리입니다.

```
SELECT * FROM $board
WHERE thread >= (
  SELECT thread FROM $board
  ORDER bY thread
  LIMIT $no,1
)
ORDER BY thread
LIMIT $page_size;
```

이 쿼리와 덧글의 수를 세는 쿼리를 합쳐야 하는데 어떻게 조합을 해야 할까요? 조금 어려운 문제가 될 수 있으나 답을 보면 의외로 매우 간단한 것을 알 수 있습니다.

```
SELECT *, (SELECT COUNT(*) FROM comment WHERE bid=board.id) as cnt
FROM $board as board
WHERE thread >= (
  SELECT thread FROM $board
  ORDER bY thread
  LIMIT $no,1
)
ORDER BY thread
LIMIT $page_size;
```

덧글의 수를 세는 쿼리를 단순히 검색 항목에 추가해주면 됩니다. 이때 서브 쿼리에서 메인 쿼리의 테이블 값을 참조해야 하는데 as 키워드를 통해서 짧게 board라는 별명을 만들고 접근하면 됩니다. 그렇게 되면 글 목록을 가져오면서 해당하는 글의 덧글 수를 세고 그 결과는 cnt라는 이름으로 접근할 수 있게 됩니다.

생각보다 쉽지 않나요? 물론 처음에 이런 쿼리를 만드는 것은 매우 어려운 일입니다. 하지만 막상 쿼리를 보면 쉽게 이해할 수 있듯이 자주 쿼리를 다루거나 남의 쿼리를 자주 보면 더 복잡한 쿼리도 쉽게 작성할 수 있게 됩니다.

이제 글 목록 쿼리를 위와 같이 수정해 줍니다.

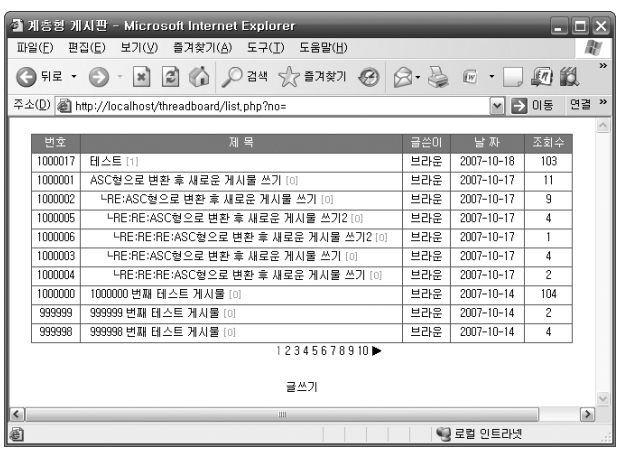
```
$query = "
       SELECT * , (SELECT COUNT(*) FROM comment WHERE bid = board.id)
                  as cnt FROM $board as board
       WHERE thread >= (
            SELECT thread FROM $board
            ORDER BY thread
```

```
            LIMIT $no,1
            )
        ORDER BY thread
        LIMIT $page_size;
    ";
$result = mysql_query($query, $conn) or
ErrorMessage('글 목록을 가져오는 도중 오류가 발생하였습니다.', false);
```

그리고 다음과 같이 글의 제목 옆에 조그맣게 덧글의 수를 표시합니다.

```
<a href=read.php?id=<?=$row[id]?>&no=<?=$no?>>
<?=remove_html($row[title]);?>
<FONT style="font-size:8pt" COLOR="orange">[<?=$row[cnt];?>]</FONT></a>
```

이제 수정된 결과를 확인해 봅시다.



[그림 13-69] 덧글의 수가 표시된 글의 목록 페이지

가장 위의 글에 1이라고 조그맣게 덧글의 수가 표시되는 것을 확인할 수 있습니다.

이렇게 최근 트랜드인 덧글 기능을 추가했습니다. 이미 구현을 통해서 느꼈듯이 덧글은 계층형 게시판에 기생하는 게시판이라고 생각할 수 있습니다. 게시판의 모양이 조금 다를 뿐 모든 기능이 게시판을 기초로 하고 있습니다. 그렇기 때문에 일반형 게시판의 소스를 대부분 그대로 차용해서 사용할 수 있었습니다. 실제로 덧글 뿐만이 아니라 대부분의 웹 프로그램이 목록, 추가, 수정, 삭제에 기반을 두고 있기 때문에 일반형 게시판 소스를 약간 변형하여 프로그램을 작성하는 경우가 대부분입니다. 이제 점점 웹 프로그램이 별것 아니라는 생각이 드나요?

파일 업로드

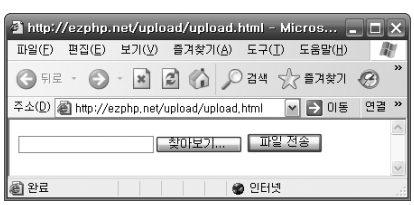
PHP에서는 별도의 컴포넌트 없이 자체적으로 파일 업로드가 가능합니다. 비록 몇 가지 제약이 있긴 하지만 성능의 큰 저하 없이 자체적으로 업로드를 제공한다는 것은 얼마나 고마운 일인지 모릅니다. 만약 PHP가 파일 업로드를 지원하지 않는다면 ActiveX 컨트롤을 이용하여 파일을 업로드해야 합니다. 여기서는 파일 업로드 방법에 대해서 배우고 이를 게시판에 적용하여 게시판에서 파일을 보관할 수 있도록 해봅시다.

파일 업로드용 폼

우리는 페이지와 페이지 간에 정보를 교환하기 위해서 폼(Form)을 이용했습니다. 파일 업로드 또한 이 폼을 이용하여 업로드를 수행하는데 기존의 폼과는 다르게 반드시 추가해야 하는 부분이 있습니다.

[예제 13-1] 업로드된 파일의 정보 얻기

```
1  <form enctype="multipart/form-data" action="주소" method="post">
2    <input name="upfile" type="file">
3    <input type="submit" value="파일 전송">
4  </form>
```



[그림 13-70] 파일 업로드 폼

파일을 업로드하려면 두 가지 설정을 반드시 해주어야 합니다. 먼저 method를 POST로 사용해야 하고, 다음으로 인코딩 형식을 multipart/form-data로 설정해야 합니다. 폼 다루기에서 이미 설명했듯이 폼 정보와 파일 두 가지 형식의 정보를 전달하려면 multipart/form-data 형식의 인코딩이 필수입니다.

```
<input name="upfile" type="file">
```

파일 타입의 입력 상자 태그를 사용하면 파일을 선택하는 창을 띄우기 위한 별도의 작업이 필요 없으며, 웹 브라우저가 나머지 작업을 알아서 처리해 줍니다.

로드된 파일의 정보 얻기

이러한 폼 데이터와 파일 정보는 action에 지정된 주소로 전송되는데 그럼 어떻게 업로드된 파일 정보를 얻을 수 있을까요? 정답은 "친절한 PHP씨"가 정말 정말 친절하게 다음 변수들을 통해 전달해 줍니다.

```
$_FILES 또는 $HTTP_POST_FILES
```

$_FILES는 PHP 버전 4.1.0 이후부터 생긴 슈퍼 글로벌 변수이고 $HTTP_POST_FILES는 그 이전 버전부터 제공되었던 변수입니다. 따라서 PHP 버전에 따라서 적절히 사용하세요.

[예제 13-1]에서 사용한 폼으로 파일을 전송했다면 $_FILES['upfile']이라는 배열이 생깁니다. 여기서 배열의 키값이 upfile인 것은 앞서 폼에서 파일 상자의 이름을 upfile로 정했기 때문입니다. 즉, 파일 상자의 이름이 자동으로 $_FILES 배열의 키값이 됩니다. $_FILES['upfile'] 배열은 다음과 같은 정보를 갖습니다.

변수	내용
$_FILES['upfile']['name']	업로드된 파일명
$_FILES['upfile']['type']	MIME 형식. 예를 들면 이미지인지 동영상 파일인지 (images/gif , images/jpg 등등)
$_FILES['upfile']['size']	업로드된 파일의 크기
$_FILES['upfile']['tmp_name']	업로드된 파일을 서버 측에서 임시로 저장해둔 이름
$_FILES['upfile']['error']	파일 업로드와 관련된 에러 코드

[표 15-1] 파일 업로드 관련 변수

위의 변수를 통해서 업로드된 파일에 대한 정보와 임시로 저장된 서버 내의 파일 이름을 알아낼 수 있습니다. 명탐정 코난이나 소년탐정 김전일을 즐겨보는 눈치 빠른 분은 이미 알아차렸을지 모르겠습니다. 그렇습니다. 폼을 통해서 전달하면 업로드된 파일은 임시로 서버에 저장됩니다. 그래서 임시로 저장된 파일을, 저장해두고 싶은 디렉토리로 옮기는 작업이 반드시 필요합니다.

[예제 13-2] 로드된 파일의 정보 얻기

```
<?
   echo "파일명 : " . $_FILES['upfile']['name'] . "<BR>";
   echo "MIME : " . $_FILES['upfile']['type'] . "<BR>";
   echo "파일 크기 : " . $_FILES['upfile']['size'] . "<BR>";
   echo "임시 파일 : " . $_FILES['upfile']['tmp_name'] . "<BR>";
   echo "에러코드 : " . $_FILES['upfile']['error'] . "<BR>";
?>
```

[그림 13-71] 업로드된 파일 정보

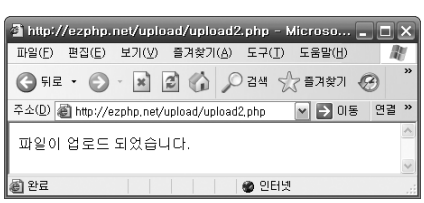
텍스트 파일을 위와 같이 업로드했더니 MIME 형식이 텍스트로 보이는 것을 확인할 수 있습니다. 또한 파일 크기가 정확히 표시되는 것도 확인할 수 있습니다. 윈도우에서는 업로드된 파일이 임시적으로 C:\windows\temp 디렉토리에 저장되며 리눅스에서는 /tmp 디렉토리에 저장됩니다. 만약 업로드할 때 에러가 발생했다면 에러 코드가 0이 아닌 다른 값을 출력할 것입니다.

업로드된 파일의 이동

파일을 업로드하면 /tmp나 c:\windows\temp와 같은 임시 디렉토리에 저장됩니다. 그래서 프로그래머가 원하는 디렉토리로 임시저장 파일을 이동시켜야 하는데 move_uploaded_file() 함수를 이용하면 쉽게 이동시킬 수 있습니다.

[예제 13-3] 임시 업로드 파일 이동하기

```
1  <?
2    if (empty($_FILES['upfile']['name']))
3    {
4      echo "업로드된 파일이 없습니다.";
5      exit;
6    }
7
8    move_uploaded_file($_FILES['upfile']['tmp_name'],'./uploads_dir');
9    echo "파일이 업로드 되었습니다.";
10 ?>
```



[그림 13-72] 파일 업로드 결과

실제 디렉토리로 이동하여 파일이 제대로 업로드되었는지 확인해 보겠습니다.

[그림 13-73] 업로드된 파일 확인

파일 업로드 용량 제한 변경

PHP를 설치하고 PHP.ini 파일의 설정을 변경하지 않았다면 파일 업로드는 2MB를 넘지 못하게 설정되어 있습니다. 이 때문에 2MB 이상의 크기를 가진 파일은 업로드되지 않습니다. 그러므로 대용량 파일을 지원하려면 부득이하게 php.ini 파일을 수정해야 합니다. php.ini 파일은 서버 관리자만이 수정할 수 있습니다. 웹 호스팅과 같이 서버 설정을 할 수 없는 경우라면 서버 관리자에게 요청합니다.

php.ini 파일을 열어보면 upload_max_filesize라는 항목이 있습니다. 2M인 기본값을 원하는 크기로 변경합니다. 보통 20M 정도면 충분합니다. 이 값을 바꾸면 웹 서버를 반드시 재시작해야 합니다. 왜냐하면 php.ini 파일은 웹 서버가 실행될 때 한 번만 로딩되기 때문입니다. 그래서 웹 서버를 재시작해서 새로운 설정 값을 받아들일 수 있게 합니다.

그러나 upload_max_filesize 설정을 변경하고 대용량 파일을 업로드해보면 여전히 파일이 업로드되지 않는 것을 알게 됩니다. 그 이유는 post_max_size 항목의 값 때문입니다. post_max_size 항목은 POST를 이용해서 전달할 수 있는 용량의 최대값을 설정하는 부분입니다. 파일 전송이 POST를 통해서 전달되므로 이 부분의 값도 늘려줘야 합니다. 그래서 다시 php.ini 파일을 열어 post_max_size 값을 upload_max_filesize보다 크게 설정합니다. 파일 용량과 폼 정보에 대한 용량이 합쳐지기 때문에 파일 업로드 용량보다 약간 크게 잡는 것입니다.

파일 업로드 보안

파일 업로드는 가장 널리 사용되는 웹 해킹의 원인 중 하나입니다. 단순히 생각해서 모든 파일을 업로드할 수 있게 한다면 사용자는 php 파일을 업로드한 후 웹 브라우저를 통해 자신이 업로드한 php 파일을 실행할 수 있습니다. 만약 사용자가 파일을 삭제하는 코드나 비밀번호 정보 등을 얻을 수 있는 php 파일을 업로드했다면 너무나 쉽게 해킹의 피해를 입게 될 것입니다. 따라서 파일의 업로드를 허용할 때는 반드시 PHP로 실행될 수 있는 파일에 대한 제약을 걸어야 합니다. 이 부분에

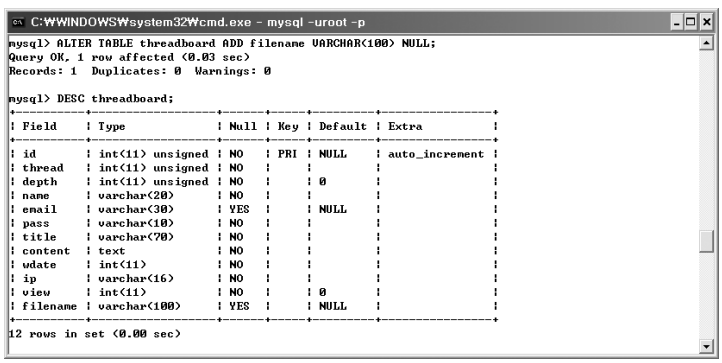
대해서는 15장 웹 해킹에서 자세히 다루겠습니다.

게시판 테이블 수정

이제 본격적으로 게시판에 파일 업로드 기능을 추가하겠습니다. 우선 업로드한 파일의 정보를 저장할 수 있는 필드를 게시판 테이블에 추가해야 합니다.

```
ALTER TABLE threadboard ADD filename VARCHAR(100) NULL;
```

파일의 이름을 저장할, 문자열 100자를 저장할 수 있는 filename이라는 필드를 추가했습니다. 파일은 업로드될 수도 있고 그렇지 않을 수도 있기 때문에 NULL 값을 가질 수 있도록 설정해 줍니다.



[그림 13-74] 파일 업로드용 필드 추가

게시판 폼 수정

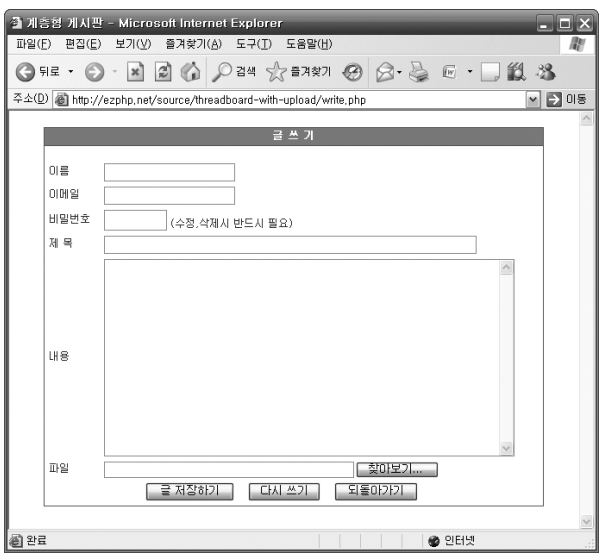
게시판에서 파일 업로드가 가능한 곳은 글 쓰기, 글 수정 그리고 답글 달기입니다. 세 부분은 거의 같은 기능을 가지며 글 수정하기의 경우 업로드한 파일에 다시 업로드하는 기능이 추가되어야 하므로 이미 업로드되어 있는 파일을 보여주는 부분을 추가합니다.

write.php, edit.php, reply.php 파일의 〈form〉 태그 부분에서 파일 업로드가 가능하도록 enctype="multipart/form-data" 부분을 추가하고 업로드를 위한 파일 상자를 적당한 위치에 추가합니다.

```
<form action=insert.php method=post enctype="multipart/form-data">
... (생략) ...
<tr>
  <td width=60 align=left >내용</td>
  <td align=left >
    <TEXTAREA name=comment cols=65 rows=15></TEXTAREA>
  </td>
```

```
</tr>
<tr>
  <td width=60 align=left >파일</td>
  <td align=left >
    <input type=file size=40 name=upfile>
  </td>
</tr>
... (생략) ...
```

업로드 파일 상자가 추가된 폼의 형태는 다음과 같습니다.



[그림 13-75] 추가된 업로드 파일 상자

업로드 파일 처리

폼을 통해서 파일을 전달할 준비가 모두 끝났습니다. 이제는 폼을 통해서 넘어온 파일을 잘 받아서 PHP로 그 파일을 업로드 디렉토리에 저장하고 데이터베이스에 파일의 정보를 기록하는 작업을 하겠습니다.

[예제 13-3]을 약간 수정하여 폼으로부터 전달된 파일이 있으면 이를 저장하도록 합니다.

```
<?
$filename =""; //변수 초기화
if (!empty($_FILES['upfile']['name']))
{
            if(move_uploaded_file($_FILES['upfile']['tmp_name'],'./
uploads_dir'))
            {
                        $filename = $_FILES['upfile']['name'];
```

```
            }
            else {
                        ErrorMessage('업로드에 실패하였습니다.');
            }

}

?>
```

게시판 디렉토리에 uploads_dir 디렉토리를 생성하고 위의 코드를 insert.php, update.php, insert_reply.php 파일의 상단부에 추가합니다. 이제 업로드된 파일의 정보를 데이터베이스에 기록하기 위해서 쿼리를 수정해야 합니다. insert.php의 쿼리는 다음과 같이 수정하면 됩니다.

```
insert into $board
(thread, depth, name, pass, email, title, view, wdate, ip, content,
filename)
values
($max_thread,0,'$_POST[name]','$_POST[pass]','$_POST[email]',
'$_POST[title]',
0,UNIX_TIMESTAMP(),'$REMOTE_ADDR','$_POST[content]', 0,'$filename')
```

이와 마찬가지로 insert_reply.php 파일도 쿼리에서 filename 부분만을 추가로 넣어 주면 제대로 동작하는 것을 확인할 수 있습니다. 그러나 이 두 파일과는 달리 update.php 파일에서는 위와 같은 방법으로 추가하면 문제점이 있습니다. 이미 업로드한 파일이 있을 경우 단순히 본문의 내용만 수정하면 새로 업로드되는 파일이 없으므로 $filename 변수의 빈 값이 기존의 업로드 파일 정보를 덮어쓰는 것입니다. 따라서 새로운 업로드 파일이 있는 경우에만 데이터베이스에 기록하도록 쿼리를 변경해야 합니다.

```
<?
  $add_query=""; //변수 초기화
  if (!empty($_FILES['upfile']['name']))
  {
    if(move_uploaded_file($_FILES['upfile']['tmp_name'],
    './uploads_dir'))
    {
      $filename = $_FILES['upfile']['name'];
      $add_query = ", filename ='$filename' ";
    }
    else {
        ErrorMessage('업로드에 실패하였습니다.');
    }

  }
```

```
    $query = "update $board set name='$_POST[name]',
    title='$_POST[title]' ";
    $query .=",email='$_POST[email]', content='$_POST[content]'";
    $query .= $add_query;//새로운 업로드 파일이 있는 경우 쿼리가 추가됨
    $query .= "where id='$_GET[id]' ";
    $result=mysql_query($query, $conn);
?>
```

이렇게 수정하면 업로드가 완성되며 read.php와 같은 곳에 적절히 파일을 다운받을 수 있도록 데이터베이스에서 정보를 가져와 뿌려주면 모든 것이 끝납니다. 업로드된 파일이 게시판 디렉토리의 uploads_dir 디렉토리에 저장되므로 "http://도메인/threadboard/uploads_dir/파일이름"과 같은 방법으로 접근할 수 있습니다. 그러나 PHP 파일을 업로드한다면 사용자가 웹 서버에서 마음껏 PHP를 실행할 수 있기 때문에 매우 위험합니다. 따라서 반드시 업로드 가능한 파일을 제한해야 합니다.

업로드 가능한 파일의 제한

앞서 말씀드렸듯이 업로드는 매우 조심해서 구현해야 합니다. PHP 파일이나 CGI 파일과 같이 웹 서버에서 실행할 수 있는 파일이 업로드되지 않도록 반드시 제한 장치를 만들어야 합니다. PHP나 CGI 확장자를 가지는 파일을 업로드하지 못하도록 파일 이름 확인 부분을 추가합니다.

```
<?
    $add_query=""; //변수 초기화
    if (!empty($_FILES['upfile']['name']))
    {
      //금지할 확장자
      $ban_ext = array('php','php3','html','htm','cgi','pl');

      //확장자를 이용하여 업로드 가능한 파일인지 체크한다.
      $fname = explode(".",$_FILES['upfile']['name']);
      //배열의 마지막 원소, 즉 확장자를 소문자로 가져온다.
      $ext = strtolower($fname[sizeof($fname)-1]);
      if (in_array($ext,$ban_ext)) {
        ErrorMessage ("업로드가 불가능한 확장자입니다.");
      }

    if(move_uploaded_file($_FILES['upfile']['tmp_name'],'./uploads_dir'))
    {
    $filename = $_FILES['upfile']['name'];
    $add_query = ", filename ='$filename' ";
    }
    else {
    ErrorMessage('업로드에 실패하였습니다.');
```

```
    }
  }
?>
```

이 코드처럼 업로드 부분을 수정하면 업로드 파일의 확장자를 분석하여 PHP나 CGI가 실행 가능한 확장자를 가진 파일이 업로드되는 것을 차단할 수 있습니다. 그러나 15장 웹 해킹에서 다시 다루겠지만 이 방법은 맹점이 있으므로 15장에서 사용하는 정규식을 이용한 확장자 검증 방법을 사용하기 바랍니다.

만약 PHP 파일이나 HTML 등의 파일을 업로드하고 웹 서버에서 실행하는 것이 아니라 사용자에게 파일을 다운로드받게 하고 싶다면 파일의 확장자 없이 파일 이름을 변경하여 저장하는 방법을 사용할 수 있습니다. 예를 들어 abc.php 파일을 업로드할 때 abc_php와 같이 이름을 변경하여 저장하고 다운로드할 때 이를 다시 abc.php로 변환하여 줄 수 있습니다. 이 방법을 사용하면 동일한 이름의 파일이 업로드될 경우 문제가 발생하는데 이때도 파일의 이름 앞에 업로드된 날짜와 시간(초 단위까지) 정보를 추가하거나 md5() 함수를 이용하여 해시 문자열을 앞에 추가하면 이 문제를 해결할 수 있습니다.

Chapter

14

디버깅

PHP 프로그래밍을 하다 보면 항상 프로그래머가 원하는 형태의 수행결과를 나타내는 것이 아닙니다. 컴퓨터가 명령을 잘못 알아듣거나 컴퓨터에 지시하는 명령에 문제가 있을 때 이런 일이 발생합니다. 우리가 말을 하다 보면 듣는 사람이 말을 잘못 알아듣거나 말하는 사람이 잘못 말하는 경우와 같습니다. 이때 어법에 맞게 말했는지, 정확한 단어를 사용했는지 등을 확인하듯 프로그래밍에서도 문법에 맞는지 또는 그 의도가 정확하게 표현되었는지를 확인해야 합니다. 이러한 절차를 디버깅이라고 하며 디버깅을 통해서 프로그램을 올바르게 수정하는 방법을 이 장에서 살펴보겠습니다.

Section 01

버그, 디버그, 에러

버그

버그는 프로그램의 오류를 뜻하는 말입니다. 더 정확히 말하면 오류를 일으키는 부분을 뜻합니다. 버그는 우리말로 벌레란 뜻입니다. 프로그램 오류와 벌레는 무슨 상관이 있을까요? 언뜻 보기엔 왜 프로그램 오류를 벌레라는 단어로 말하는지 이해가 되질 않습니다. 하지만 여기엔 다 사연이 있습니다. 뭐 버그의 역사를 보면 진공관 시절로 돌아가야 하지만 가장 널리 사용되기 시작한 계기는 1944년으로 생각됩니다. 하버드 대학에서 컴퓨터 프로그램을 개발하던 그레이스 호퍼가 컴퓨터 작업 중에 잘 동작하던 프로그램이 제대로 동작하지 않자 그 원인을 찾던 중 컴퓨터에 들어가 있던 벌레(나방) 한 마리를 발견한 일화를 통해 프로그램의 오류가 버그라는 말로 널리 알려지게 되었습니다. 이 나방은 해군에 여러 해 동안 전시되고 스미스쏘니언 박물관에 소장되어 있다고 전해집니다.

디버그

우리는 중, 고등학교 때 영어 시간에 접두사에 대해서 배웠습니다. 능률 보케블러리라는 책이 매우 인기가 있었는데 어휘력을 기르기 위해서 접두사와 접미사 등을 이용해서 모르는 단어도 접두사나 접미사를 통해 뜻을 짐작할 수 있다는 발상으로 만든 책이었습니다. 옛 기억을 되살려보면 de라는 접두사는 un 등과 함께 원래 단어의 뜻 부정이나 반대말을 만드는데 자주 사용이 됩니다. 디버그(debug)는 바로 벌레를 잡는다는 뜻입니다. 앞서 벌레가 프로그램의 오류를 뜻한다고 하였으니 디버그는 프로그램의 오류를 잡는다는 말이 됩니다. 즉, 프로그램의 오류를 찾아서 수정하는 것을 디버그라고 합니다. 디버깅(debugging)이라는 말도 자주 사용하는데 디버그가 오류 수정 프로그램과 그 작업을 통칭하는 반면 디버깅은 오류 수정을 하는 작업 자체를 의미하는 말로 사용됩니다. 오류 수정 프로그램은 디버거(debugger)라고 합니다.

에러

에러는 프로그램이 사용자가 기대하는 바와 다르게 다른 결과를 나타내는 것을 말합니다. 즉, 앞서 말한 버그와 같은 것으로 인해서 발생하거나 사용자의 실수나 오작동으로 일어나는 현상입니다.

에러에는 여러 가지 종류가 있는데 다음과 같이 4종류로 분류할 수 있습니다.

① 문법 에러 (Syntax Error)

② 실행 시간 에러 (Runtime Error)

③ 논리 에러 (Logic Error)

④ 컴퓨터 환경에 따른 에러 (System Error)

뭔가 복잡해 보이는데 하나씩 짚어가 보겠습니다.

문법 에러

문법 에러는 말 그대로 문법을 잘못 이해하거나 잘못 사용해서 생기는 에러를 말합니다. PHP 문법에서 세미콜론(;)의 중요성을 아시죠? 모든 문장은 세미콜론으로 끝나는데 프로그래머는 자주 세미콜론을 붙이지 않는 실수를 저지르곤 합니다. 이 경우에 문법적으로 틀린 문장을 사용했기 때문에 문법 오류가 발생합니다.

이 문법 에러는 문법을 제대로 숙지하고 있다면 매우 쉽게 발견하고 수정할 수 있습니다. 특히 컴파일러나 인터프리터가 문법 에러를 찾아서 에러 메시지로 알려주기 때문에 처리하기 매우 수월한 에러입니다. (특히 PHP와 같은 인터프리터 언어에서는 문법 에러를 파스 에러(parse error)라고 합니다.)

실행 시간 에러

실행 시간 에러는 프로그램을 실행하는 도중에 생기는 에러를 뜻합니다. 문법 에러는 소스 자체에 문제가 있기 때문에 실행 자체가 불가능한데 실행 시간 에러는 문법적으로 전혀 이상이 없어서 프로그램이 실행되었으나 사용자의 입력에 대한 처리 미숙과 존재하지 않는 파일을 인클루드하는 등의 문제로 인해서 프로그램이 올바르게 실행되지 않는 에러입니다. 대부분의 경우 경고나 에러 메시지를 출력하기 때문에 비교적 찾기 쉬운 에러입니다.

논리 에러

논리 에러는 논리적으로 발생하는 에러를 말합니다. 문법상으로 틀린 것이 없지만 더해야 할 부분을 빼다든가 곱해야 할 부분을 나눈다든가 하는 등의 프로그래머의 부주의나 잘못된 이해에서 생기는 에러입니다. 논리 에러는 소스 코드의 실행에 문제가 없고 단지 원치 않는 결과를 도출하기 때문에 그 원인을 찾기가 매우 힘이 듭니다. 이 에러는 주로 수식이나 변수의 처리 결과를 순서대로 확인하는 방법을 통해서 에러 부분을 찾습니다.

컴퓨터 환경에 따른 에러

시스템 에러는 컴퓨터 환경에 따라서 발생하는 에러를 말합니다. PHP는 거의 대부분의 플랫폼에서

사용할 수 있지만 그 기반이 리눅스 계열의 OS에서 시작되었기 때문에 윈도우용 PHP에서는 지원하지 않는 함수들이 더러 있습니다. 리눅스 계열의 OS에서만 사용할 수 있는 함수를 윈도우용에서 사용하는 경우 시스템 에러가 발생할 수 있습니다. 그러나 이 에러는 에러 메시지를 통해서 쉽게 알 수 있기 때문에 어려움 없이 수정할 수 있습니다. 그러나 해당 함수를 사용할 수 없으므로 이를 대체할 수 있는 함수를 찾거나 해당 함수를 직접 구현하는 노력이 필요합니다.

프로그램을 만들고 사용하다 보면 의도한 대로 동작하지 않을 때가 많습니다. 이럴 때 프로그램에 버그가 있다거나 에러가 발생했다고 합니다. 에러가 생긴 원인 중에는 사용자의 실수도 있을 것이고 프로그램 자체의 오류도 있을 것입니다. 이때 프로그램 자체의 오류를 버그라고 합니다.

Section 02

대표적인 에러

실제로 각 에러에 대해서 가장 빈번히 발생하는 예를 들어보겠습니다.

문법 에러

문법 에러는 문법을 잘못 이해하고 있거나 실수로 잘못 사용함으로 인해서 일어나는 에러입니다. 주로 다음 세 녀석 때문에 발생합니다. 매우 위험한 녀석들이니 공개 수배합니다.

[그림 14-1] 문법 에러의 주범 3인조

세미콜론을 잊은 경우

세미콜론의 현상금이 제일 많듯이 가장 많은 문법 에러가 바로 이 세미콜론으로 인해 발생합니다. PHP에서 한 문장을 끝내려면 세미콜론이 필요합니다. PHP를 시작한 지 얼마 되지 않는 입문자에게서 가장 빈번하게 발생하는 문법 에러이며 아무리 많은 경험을 가진 개발자라고 하더라도 꼭 한

번씩 하는 실수가 바로 세미콜론을 잊어서 발생하는 에러입니다.

[예제 14-1] 세미콜론을 잊어서 발생하는 에

```
1  <?
2     echo "이 에러가 문법 에러의 주범" // <-- 여기 세미콜론이 빠졌어요.
3     echo "세미콜론을 잊지 말아요~~";
4  ?>
```

[예제 14-1]은 두 번째 줄에서 세미콜론을 잊어버리고 기입하지 않았습니다. 모든 문장에 세미콜론으로 끝이 나야 하는데 세미콜론이 빠졌으니 당연히 문법 에러가 발생합니다.

[그림 14-2] 세미콜론으로 인한 문법 에러

앞서서 PHP와 같은 인터프리터 언어에서는 문법 에러를 파스 에러라고 한다고 하였습니다. 그래서 위와 같이 Parse Error가 발생합니다.

에러 메시지를 살펴보면 해당 파일의 세 번째 줄에서 콤마(,)나 세미콜론(;)을 기대하고 있는데 기대하지 않는 echo가 발생했다고 합니다. 정말 매우 똑똑하군요. 있어야 할 곳에 세미콜론이 없으므로 PHP는 위의 소스 코드를 다음과 같이 인식합니다.

echo "이 에러가 문법 에러의 주범" echo "세미콜론을 잊지 말아요~~";

위의 문장을 한 문장으로 생각하기 때문에 따옴표 다음에 나타난 echo를 이해할 수 없는 것입니다. 올바르게 하려면 문장을 끝내거나 echo 대신에 문자열의 합 연산자인 점(.)이나 콤마(,)가 나와야 올바른 문법이 됩니다.

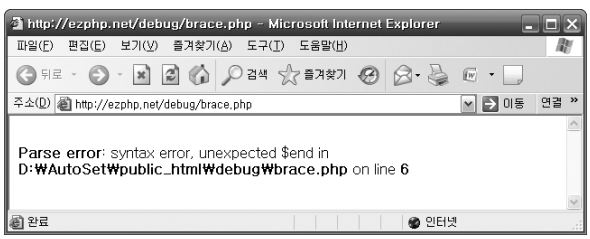
중괄호를 안 닫은 경우

중괄호는 함수나 제어 구문에서 매우 많이 사용하는데 쌍을 맞추지 못해서 자주 발생하는 에러입니다. 100이면 99는 중괄호를 열어놓고 닫지 않아서 발생합니다.

[예제 14-2] 중괄호로 인한 문법 에러

```
1  <?
2     echo "이 에러가 문법 에러의 또 다른 주범";
3     if ($name=="브라운") {
4        echo "내 별명은 " . $name;
5        // <-- 여기에 } 요놈이 있어야 하는데 없군요.
6  ?>
```

[예제 14-2]의 세 번째 줄에 IF 구문으로 중괄호가 열렸습니다. 그러나 다섯 번째 줄에서 마땅히 짝을 이뤄야 할 닫는 중괄호가 없어서 이 소스는 에러가 발생합니다.



[그림 14-3] 중괄호로 인한 문법 에러

문법 에러이기 때문에 Parse Error가 발생하고 6번째 줄에서 뭔가 더 있어야 하는데 갑자기 PHP 문장이 종료되었다고 말하고 있습니다.

PHP에서는 중괄호가 열리면 항상 자신의 짝이 되는 닫는 중괄호를 찾습니다. 그런데 [예제 14-2]에서는 열린 중괄호는 있으나 닫힌 중괄호가 없어서 문법 에러를 발생시킨 것입니다. 위의 소스와 같이 중괄호가 하나만 쓰인 경우에는 쉽게 찾을 수 있으나 중괄호가 이중 삼중으로 사용된 경우에는 그 쌍을 찾기가 힘듭니다. 그래서 그 쌍을 쉽게 찾을 수 있도록 여는 중괄호와 닫는 중괄호를 동일 선상에 들여쓰기하는 방법을 많이 사용합니다. 때로는 중괄호에 주석을 추가하여 어느 것과 쌍을 이루는지 문구를 남겨두기도 합니다.

중괄호로 인한 문법 에러는 매우 흔하게 발생하기 때문에 괄호의 쌍을 보여주어 실수를 사전에 막을 수 있도록 도와주는 에디터도 있습니다.

따옴표를 안 닫은 경우

중괄호와 마찬가지로 쌍을 이루어야 하는데 쌍을 이루지 못해서 발생하는 문법 에러입니다.

[예제 14-3] 따옴표로 인한 문법 에러

```
1  <?
2     echo "이 에러가 문법 에러의 또 다른 주범;
3     $sum = $a + $b;
4  ?>
```

[예제 14-3]의 소스에서는 두 번째 줄에서 문자열을 감싸는 따옴표가 하나 빠졌습니다. 따옴표가 쌍을 이루지 못하기 때문에 문법 에러가 발생합니다.

[그림 14-4] 따옴표로 인한 문법 에러

따옴표는 큰따옴표든 작은따옴표든 항상 쌍을 이루어야 합니다. 문자열을 감싸고자 사용하는 따옴표에서 닫는 따옴표를 사용하지 않았기 때문에 위의 소스 코드는 다음과 같이 인식됩니다.

```
echo "이 에러가 문법 에러의 또 다른 주범; $sum = $a + $b;
```

합을 구하는 산술 연산 부분까지 모두 문자열의 일부라고 생각합니다. 에러의 위치가 2번째가 아니라 4번째 줄인 이유가 바로 여기에 있습니다. PHP는 문자열을 감싸는 닫는 따옴표를 찾아서 산술 연산 부분까지 문자열이라고 생각하기 때문에 3번째 줄까지는 아무런 이상이 없습니다. 그런데 쌍을 이뤄야 할 닫는 따옴표가 나오기 전에 PHP 부분이 끝이 나버려서 예기치 않은 종료 에러가 발생한 것입니다.

따옴표로 인한 에러는 에디터의 기능을 이용하면 매우 쉽게 알아낼 수 있습니다. 바로 문법 강조(Syntax Highlight) 기능입니다. 문법 강조 기능은 코드를 구분하기 쉽게 색깔 변화를 주는 것을 말합니다. 메모장에서는 모든 코드가 검은색 글씨로만 보이지만 문법 강조를 지원하는 에디터에서는 변수나 키워드 그리고 문자열과 같은 것들을 서로 구분해서 다른 색깔로 보여줍니다. 이 색깔을 이용하면 쉽게 따옴표 에러를 잡을 수 있습니다.

잘못된 코드	<?
	  echo "이 에러가 문법 에러의 또 다른 주범;
	  $sum = $a + $b;
	?>

올바른 코드	<?
	  echo "이 에러가 문법 에러의 또 다른 주범";
	  $sum = $a + $b;
	?>

[표 14-1] 문법 강조를 통한 에러 찾기

이 책이 컬러가 아니어서 색상이 드러나지 않기에 볼드체로 표기했습니다. 잘못된 코드에서는 echo 이후가 모두 문자열로 취급되어 볼드체로 표기되었습니다. 반면에 올바른 코드에서는 문자열이 제대로 끝났으므로 나머지 부분이 일반 폰트로 표현된 것을 알 수 있습니다. 이처럼 일부 에디터는 문법 강조 기능으로 색상이 이상한 부분을 감지해서 해당 부분의 에러를 쉽게 찾아냅니다.

기타

위의 세 가지 경우 이외에도 몇 가지 흔한 에러가 더 있습니다.

[예제 14-4] 주석 처리로 인한 에러

```
1  <?
2     /* 주석 처리를 해놓고 닫지 않으면 전부 주석 처리됩니다!!
3     $sum = $a + $b;
4  ?>
```

[예제 14-4]와 같이 여러 줄에 대한 주석을 처리하고자 할 때 닫지 않는 것으로 발생하는 문법 에러가 있습니다.

[그림 14-5] 주석으로 인한 문법 에러

이 경우에는 그 이후 모든 부분이 주석 처리되기 때문에 경고 메시지가 발생합니다. 경고 메시지에는 주석이 시작되었으나 주석 처리가 끝나지 않았다고 말하고 있습니다.

[예제 14-5] PHP 시작 태그로 인한 에러

```
1  <?
2     $sum = $a + $b;
3  <? // 닫지 않고 PHP 시작 태그를 사용했을 때
4
5  ?>
```

PHP 태그도 시작과 끝이 짝을 이루어야 합니다. [예제 14-5]와 같이 PHP 시작 태그가 짝을 이루지 않고 열린 상태에서 다시 열리면 문법 에러가 발생합니다.

[그림 14-6] PHP 시작 태그로 인한 문법 에러

실행 시간 에러

실행 시간 에러는 주로 다음과 같은 경우에서 발생합니다.

① 존재하지 않는 파일을 사용할 때
② 존재하지 않는 함수를 호출할 때
③ 0으로 나누었을 때
④ 사용자의 입력을 제대로 처리하지 못했을 때
⑤ 데이터베이스를 올바르게 사용하지 못할 때

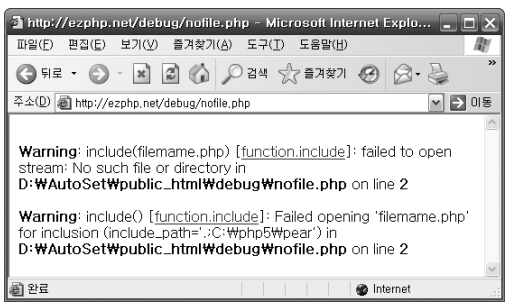
존재하지 않는 파일을 사용할 때

파일을 다루다 보면 오타나 실수로 존재하지 않는 파일을 사용하는 경우가 있습니다. 주로 파일을 읽고 쓰거나 인클루드하는 경우에 발생합니다.

[예제 14-6] 존재하지 않는 파일로 인한 에러

```
1  <?
2     include "filemame.php";
3  ?>
```

filename.php라고 입력하려고 했으나 오타로 인해서 filemame.php를 인클루드했습니다. 그런데 해당 파일이 시스템에 존재하지 않기 때문에 다음과 같이 경고 메시지가 출력됩니다.



[그림 14-7] 파일이 존재하지 않을 때의 에러

첫 번째 경고는 해당 이름을 가진 파일이나 디렉토리를 찾을 수 없다는 것이고 두 번째 경고는 PHP 설정 파일인 php.ini 파일에 지정된 include_path에서도 해당 파일을 찾을 수 없다는 메시지입니다.

존재하지 않는 함수를 호출할 때

파일에서와 마찬가지로 오타나 실수로 존재하지 않는 함수를 호출하거나 등록되지 않은 라이브러리의 함수를 사용하고자 할 때 발생하는 에러입니다.

[예제 14-7] 존재하지 않는 함수로 인한 에러

```
1  <?
2     function func1()
3     {
4     echo "func1";
5     }
6
7     //존재하지 않는 함수를 호출
8     func2();
9  ?>
```

[예제 14-7]에서 func2() 함수는 존재하지 않는 함수이기 때문에 호출하면 에러가 발생합니다.

[그림 14-8] 존재하지 않는 함수 호출로 인한 에러

문법 에러가 아니므로 Fatal Error가 발생합니다. 에러 메시지는 func2()라는 함수가 알려진 함수가 아니라고 말하고 있습니다.

0으로 나누었을 때

수를 0으로 나누게 되면 그 수가 얼마가 되었든지 간에 양의 무한대나 음의 무한대가 됩니다. 그래서 PHP에서는 수를 0으로 나누지 못하도록 하고 있습니다.

[예제 14-8] 0으로 나누었을 때

```
1  <?
2     $a = 5;
3     $b = $a - 5;
4     $c = $a / $b; //0으로 나눈다.
5  ?>
```

[예제 14-8]의 $b는 5에서 5를 빼게 되어 결국 0이 됩니다. 그런데 네 번째 줄에서 $b 값으로 나누기 때문에 결국 0으로 나누어지게 됩니다. 그래서 다음과 같은 경고가 발생합니다.

[그림 14-9] 0으로 나누었을 때의 경고

4번째 줄에서 0으로 나누고 있다는 경고 메시지입니다.

사용자의 입력을 제대로 처리하지 못했을 때

폼을 통해서 사용자로부터 값을 전달받는 경우 사용자의 실수로 원치 않는 값을 전달받을 때가 있습니다. 사용자가 실수할 수 있는 경우를 예측하여 초기화 등의 조치를 해야 하지만 이런 예외 처리에 둔감하여 사용자의 입력을 100% 신뢰하는 경우 실행 시간 에러가 발생할 수 있습니다. 예를 들어 두 수를 폼으로 입력받아 나눈 다음 그 결과를 출력해주는 프로그램을 개발하였는데 사용자가 나누는 수 부분에 0을 입력하거나 숫자를 입력하지 않는 경우 앞서 배운 0으로 나누는 에러가 발생할 수 있습니다.

이 에러는 사용자가 입력하는 부분에 대한 제약을 두거나 사용자가 입력하지 않은 부분에 대해서는 초기화를 해주는 등의 절차를 통해서 사전에 방지할 수 있습니다.

데이터베이스를 올바르게 사용하지 못할 때

데이터베이스가 시작되어 있지 않거나 쿼리를 잘못 작성하였을 때 쉽게 발생하는 에러입니다.

[예제 14-9] 데이터베이스의 이상으로 인한 에러

```
1  <?
2    mysql_connect("localhost","brown","1234");
3  ?>
```

소스 코드에는 전혀 이상이 없는데 MySQL 데이터베이스가 종료되어 있어서 에러가 발생할 수 있습니다.

[그림 14-10] 데이터베이스의 문제로 인한 에러

PHP에서 MySQL로 연결을 시도했지만 MySQL Server가 응답하지 않아서 연결할 수 없다고 말하고 있습니다. 이와 유사하게 데이터베이스는 제대로 동작하고 있으나 사용자 아이디와 비밀번호가 잘못되었거나 올바르지 못한 쿼리를 던지는 것등으로 이와 유사한 에러가 발생할 수 있습니다.

주로 사용자의 입력에 대한 값을 통해 쿼리를 작성하는 경우 사용자의 입력을 제대로 확인하지 않

아서 쿼리에 문제가 발생하는 일이 많습니다.

논리 에러

앞서 배운 두 종류의 에러는 그래도 비교적 쉽게 고칠 수 있습니다. 그러나 논리 에러는 어느 부분이 틀렸는지 찾기가 힘들어서 며칠 동안이나 디버깅을 하는 경우가 많습니다. 그 이유는 문법 에러나 실행 시간 에러처럼 에러 메시지가 나타나는 것이 아니기 때문입니다.

문법적 오류나 실행 시간에 찾아낼 수 있는 에러가 아니라 모든 것이 올바르지만 단지 논리적으로 잘못된 것이기 때문에 컴퓨터는 절대 이것을 찾아낼 수 없습니다. 더해야 할 부분에서 빼기를 해 버린다든가 곱해야 할 부분을 나누게 되는 경우 컴퓨터는 프로그래머의 실수를 전혀 알 길이 없습니다. 그래서 컴퓨터는 문법상 에러가 없으니까 맞다고 생각하고 넘어갑니다. 왜냐하면 더해야 할지 빼야 할지는 프로그래머 자신만 알고 있기 때문입니다.

논리 에러는 주로 프로그래머의 부주의나 프로그램에 대한 잘못된 이해로 발생합니다. 가장 일반적인 두 가지 경우에 대해서 알아보겠습니다.

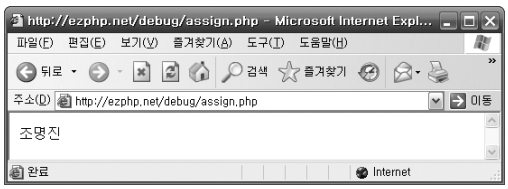
대입 연산자와 비교 연산자의 혼동

우리가 상식적으로 알고 있는 '같다(=)'와 PHP에서 '같다(==)'는 기호가 다르다는 것을 우리는 이미 알고 있습니다. 그러나 매우 능숙한 프로그래머도 종종 이를 별생각 없이 실수할 때가 많습니다. 이는 혼동을 해서라기보다는 우리가 수학에서 너무나 당연하게 받아들이고 있기 때문에 발생하는 것입니다.

[예제 14-10] 대입 연산자와 비교 연산자의 혼동

```
1  <?
2    $name = "조맹진";
3    if ($name="조명진") echo $name;
4    else echo "내 이름은 $name이 아니라 조명진이야!!! 버럭!";
5  ?>
```

[예제 14-10]에서 3번째 줄에는 사용자가 입력한 $name 변수의 값이 "조명진"과 같은지 확인하려고 하였으나 실수로 대입 연산자를 사용하여 사용자가 어떤 값을 입력하든지 상관없이 $name 변수에는 조명진이란 값이 저장되고 IF 구문은 항상 참이 됩니다. 그래서 사용자가 조맹진으로 입력하더라도 항상 올바르게 **조명진**을 출력합니다.



[그림 14-11] 대입 연산자의 잘못된 사용으로 인한 오류

분명히 $name 변수에 값을 "조맹진" (고향에서는 저를 이렇게 부릅니다. "명" 발음이 잘 안되나 봅니다.)이라고 입력하였는데 **조명진**이라는 결과를 얻었습니다. 실제로는 "내 이름은 조맹진이 아니라 조명진이야!!! 버럭!"이라고 출력되어야 하는 데 말입니다.

앞서 배운 두 종류의 에러와 달리 전혀 에러가 발생하지 않기 때문에 그 원인을 찾기가 힘듭니다. 소스 코드를 다시 차근차근 살펴보는 수밖에 없죠. 대입 연산자와 비교 연산자의 혼동은 정말 자주 발생하는 실행 시간 에러입니다. 그래서 항상 주의해서 사용할 필요가 있습니다.

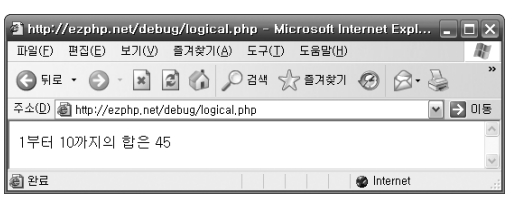
부주의로 인한 에러

프로그래머가 의도하지 않게 부주의로 인해서 논리 에러를 발생시키는 경우가 많습니다.

[예제 14-11] 프로그래머의 부주의로 인한 에러

```
1  <?
2    for ($i=1;$i<10;$i++) $sum = $sum + $i; //9까지만 더한다.
3    echo "1부터 10까지의 합은 " . $sum;
4  ?>
```

프로그래머는 1에서 10까지의 값을 더하고자 하였지만 부주의해서 9까지만 더하게 되었습니다.



[그림 14-12] 논리적인 실수로 인한 오류

이 경우에는 소스가 간단하기에 쉽게 원인을 찾을 수 있지만 수식이 매우 복잡한 과정을 통해서 도출되었다면 어느 부분에서 잘못 계산이 되었는지 찾기가 힘들어집니다. 심지어는 그 결과를 신뢰해서 엉뚱한 결과를 올바른 결과로 오해할 수도 있습니다.

이러한 논리 에러를 찾고자 변수 값을 추적하는 방법이 가장 일반적입니다. PHP에는 무료로 제공하는 멋진 디버거가 없기 때문에 일일이 echo를 통해서 변수의 변화 과정을 확인해보는 수밖에 없습니다. 그래서 다른 어떤 에러보다 문제를 찾기 힘듭니다.

Section 03 에러에 대체하는 자세

많은 사람이 에러가 영문이라서 에러 메시지를 읽어보지도 않고 질문하는 경우가 많습니다. 논리 에러를 제외한 대부분의 에러는 에러 메시지에서 그 해답을 쉽게 찾을 수 있는데도 말입니다. ezphp.net 홈페이지나 phpschool.com과 같은 개발자 커뮤니티에는 에러 메시지를 올려놓고 해결책을 알려달라거나 심지어는 소스 코드를 그대로 올려놓고 수정해 달라고 하는 많은 초보자가 있습니다. 실제로 에러 메시지를 검색엔진으로 확인해보면 거의 대부분 그에 대한 해답이 올라와 있을 정도입니다. 보다 수준 높은 프로그래머가 되려면 에러 메시지를 이해하고 왜 그러한 에러 메시지가 발생하게 된 것인지를 생각해보는 자세가 필요합니다.

Chapter

15

웹 해킹

근래에 발생한 해킹 사례를 보면 웹 애플리케이션의 취약점을 이용한 해킹이 주를 이루고 있습니다. 특히 많은 사이트에서 이용하고 있는 제로보드와 같은 공개 게시판이 대표적인 공격 타겟이 되고 있습니다. 제로보드의 경우는 지속적으로 보안 취약점을 공지하고 보안이 강화된 버전의 패치를 제공하고 있기 때문에 지속적인 관심을 보인다면 심각한 웹 해킹 사고로부터 사이트를 보호할 수 있습니다. 그러나 공개 게시판을 사용하는 대부분의 웹 사이트에서 제때 버그 패치를 해주지 않아서 웹 해킹 사고가 실제로 빈번히 일어나고 있습니다.

많은 프로그래머가 짧은 개발 일정 탓에 기능 구현에 중점을 두다 보니 보안에 소홀한 경우가 많습니다. 웹 사이트 개발 완료 후에도 사이트 오픈 일정이 임박한 탓에 충분히 웹 사이트를 테스트하지 못하고 보안 취약점을 고스란히 남겨둔 채로 웹 사이트를 오픈하는 경우가 허다합니다. 이는 비단 개인 사이트나 중소기업 사이트에만 해당하는 말이 아닙니다. 정보 보호 전문 기업인 NSHC(www.nshc.net)에 따르면 공공기관과 그룹 계열사를 비롯한 100여 개의 웹 사이트를 모의 해킹했을 때 90% 이상의 사이트에서 보안 취약점을 안고 있는 것으로 드러났습니다.

국제 웹 보안 표준 기구인 OWASP(Open Web Application Security Project)는 해마다 자주 발생하는 웹 보안 취약점 "Top 10"을 선정하여 발표하고 있습니다. PHP에 국한된 것이 아니라 웹 애플리케이션에서 발생할 수 있는 전반적인 보안 취약점을 다루고 있습니다. 이러한 사이트들의 도움으로 최근 웹 애플리케이션 보안에 대한 관심이 점차 높아지고 있습니다. 하지만 대부분의 웹 개발자는 보안보다는 기능 구현에 중점을 두고 있기 때문에 여전히 많은 문제를 가지고 있습니다. 그래서 이 단원을 통해 PHP에서 발생할 수 있는 대표적인 웹 보안 취약점 공격의 유형과 그 해결책에 대해서 알아보겠습니다.

이 웹 해킹 단원은 양날의 검과 같습니다. 좋은 의도로 사용하는 경우 웹 해킹의 공격으로부터 보호받을 수 있고 나쁜 의도로 사용하는 경우 여러분이 크래커가 될 수 있습니다. 부디 좋은 의도로만 사용하길 부탁드립니다.

Section 01

PHP 웹 보안 취약점 Top 5

웹 서버에 방화벽을 완벽히 설치하고 성능 좋은 바이러스 백신까지 최신 버전으로 업데이트를 유지하고 있으니 웹 해킹에 대한 걱정은 할 필요가 없다고 생각하는 사람들이 있습니다. 방화벽과 바이러스 백신은 시스템에 대한 네트워크 공격과 바이러스의 확산을 막는 것일 뿐입니다. 웹 해킹은 방화벽이 안전하다고 믿는 HTTP 포트를 통해서 이루어지고 그 형태 또한 HTTP 규약에 맞게 전송되기 때문에 방화벽으로 웹 해킹을 차단할 수 없습니다.

시스템을 일반 가정집이라고 가정해 봅시다.

> 아이들을 위해서 매일 우유를 배달시켜 먹고 있는데 매일 새벽 5시에 우유가 배달됩니다. 매일 새벽마다 우유를 일일이 받으러 나갈 수 없어서 문에 우유 투입구를 만들어 우유를 집안에 넣어주도록 했습니다. 주말을 맞아 온 가족이 동해로 1박 2일 여행을 떠났습니다. 혹시나 도둑이 들까 봐 모든 창문과 문을 단단히 걸어잠그고 여행을 떠났습니다. 그런데 여행을 다녀왔더니 온 집안이 쑥대밭이 되어 있고 각종 귀중품이 모두 사라져 버린 게 아니겠습니까? 놀라서 경찰에 신고했더니 경찰은 도둑이 우유 투입구를 통해서 문을 열고 들어온 것으로 판단했습니다. 아니 이럴 수가 우유 투입구로 문을 열다니!!

우유 투입구는 HTTP 포트와 마찬가지입니다. 방화벽으로 모든 포트에 대한 접근을 차단할 수 있지만 웹 서비스를 하려면 반드시 출입이 자유로운 열린 포트가 필요합니다. 우유를 배달하기 위해서도 우유 투입구가 필요합니다. 우유 투입구로 사람이 들어오는 것은 불가능하기 때문에 이를 신뢰하여 신경을 쓰지 않은 것이 도둑이 들게 된 원인입니다. 그 투입구를 통해 사람은 드나들 수 없지만 그 투입구에 맞는 도구를 집어넣어서 문을 여는 것은 가능하기 때문입니다. 만약 보다 신경을 썼더라면 문의 잠금장치에 접근할 수는 없도록 해야 했을 것입니다. 도둑이 보안에 취약한 우유 투입구를 악용하여 도둑질하듯이 크래커는 보안에 취약한 웹 애플리케이션을 통해서 시스템을 공격할 수 있습니다.

그래서 웹 프로그래머가 얼마나 보안에 대해서 신경 써서 웹 사이트를 개발하였느냐가 무엇보다도 중요합니다. 대개 웹 보안 취약점에 대한 해결 방법은 매우 간단합니다. 집을 나가기 전에 모든 창과 문단속을 하듯이 잠금장치를 걸어두는 정도의 간단한 방법으로 취약점을 해결할 수 있습니다. 우리 집은 10층에 있으니 설마 창으로 누가 들어오진 않겠지라고 생각하고 창문을 닫지 않는다면 도둑이 줄을 타고 내려오든가 아니면 가스 배관 등을 타고 올라올지도 모르는 일입니다. 잠금장치를 걸어두는 작은 습관으로 대다수의 웹 해킹을 막을 수 있습니다.

우선 국제 웹 보안 표준 기구인 OWASP(www.owasp.org)에서 잘 정리해둔 PHP 웹 보안 취약점 Top 5를 통해 어떠한 취약점이 있는지를 알아보고 대표적인 경우의 공격 방법과 그 해결책에 대해서 자세히 알아보도록 하겠습니다.

① 원격 코드 실행 (Remote Code Execution)

② 크로스 사이트 스크립팅 (XSS, Cross Site Scripting)

③ SQL 인젝션 (SQL Injection)

④ 안전하지 않은 PHP 설정

⑤ 파일 시스템 공격 (File System Attacks)

원격 코드 실행

원격 코드 실행 취약점은 사용자의 입력으로부터 불러들일 파일을 결정하는 코드가 있는 경우에 발생할 수 있습니다. 주로 PHP에서 include 함수나 fopen과 같은 파일 관련 함수들을 통해서 변수로 넘겨받은 사용자의 입력에 따라서 읽어들일 파일을 결정하게 될 때 발생합니다.

```
include $_POST['filename'];
```

위와 같은 코드는 사용자가 입력한 filename 변수에 따라서 인클루드할 파일이 결정됩니다. 이때 사용자가 올바른 파일의 경로가 아닌 외부 사이트의 악의적인 스크립트 파일을 읽어들여서 크로스 사이트 스크립팅 공격을 할 수 있습니다. 이 취약점은 다른 웹 프로그래밍 언어에서보다 특히 PHP에서 자주 일어납니다. 그 이유는 PHP.ini 파일의 allow_url_open 항목 때문입니다. 이 항목은 URL 형식의 파일 읽기를 가능하게 할 것인지를 결정하는 것인데 PHP 4.0.4 버전부터 기본값으로 "on"으로 설정되어 있기 때문에 URL 형식의 파일 열기가 가능합니다. 그래서 수많은 사이트에서 원격의 파일을 웹 페이지에 삽입하는 공격인 원격 파일 인젝션(Remote File Injection) 공격이 일어났고 많은 사이트가 웹 페이지 변조 공격을 받았습니다. 이로 인해 PHP 4.3.10 버전부터 기본값을 "off" 설정으로 하여 URL 형식의 파일 열기를 금지하고 있습니다.

기본적으로 이 공격에 대한 해결 방법은 allow_url_open 항목을 off 시켜서 일단 해결할 수 있습니다. 그러나 allow_url_open 항목은 단지 URL을 통해서 외부 사이트의 파일을 읽어들일 수 없도록 막는 것이지 파일 경로와 이름이 올바른지를 검증하는 것이 아니므로 파일 업로드 취약점을 이용하여 악성 파일을 업로드하고 이를 통해서 로컬의 파일을 불러들이는 방법으로 공격할 수 있습니다.

따라서 이 취약점을 해결하기 위해서는 파일과 관련된 함수나 제어 구조를 이용할 경우 부득이하게 사용자의 입력에 따라 파일 이름을 결정지어야 한다면 반드시 그 파일의 경로와 파일의 이름이 올바른지 확인하는 구문을 넣어서 검증하고 난 후에 파일을 열 수 있도록 처리해야 합니다.

```
$valid_filename = check_validation ($_POST['filename']); //검증 함수를 만들 것
include $valid_filename;
```

크로스 사이트 스크립팅

크로스 사이트 스크립팅(Cross Site Scripting)은 웹 해킹의 가장 대표적인 공격 방법으로 줄여서 XSS라고 말합니다. XSS는 악의적인 크래커가 클라이언트 측에서 실행되는 스크립트(예를 들면 JavaScript나 VBScript)를 공격하고자 하는 타겟 사이트(Target Site)에 등록하여 사이트에 접속한 사용자들이 자신도 모르게 스크립트를 실행하게 만드는 방법을 말합니다. 그리고 크래커가 등록한 스크립트가 사용자에게 실행될 수 있을 때, XSS 취약점이 있다고 합니다.

사용자에게 스크립트가 실행되도록 하는 것이 뭐가 그리 위험하다는 것인지 잘 이해가 안 되리라 생각합니다. 이 방법을 사용하면 웹 사이트를 변조하거나 사용자의 쿠키나 세션 정보를 훔칠 수 있는데, 쿠키와 세션 정보를 가로채는 경우 이를 통해서 해당 사용자의 계정으로 로그인할 수 있고 특히 관리자의 계정을 가로채는 경우 모든 사용자의 개인정보나 핵심 정보가 유출될 수 있습니다.

아마도 웹 프로그래머라면 한 번쯤 싸이월드나 채팅방에서 스크립트를 삽입하여 스킨을 변경한다든지 음악을 재생한다든지 하는 장난을 해 본 경험이 있을 거라 생각합니다. 바로 이와 같은 행위가 XSS 취약점을 이용한 대표적인 공격방법입니다.

이 방법에 대해서는 잠시 후에 자세히 다루도록 하겠습니다.

SQL 인젝션

인젝션(Injection) 취약점이란 사용자가 입력한 값을 통해서 명령어를 만들어 사용하는 경우 이를 이용하여 명령어를 악의적으로 수정하거나 추가하는 방법을 말합니다. 대표적인 경우가 SQL 인젝션입니다. 일반적으로 사용자 인증을 위해서 사용자가 입력한 아이디와 비밀번호를 가지고 SQL 쿼리를 만들어 사용합니다. 이 쿼리는 데이터베이스로 전송되고 실행되기 때문에 만약 SQL 쿼리를 수정할 수 있다면 데이터베이스의 데이터를 검색하거나 수정, 삭제하는 것도 가능해집니다. 또한 사용자 인증을 무효화시켜서 관리자로 로그인할 수도 있습니다.

이 방법은 다음 절에서 자세히 다루도록 하겠습니다.

안전하지 않은 PHP 설정

개인이 운영하는 서버나 중소규모의 사이트에서는 개발의 편의를 위해서 프로그래밍하기 편리한

PHP 설정을 해두는 경우가 많습니다. 대표적인 예가 앞서 다룬 원격 파일 실행 취약점을 발생시키는 allow_url_open 설정입니다. 취약점을 발생시키는 PHP 설정은 다음과 같습니다.

① register_globals

② allow_url_open

③ magic_quotes_gpc

④ safe_mode, open_basedir

register_globals 항목은 최근에 대부분 Off가 기본값으로 설정되어 있습니다. 그러나 초기에는 On이 기본값이었으며 이는 외부로부터 넘어온 변수들을 간단히 $varname과 같은 방법으로 접근할 수 있도록 자동으로 글로벌 변수화해주는 기능입니다. 이 기능은 일일이 변수의 출처에 따라 접근하는 $_GET, $_POST, $_COOKIE와 같은 방식에 비해서 모든 변수에 간단하게 접근할 수 있는 장점이 있어서 매우 유용하게 사용되었습니다. 그러나 이 방법은 변수의 출처가 불명확하기 때문에 보안 문제가 발생할 수 있습니다.

```
<?
  if ($cookie == 'brown') echo "로그인 되었습니다.";
  else echo "로그인에 실패하였습니다.";
?>
```

위와 같이 쿠키값을 읽어서 로그인 상태를 유지하는 코드가 있다면 $cookie 변수가 쿠키로부터 만들어진 값인지 사용자의 입력으로 인한 GET이나 POST 방식으로 넘어온 변수인지 구분할 수가 없습니다. 그래서 다음과 같은 방법으로 로그인을 성공시킬 수 있습니다.

```
http://victim.com/hacking/login.php?cookie=brown
```

[그림 15-1] register_globals=On 취약점을 이용한 로그인

그러나 register_globals 항목을 Off로 설정하면 $cookie 변수는 출처가 표시되지 않았으므로 로컬 변수로 인식되기 때문에 올바르게 동작시키려면 쿠키로부터 생긴 변수인 $_COOKIE['cookie'] 변수로 변경해야 합니다.

```
<?
  if ($_COOKIE['cookie'] == 'brown') echo "로그인 되었습니다.";
```

```
    else echo "로그인에 실패하였습니다.";
?>
```

이렇게 수정되면 GET 방식으로 넘어온 변수는 $_GET['cookie'] 변수로 접근할 수 있고 이는 $_COOKIE['cookie'] 변수와 구분되기 때문에 다음과 같이 올바르게 동작합니다.

[그림 15-2] register_globals = Off로 취약점 제거

두 번째 항목인 allow_url_open은 이미 언급하였고 세 번째 항목인 magic_quotes_gpc 항목은 On으로 설정해두면 GET, POST, COOKIE로 만들어진 변수에 작은따옴표('), 따옴표("), 역슬래시(\), 널문자 등이 있을 경우 자동으로 이 특수문자 앞에 역슬래시를 추가해 줍니다. 만약 이 항목이 Off로 되어 있다면 사용자 입력에 대해서 이러한 처리를 해주지 않게 되는데 이 경우 SQL Injection 취약점을 발생시키게 됩니다. 이 부분에 대해서는 뒤에서 자세히 다룹니다.

마지막으로 safe_mode와 open_basedir 설정은 여러 사람이 공유해서 사용하는 웹 서버에 적용되는 설정입니다. safe_mode는 파일에 접근할 때 파일의 소유권을 확인하여 파일에 대한 접근을 보다 강력하게 확인하도록 하는 등의 기능을 합니다. 예를 들어 /etc/passwd 파일과 같은 시스템의 중요한 파일에 접근할 수 없도록 막는 역할을 합니다. 이와 유사하게 open_basedir 설정은 지정된 디렉토리 외의 파일 접근을 막는 역할을 합니다. 그러나 safe_mode는 PHP 6.0.0 버전부터 제거될 예정입니다.

이처럼 간단한 PHP의 설정만으로 취약점을 해결할 수 있고 반대로 취약점을 드러낼 수도 있습니다. PHP 버전을 최신으로 유지하면 대부분 설정이 보안에 안전한 방향으로 설정되므로 취약점을 해결할 수 있습니다. 그런데 특히 register_globals처럼 기존의 코드를 수정하지 않으면 동작하지 않는 설정때문에 코드 수정의 번거로움을 피하고자 기본값을 다시 On으로 바꾸는 일은 되도록이면 자제하기 바랍니다.

파일 시스템 공격

파일 시스템 공격은 주로 웹 서비스를 통해서 디렉토리나 파일에 접근할 수 있어서 발생하는 취약점입니다.

① 로그 파일 혹은 시스템 파일 및 설정 파일에 접근

② 세션 파일에 접근

③ 업로드된 파일에 접근

/etc/passwd 파일과 같은 시스템 파일에 웹 서비스를 통해서 접근할 수 있다면 무작위적인 로그인 공격을 받을 수 있습니다. 만약 쉬운 형식의 비밀번호를 사용한다면 쉽게 시스템의 사용자 계정을 훔칠 수 있습니다. 또한 웹 서버 설정 파일인 http.conf나 php.ini 파일과 같은 파일에 접근할 수 있다면 설정상의 취약점 등을 통해서 공격을 받을 수 있습니다. 따라서 "./../../etc/passwd" 같은 형식으로 웹 서비스로부터 접근할 수 없도록 막아야 합니다.

세션은 서버 측에 저장되는 값이기 때문에 외부에서 접근할 수 있는 디렉토리에 세션값을 저장한다면 이를 통해서 세션 정보를 가로채서 웹 사이트의 사용자 계정을 마음껏 사용할 수도 있습니다. 그렇기 때문에 세션 파일은 반드시 홈 디렉토리 상위의 웹 서비스에서 직접 접근이 불가능한 디렉토리에 만들어야 합니다.

마지막으로 업로드된 파일에 URL을 통해서 접근할 수 있는 경우 PHP 파일과 같은 서버 측 스크립트를 통해서 시스템 공격이 가능해질 수 있습니다. 되도록이면 업로드된 파일은 URL을 통해서 접근이 불가능한 디렉토리에 설정하는 것이 바람직합니다. 파일 업로드 취약점에 대해서는 조금 후에 자세히 다루겠습니다.

이제 실제 공격의 예를 통해서 어떤 식으로 해킹이 이루어지며 그에 대한 방어법은 어떤 것이 있는지 알아보도록 하겠습니다.

Section 02 관리자 페이지 인증 취약점

대부분의 사이트에는 웹 사이트의 효율적인 관리를 위해서 관리자 페이지를 별도로 두고 있습니다. 예를 들면 공지사항의 추가라든지 사이트 사용자의 관리 혹은 고객이 주문한 상품을 관리하는 등 웹 사이트의 여러 가지 기능을 관리하기 위해서 관리자 페이지를 만들곤 합니다. 관리자 페이지를 만들면 제일 먼저 관리자인지 일반 사용자인지를 구분해야 하기 때문에 인증 절차가 반드시 필요합니다. 그런데 이때 초보 프로그래머가 사용자 인증에 대한 경험이 부족하여 관리자 인증 처리를 미흡하게 해두는 경우가 있습니다.

취약점의 예

대표적으로 관리자 페이지 전체에 대해서 사용자 인증을 하지 않고 첫 페이지에 대한 접근에 대해서만 사용자 인증을 처리하는 경우입니다.

```
http://www.victim.com/admin/index.php
```

예를 들어 위와 같은 주소로 관리자 페이지에 접근한다고 했을 때 초보 프로그래머들이 index.php 파일에만 사용자 인증 소스를 삽입하는 경우를 말합니다. 즉, admin 디렉토리에 있는 다른 관리자 페이지에 대해서는 관리자 인증을 하지 않는다는 뜻입니다. 그래서 만약 사용자 목록 페이지인 http://www.victim.com/admin/userlist.php 파일에 URL을 통해서 직접 접근하게 되는 경우 관리자 인증 절차 없이 곧바로 사용자 정보를 볼 수 있게 되는 것을 말합니다.

'이같이 어리석은 짓을 누가 할까?'라고 생각하겠지만 실제로 위와 같은 방법으로 관리자 페이지를 사용하는 경우가 많습니다. 덧붙여 관리자 페이지의 경우 위의 예에서와 마찬가지로 다음과 같은 형식으로 접근하는 경우도 많습니다.

```
http://www.victim.com/admin.html
http://www.victim.com/admin/index.html
http://www.victim.com/admin/admin.html
http://www.victim.com/administrator/index.html
http://www.victim.com/manager/index.html
http://www.victim.com/master/index.html
```

그래서 크래커는 쉽게 관리자 페이지의 접근 경로를 추측할 수 있고 무차별 공격(Brute Force, 무차별적으로 대입, 사전 등의 방법을 통해서 공격하는 해킹 기법)을 통해서 관리자 페이지의 접근 여부를 확인할 수 있습니다.

취약점 해결 방법

관리자 페이지 우회를 통한 관리자 인증의 회피는 예에서 보다시피 프로그래머의 미숙함이나 실수로 발생합니다. 따라서 다음과 같은 방법을 통해서 쉽게 해결할 수 있습니다.

① 관리자 페이지 접속 경로를 쉽게 유추할 수 없는 주소로 바꾼다.

② 관리자 페이지 전체 파일에 사용자 인증 구문을 추가한다.

③ 관리자 디렉토리에 접근할 때 .htaccess를 이용하여 사용자 인증을 처리한다.

Section 03

쿠키를 이용한 세션 관리 취약점

사용자 인증 단원에서 쿠키를 이용하여 인증 상태를 유지하는 방법에 대해서 배웠습니다. 로그인이 검증되면 쿠키를 만들어주는 방법으로 로그인 상태를 유지했습니다. 그런데 이 방법에는 큰 위험요소가 있습니다.

```
<?
    if ($_GET['userid'] == 'brown' and $_GET['password'] == '1111') {
    setcookie("userid", $_GET['userid']);
    echo "로그인에 성공하였습니다.";
    echo "<a href='login_check_2.php'>Next</a>";
}
else echo "로그인에 실패하였습니다.";
?>
```

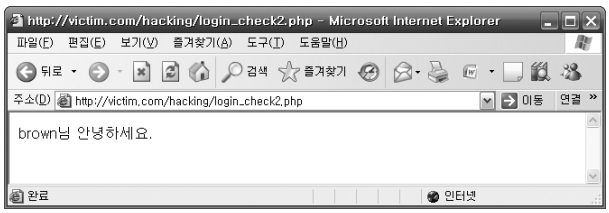
위와 같이 로그인에 성공한 경우 userid 쿠키에 사용자의 아이디를 저장하여 이를 통해서 인증 상태를 유지한다고 합시다.

```
<?
  if ($_COOKIE['userid']) echo $_COOKIE['userid'] . "님 안녕하세요.";
  else { echo "로그인 하세요."; exit; }

  if ($_COOKIE['userid'] == 'admin') echo "<BR>관리자로 로그인되었습니다.";
?>
```

실제로 많은 개발자가 위와 같은 코드로 로그인 상태를 유지합니다. 단순히 쿠키가 있는지 없는지를 통해서 로그인을 유지하는데 코드를 실행해서 로그인을 해보면 다음과 같이 로그인 상태가 제대로 유지되는 것을 확인할 수 있습니다.

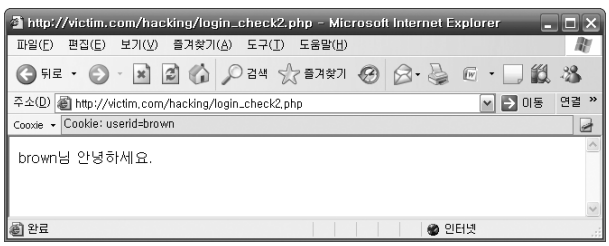
[그림 15-3] 쿠키를 이용한 로그인 유지

Next 버튼을 눌러서 로그인 상태가 유지되는지를 확인해보면 다음과 같습니다.



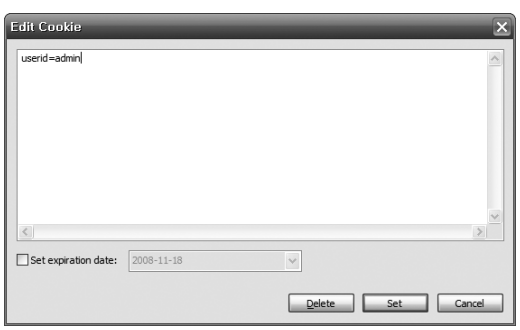
[그림 15-4] 쿠키를 이용한 로그인 유지 확인

그런데 이같은 경우에 쿠키를 살짝 고치는 것으로 쉽게 거짓으로 로그인할 수 있습니다. 쿠키는 사용자의 컴퓨터 측에 저장되므로 사용자가 쉽게 고칠 수 있기 때문입니다. 쿠키를 쉽게 수정할 수 있는 기능을 가진 Cooxie 툴바를 가지고 쿠키값을 변경해 보겠습니다.



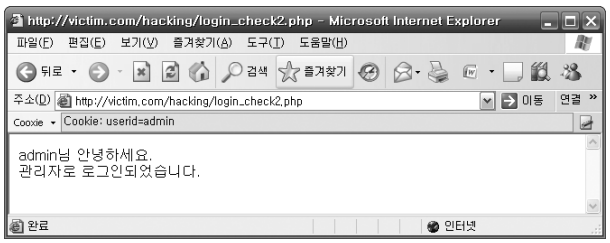
[그림 15-5] Cooxie 툴바

Cooxie 툴바를 사용하면 위와 같이(설정에 따라서 보이는 모양은 다릅니다.) 현재 설정된 쿠키의 값이 보입니다. 그리고 오른쪽에 있는 버튼을 통해서 쿠키를 수정할 수 있습니다.



[그림 15-6] Cooxie 툴바를 이용한 쿠키값 변경

userid=brown이라고 되어 있는 값을 위와 같이 userid=admin이라고 변경해 봅시다. 새로 고침을 통해서 다시 페이지를 확인하면 다음과 같이 관리자로 로그인된 것을 확인할 수 있습니다.



[그림 15-7] 쿠키 수정을 통한 관리자 로그인

이처럼 쿠키를 통해서 단순히 검증하는 구조에서는 간단하게 크래킹이 가능함을 알 수 있습니다. 이를 방지하기 위해서 쿠키를 암호화하는 방법을 택할 수 있습니다.

다음은 아이디를 MD5 해시 함수를 통해서 해시스트링을 만드는 함수입니다.

```
function make_sid($id) {

  $text = $id . "brown"; //사용자의 시그너쳐 문자열

  SetCookie("userid",$id,0,"/");
  SetCookie("sid",md5($text),0,"/");

}
```

이 함수를 통해서 복호화가 불가능한 sid 문자열을 만들 수 있습니다. 즉, 사용자의 ID와 특수한 문자열(여기서는 brown)을 이용하여 userid가 변경되는 것을 방지합니다.

```
<?
  function make_sid($id) {

    $text = $id . "brown";

    SetCookie("userid",$id,0,"/");
    SetCookie("sid",md5($text),0,"/");

  }

    if ($_GET['userid'] == 'brown' and $_GET['password'] == '1111') {
      make_sid($_GET[userid]);
      echo "로그인에 성공하였습니다.";
      echo "<a href='login_check_md5_2.php'>Next</a>";
    }
    else echo "로그인에 실패하였습니다.";
?>
```

SID를 검증하기 위해서는 다음과 같은 함수가 필요합니다.

```
function check_sid() {

    $get_userid = $_COOKIE['userid'];
    $get_sid = $_COOKIE['sid'];

    $get_userid = $get_userid . "brown";
    $real_sid = md5($get_userid);

    if ($get_sid == $real_sid) return true;
    else return false;
}
```

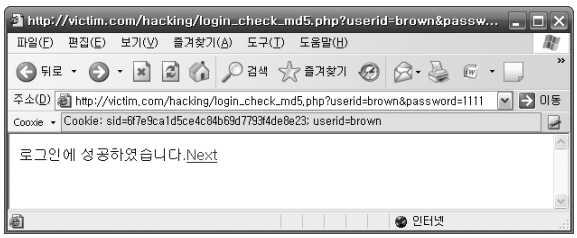
이 함수는 make_sid 함수에서 했던 MD5 해시 스트링을 다시 한번 만들어서 서로 같은지 비교합니다. 즉, 중간에 userid 값을 변경했다면 다시 MD5 해시 스트링을 만들었을 때 해시 스트링이 달라지기 때문에 수정이 된 것임을 알 수 있습니다.

완성된 검증 소스는 다음과 같습니다.

login_check_md5.php

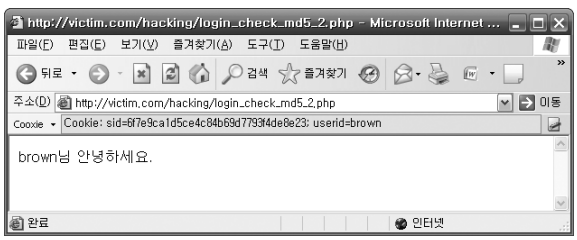
```
1  <?
2    function check_sid() {
3
4      $get_userid = $_COOKIE['userid'];
5      $get_sid = $_COOKIE['sid'];
6      if (!$get_userid) return false;
7      if (!$get_sid) return false;
9
10     $get_userid .="brown";
11     $real_sid = md5($get_userid);
12
13     if ($get_sid == $real_sid) return true;
14     else return false;
15   }
16
17   if (check_sid()) echo $_COOKIE['userid'] . "님 안녕하세요.";
18   else { echo "로그인 하세요."; exit; }
19
20   if ($_COOKIE['userid'] == 'admin') echo "<BR>관리자로 로그인되었습니다.";
21 ?>
```

이 수정된 코드를 통해서 로그인을 검증해보면 다음과 같습니다.



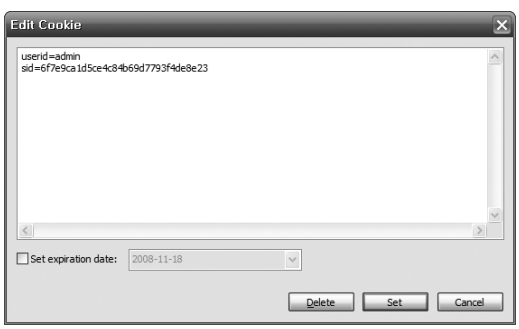
[그림 15-8] MD5 해시를 이용한 쿠키 암호화

로그인에 성공하면 위와 같이 SID가 생깁니다.



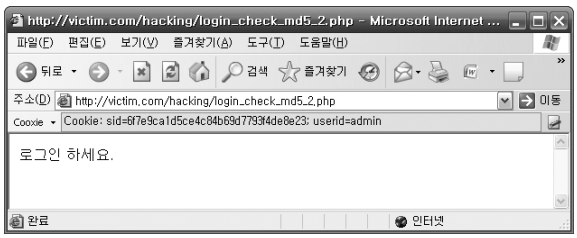
[그림 15-9] MD5 해시를 이용한 로그인 유지

Next를 클릭하면 위와 같이 사용자 인증 상태가 유지됨을 알 수 있습니다. 이제 cooxie 툴바를 이용하여 쿠키를 변경해 보도록 합시다.



[그림 15-10] 쿠키값 변경

위와 같이 userid를 admin으로 변경하였습니다. 기존의 코드는 이처럼 단순히 수정하는 것만으로 로그인이 성공하였습니다. 수정된 코드는 어떤 결과를 보일지 확인해 봅시다.



[그림 15-11] MD5 검증을 통해서 쿠키 변조를 확인

userid가 admin으로 바뀌면서 admin 문자열로 생성되는 sid 값과 brown 문자열로 생성되는 SID 값이 서로 달라서 로그인 상태가 유지되지 않는 것을 확인할 수 있습니다.

이처럼 간단한 암호화 구문을 통해서 쿠키를 이용한 로그인 유지가 한층 안전해졌습니다. 그러나 이 코드도 사실 문제를 가지고 있습니다. 사용자 혹은 관리자의 쿠키 정보를 알아낼 수 있다면 sid를 쿠키로 구워서 로그인할 수 있기 때문입니다. 이 방법에 대해서는 XSS에서 다루도록 하겠습니다.

만약 XSS 공격이 불가능한 웹 사이트라면 이 방법을 통해서 최대한 가벼운 방법으로 로그인 유지를 안전하게 할 수 있습니다. 그러나 서버에 부담이 되지 않는다면 쿠키와 세션 장에서 배운 세션을 이용한 로그인 유지 방법을 사용하는 것이 좋습니다.

Section 04 세션 하이재킹

쿠키는 클라이언트 측에 저장되기 때문에 수정할 수 있어서 앞서 배운 것과 같이 쉽게 공격을 받을 수 있습니다. 그러나 세션은 서버 측에 저장되기 때문에 서버에 접근하여 수정할 수 있는 권한이 없다면 세션을 수정하는 것은 매우 어려운 일이 될 것입니다. 이러한 장점 때문에 많은 사이트에서 쿠키보다는 세션을 이용하여 로그인을 유지하고 있습니다.

그러나 XSS를 통해서 세션 스니핑을 하여 세션 정보를 얻어내는 경우 세션 하이재킹 공격을 받을 수 있습니다.

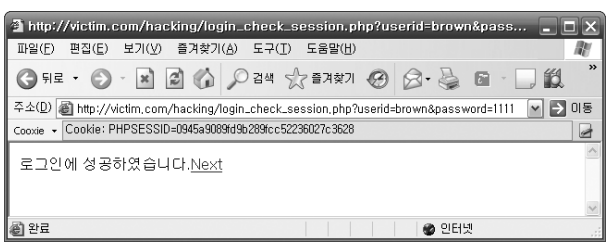
```
<?
    if ($_GET['userid'] == 'brown' and $_GET['password'] == '1111')
    {session_start();
    $_SESSION['userid'] = $_GET['userid'];
    echo "로그인에 성공하였습니다.";
    echo "<a href='login_check_session_2.php'>Next</a>";
```

```
    }
    else echo "로그인에 실패하였습니다.";
?>
```

로그인에 성공하게 되면 세션이 시작되며 userid라는 세션 변수가 등록됩니다. 로그인을 유지하기 위해서 세션 변수인 userid를 확인하여 값이 있으면 로그인이 된 것으로 간주합니다.

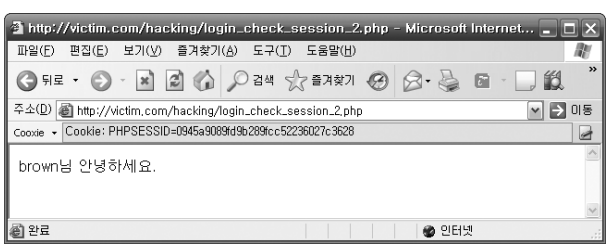
```
<?
    session_start();

    if (!empty($_SESSION['userid'])) echo $_SESSION['userid'] . "님 안녕하세요.";
    else { echo "로그인 하세요."; exit; }

?>
```

그러면 이 소스를 통해서 사용자 A로 로그인을 해보겠습니다.



[그림 15-12] 사용자 A로 로그인

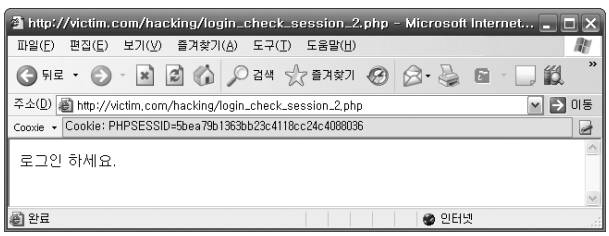
쿠키를 보면 PHPSESSID가 자동으로 등록된 것을 확인할 수 있습니다. 이 세션 ID를 통해서 세션 변수들을 관리합니다.



[그림 15-13] 세션을 이용한 로그인 유지

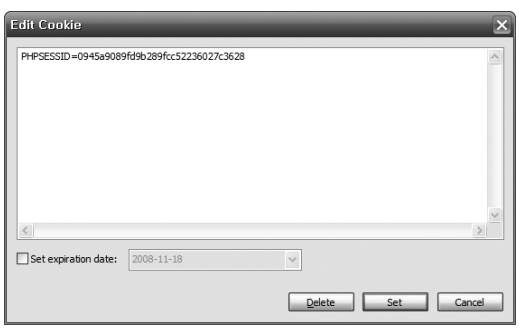
사용자 A는 로그인한 후 위와 같이 로그인이 유지된 상태로 웹 사이트를 돌아다니고 있다고 가정합시다. 사용자 A가 브라우저를 닫고 20분 이상의 시간이 흘렀거나 로그아웃을 통해서 세션을 제거했다면 PHPSESSID 값은 더 이상 무용지물이 되고 맙니다. 그러나 이처럼 사용자가 웹 사이트 내에 있거나 브라우저를 닫고 나간 지 20분이 되지 않았다면 세션 파일은 여전히 서버에 남아있게 되

고 XSS를 통해서 알아낸 PHPSESSID를 통해서 사용자 A의 아이디인 brown으로 로그인할 수 있게 됩니다.



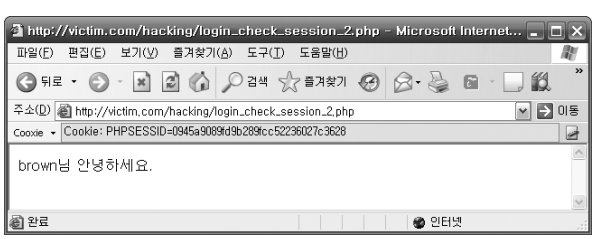
[그림 15-14] 사용자 B의 로그인이 필요한 페이지 접속

사용자 B는 위의 그림과 같이 로그인이 필요한 페이지에 접속하면 로그인 인증이 되지 않았기 때문에 로그인을 하라는 메시지를 보게 됩니다. 그리고 접속에 따른 자신의 세션 ID를 갖게 됩니다. 만약 XSS를 통해서 사용자 A의 PHPSESSID를 알게 되었다면 쿠키 변조를 통해서 사용자 A의 아이디로 로그인할 수 있습니다.



[그림 15-15] 사용자 A의 PHPSESSID 값으로 쿠키를 변조

사용자 A의 PHPSESSID 값으로 쿠키를 변조하고 페이지를 새로 고침하면 다음과 같이 정상적으로 로그인된 것을 확인할 수 있습니다.



[그림 15-16] 세션 하이재킹을 통한 로그인

위와 같이 세션을 통한 로그인 인증의 경우 세션 ID가 유출된다면 쿠키를 변조하여 세션 변수들에

접근할 수 있기 때문에 안전하지 않습니다. 따라서 세션을 이용하여 로그인 인증을 한다면 세션 파일의 관리를 철저히 하고 세션 ID가 유출되지 않도록 신경 써야 합니다. 그렇게 하려면 XSS 공격이 불가능하도록 처리하는 것이 중요합니다.

Section 05 크로스 사이트 스크립팅(XSS)

게시판에 글을 쓸 때 사용자가 입력한 값을 그대로 데이터베이스에 등록하고 글 읽기 페이지에서 그대로 출력을 해준다고 가정해 봅시다. 크래커는 글의 내용에 다음과 같은 자바스크립트 코드를 삽입합니다.

```
<script> location.href='http://cracker-site.com'; </script>
```

사이트 방문자가 해당 글을 읽으려고 하면 글의 내용은 온데간데 없고 엉뚱한 사이트로 이동한 것을 깨닫게 됩니다. 만약 사용자의 글이 사이트의 첫 페이지에 최신 글의 형태로 제목이 노출된다면, 제목에 위와 같은 코드를 삽입하여 사이트에 접속했을 때 자신이 원하는 사이트로 이동시켜서 해당 사이트를 무력화시킬 수도 있습니다.

XSS 취약점은 사용자가 입력한 값을 검증하지 않고 그대로 출력하는 것이 원인입니다. 그래서 XSS 취약점을 해결하는 방법은 간단합니다. 사용자로 하여금 클라이언트 측에서 실행되는 스크립트를 삽입할 수 없도록 하는 것입니다.

일반적인 입력 값이라면 다음과 같은 방법으로 간단하게 처리할 수 있습니다.

```
<?=strip_tags($user_input);?>
```

바로 strip_tags() 함수를 이용하여 모든 HTML 태그와 스크립트를 무력화시키는 것입니다. HTML을 허용하는 경우가 아닌 모든 항목에 대해서 이와 같은 코드를 넣어서 간단히 XSS 취약점을 보완할 수 있습니다. 게시판을 만들면서 이미 다루었듯이 HTML을 일부 허용하고자 한다면 허용하고자 하는 태그를 다음과 같이 등록해주면 됩니다.

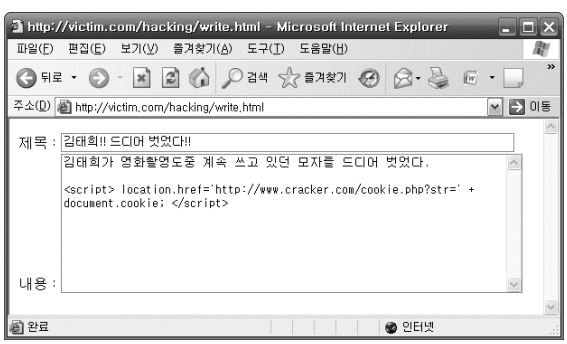
```
<?=strip_tags($user_input, "<img><b>");?>
```

그러나 가능하면 모든 HTML을 막는 것이 제일 안전합니다. 왜냐하면 다음과 같은 방법으로 XSS 공격을 시도할 수 있기 때문입니다.

```
<img src="없는파일이름.gif" onError="location.href='http://cracker-site.
com'">
```

자바스크립트에는 다양한 이벤트 함수가 있는데 이를 이용하여 사용자가 페이지를 보기만 해도 스크립트를 실행할 수 있기 때문입니다. 위의 방법은 존재하지 않는 이미지를 소스로 지정해서 파일을 찾을 수 없는 에러를 발생시킵니다. 에러가 발생하면 onError 이벤트가 발생하고 등록된 스크립트가 실행되어 XSS 공격이 가능해집니다. 이를 막으려면 모든 가능한 이벤트 핸들러를 사용할 수 없도록 막아야 합니다. 일반적으로 "OnError="와 같은 문자열이 들어오면 str_replace 함수나 preg_replace 함수 등을 사용하여 "XXnError="와 같이 변경해주는 방법을 사용합니다. 이때 중요한 것은 대소문자 구분을 하지 말고 처리하는 것입니다. 대소문자 구분을 하게 되면 OnError, onError, ONError, onerror 등이 모두 다르게 처리되기 때문에 고려되지 않은 문자열에 대해서는 처리가 되지 않을 수 있습니다.

XSS 공격이 가능해지면 쿠키 스니핑이나 세션 스니핑을 통해서 사용자의 정보를 훔쳐낼 수 있습니다. 예를 들면 사람들이 많이 접근하는 게시판에 다음과 같은 글을 남깁니다.



[그림 15-17] XSS를 이용한 쿠키 스니핑

사람들이 관심을 가질만한 제목을 통해 많은 사람이 글을 읽게 합니다. 글을 읽은 사람들의 쿠키 정보는 크래커의 사이트에 계속해서 기록됩니다. 크래커는 이를 통해서 사용자 혹은 관리자의 쿠키를 간단히 알아낼 수 있습니다. 이를 이용하여 앞서 배운 쿠키 변조나 세션 하이재킹을 통해서 사용자의 아이디로 로그인할 수 있습니다.

따라서 위와 같이 사용자가 입력하는 모든 항목에 대해서 스크립트를 사용할 수 없도록 처리하는 것이 가장 바람직합니다. 대부분의 경우 많은 사람들이 입력하는 내용 부분 정도에만 이러한 처리를 한다는 것입니다. 제목이나 이메일 주소 입력 상자 등 모든 입력에 대해서 처리하는 것이 중요합니다.

Section 06 SQL Injection

SQL Injection은 XSS와 함께 가장 대표적인 웹 해킹 기법으로 데이터베이스에 사용되는 SQL 쿼리를 사용자가 변조하는 공격 기법입니다. 사용자 인증의 경우를 생각해보면 사용자가 입력한 아이디와 비밀번호를 가지고 검증에 필요한 SQL 쿼리문을 작성합니다. 이와 유사하게 데이터베이스와 연동하는 대부분의 웹 애플리케이션들은 사용자의 입력을 바탕으로 SQL 쿼리문을 작성합니다. 그런데 사용자의 입력값을 검증하지 않으면 사용자가 임의로 SQL 쿼리문을 수정할 수 있습니다. 이렇게 사용자가 SQL 쿼리문을 마음대로 수정할 수 있으면 데이터베이스의 자료를 조회하거나 수정하는 등의 많은 작업을 할 수 있습니다.

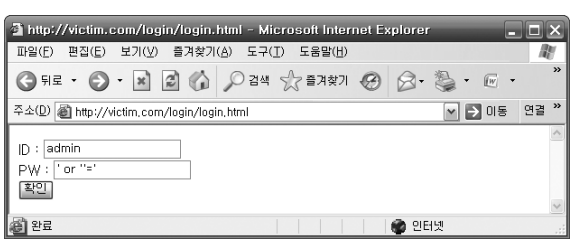
작은따옴표를 이용한 공격

가장 대표적인 SQL Injection 방법은 사용자 인증을 우회하는 것입니다. 다음과 같은 사용자 인증 SQL 쿼리문이 있다고 합시다.

```
SELECT count(*) FROM users
WHERE userid = '$_POST[userid]' and userpw = '$_POST[userpw]'
```

이 쿼리문은 사용자의 입력을 통해 userid와 userpw 값을 얻어와서 users 테이블에 해당 아이디와 비밀번호를 가진 사용자가 존재하는지 여부를 판단합니다. 이 경우 아이디와 비밀번호가 모두 일치해야만 로그인이 가능하지만 SQL Injection 공격을 통해서 비밀번호를 알지 못하더라도 해당 아이디로 로그인할 수 있습니다.

공격자는 로그인하고자 하는 아이디를 입력하고 비밀번호 항목에 다음과 같은 코드를 삽입합니다.



[그림 15-18] SQL Injection

비밀번호 항목에 ' or ''=' 과 같은 값을 넣으면 SQL 쿼리문은 다음과 같이 수정됩니다.

```
SELECT count(*) FROM users
WHERE userid = 'admin' and userpw = '' or ''=''
```

위와 같이 쿼리문이 변경되면 비밀번호 항목에서 비록 거짓이 되지만 "=" 이라는 항상 참인 값 때문에 전체 WHERE 절은 참이 되어서 admin 아이디로 로그인됩니다.

[그림 15-19] SQL Injection 결과

또한 MySQL에는 SQL 주석이 있는데 하이픈 두 개에 이은 스페이스(--) 입니다. 이 SQL 주석을 이용하면 작은따옴표를 처리하기 위해서 마지막에 작은따옴표를 쌍으로 만들어주는 일("=')을 할 필요가 없습니다. 그렇기 때문에 보다 다양한 SQL Injection이 가능해 집니다.

```
SELECT count(*) FROM users
WHERE userid = 'admin' and userpw = '' or 1=1 -- '
```

위처럼 꼭 따옴표로 끝내지 않더라도 주석(--) 뒤는 모두 무시되기 때문에 올바르게 수행됩니다. 이 방법은 언제나 참이 되는 조건절을 추가하므로 해서 사용자 인증을 무력화시키는 방법입니다.

이 방법을 막으려면 제일 먼저 사용자가 입력한 작은따옴표가 쿼리문에 직접 반영되지 않도록 해야 합니다. 지금까지의 공격 방법은 모두 작은따옴표를 추가하여 개발자가 작성한 쿼리문을 확장하는 방법이었습니다. 만약 여기에 작은따옴표를 사용할 수 없다면 위와 같은 공격 기법도 무용지물이 될 것입니다.

이에 PHP에서는 magic_quotes_gpc 항목과 같은 Magic Quote 설정이 있습니다. Magic Quote를 설정하면 GET, POST, Cookie 등을 통해서 넘어오는 작은따옴표에 모두 역슬래시(\)를 자동으로 추가하여 줍니다. 그래서 단순히 작은따옴표로 쿼리문을 수정하는 것을 막을 수 있습니다.

숫자 형식의 비교문을 이용한 공격

Magic Quote만으로 모든 SQL Injection을 막을 수 있는 것은 아닙니다. 많은 사이트가 이를 착각하여 많은 SQL Injection 공격의 대상이 되고 있습니다. 예를 들어 Magic Quote가 설정된 서버에서 다음과 같은 쿼리문에 대해 살펴봅시다.

```
DELETE * FROM threadboard WHERE id = {$_POST[id]}
```

이 SQL 쿼리문의 경우에는 해당 게시물을 지우고자 게시물 번호인 id 값을 넘겨주고 있습니다. 게시판은 해당 게시물을 삭제하기 위해서 id 값을 게시판이 넘겨주는데 만약 사용자가 이 id 값을 변경할 수 있다면 공격자는 자신이 원하는 모든 게시물을 지울 수 있을 것입니다.

실제로 우리가 구현한 계층형 게시판 소스를 통해서 살펴보겠습니다. 게시물을 삭제하는 파일인 del.php를 열어보면 다음과 같은 코드가 존재합니다.

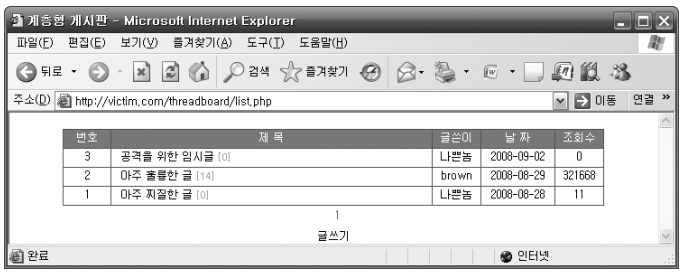
```
$result=mysql_query("SELECT pass FROM $board WHERE id=$_GET[id]",
$conn) or ErrorMessage('글을 삭제하는데 실패하였습니다.');
$row=mysql_fetch_array($result);

if ($_POST[pass]==$row[pass] )
{
  $conndel = "DELETE FROM $board WHERE id=$_GET[id] ";
  $result=mysql_query($conndel, $conn)
  or ErrorMessage('글을 삭제하는데 실패하였습니다.');;
}
else
{
ErrorMessage('비밀번호가 틀립니다.');
}
```

이 코드는 id 값을 통해서 우선 비밀번호를 알아내고 그 비밀번호와 사용자가 입력한 비밀번호가 동일한지 여부를 판단하여 게시물을 삭제하고 있습니다. 사실 검증도 제대로 하고 있어서 문제가 없어 보이지만 실제로는 SQL Injection 공격에 노출되어 있습니다.

[그림 15-20] SQL Injection을 이용한 게시판 공격

위와 같이 게시판에 두 개의 글(실제로는 많아도 관계없습니다.)이 있다고 합시다. 우리가 "나쁜 놈"이라고 생각하고 질투나는 "아주 훌륭한 글"을 삭제해 보도록 합시다.

글이 두 개 밖에 없지만 많이 있다고 가정하고 우선 테스트 글을 하나 등록합니다. 테스트 글을 등록하는 이유는 가장 최근에 등록된 글이 필요하기 때문입니다. 그 이유에 대해서는 잠시 후에 알게 될 것입니다. 일단 새로 글을 하나 등록합니다. 자신의 글 이후에 다시 다른 글이 등록되면 안 되니까 되도록이면 사람들이 잠든 밤에 등록합니다.



[그림 15-21] 공격을 위한 임시글 등록

게시물이 등록되면 글 삭제를 위해 비밀번호를 입력하는 페이지로 이동합니다. 먼저 위의 검증 코드에서 핵심이 되는 두 개의 SQL 쿼리문을 따로 살펴보면 다음과 같습니다.

```
SELECT pass FROM $board WHERE id=$_GET[id]
DELETE FROM $board WHERE id=$_GET[id]
```

일단 우리가 목표로 하는 것은 두 번째 삭제 쿼리문을 통해서 2번 글인 "아주 훌륭한 글"을 삭제하는 것입니다. 그런데 id 값을 바꿔치기해서 2로 변경하면 비밀번호 검증 부분까지 바뀌게 되어 비밀번호 비교 부분을 넘어갈 수가 없습니다. 따라서 임시적인 글을 통해서 이 비밀번호 검증 부분을 무사히 통과해야 합니다.

```
SELECT pass FROM $board WHERE id=3 or id=2
DELETE FROM $board WHERE id=3 or id=2
```

id 값을 위와 같이 변경하여 3번 글과 2번 글을 모두 지우는 것을 시도합니다. 그런데 문제는 ORDER에 따라서 SELECT 문의 결과가 2번 글의 비밀번호보다 먼저 출력되므로 이를 회피하기 위해서 다음과 같이 정렬을 역으로 하여 게시물의 가장 최근 글(우리가 입력한 글)이 제일 먼저 출력되게 합니다.

```
SELECT pass FROM $board WHERE id=3 or id=2 ORDER BY id DESC
DELETE FROM $board WHERE id=3 or id=2 ORDER BY id DESC
```

이렇게 되면 3번 글의 비밀번호를 통해서 첫 번째 비밀번호 비교 검증을 통과할 수 있고(왜냐하면 3번 글의 비밀번호는 우리가 알고 있으니까) 3번 글과 함께 2번 글이 같이 지워집니다.

[그림 15-22] SQL Injection을 이용한 게시물 삭제

비록 Magic Quote가 설정되어 있어서 작은따옴표를 추가하는 Injection 공격은 할 수 없지만 이처럼 숫자 형식의 값을 변경하는 것으로도 충분히 SQL Injection 공격이 가능합니다. 따라서 숫자 형식의 값은 반드시 숫자인지를 확인하는 구문이 필요합니다. 특별히 MySQL을 사용하는 경우에는 숫자 항목도 작은따옴표로 둘러싸는 방법으로 이러한 공격을 회피할 수 있습니다.

위의 소스를 안전하게 수정하면 다음과 같습니다.

```
$id = (int)$_GET[id];
$result=mysql_query("SELECT pass FROM $board WHERE id=$id", $conn)
  or ErrorMessage('글을 삭제하는데 실패하였습니다.');
$row=mysql_fetch_array($result);

if ($_POST[pass]==$row[pass] )
{
   $conndel = "DELETE FROM $board WHERE id=$id ";
   $result=mysql_query($conndel, $conn)
   or ErrorMessage('글을 삭제하는데 실패하였습니다.');;
}
else
{
  ErrorMessage('비밀번호가 틀립니다.');
}
```

UNION을 이용한 공격

데이터베이스의 UNION 키워드를 이용하여 다른 사용자의 비밀번호를 알아내 보도록 하겠습니다. 이 공격은 앞서 두 가지 공격 방법에 대해서 대처했을 때는 사용할 수 없습니다. 즉, 둘 중 하나의 취약점을 가져야만 가능합니다.

UNION 키워드 공격은 개발자가 출력하고자 하는 값 대신에 공격자가 원하는 항목을 보이기 위한 방법입니다. 예를 들어 관리자는 사용자의 이름을 출력하고자 하였으나 공격자가 이를 변조하여 사용자의 비밀번호 정보를 출력하게끔 할 수 있습니다.

다음과 같은 SQL 쿼리문을 살펴봅시다.

```
SELECT name FROM users WHERE userid = '{$_GET[userid]}'
```

이 SQL 쿼리문은 사용자의 userid 값을 통해서 해당 사용자의 이름 정보를 가져오는데 이때 UNION 키워드를 이용하여 비밀번호 정보를 알아낼 수 있습니다.

```
SELECT name FROM users WHERE userid = 'brown'
UNION
SELECT userpw FROM users WHERE userid = 'brown'
```

이 쿼리문은 $_GET[userid] 값에 볼드 부분(첫 번째 줄의 b부터 세 번째 줄의 n까지. 첫 번째 줄의 첫 번째 따옴표와 세 번째 줄의 마지막 따옴표는 제외)을 넣어서 생성한 것으로 원래의 쿼리문은 이름에 해당하는 한 행의 결과만을 되돌려 주지만 UNION을 통해서 두 개의 쿼리에 대한 결과가 합쳐지기 때문에 다음과 같이 두 개의 행을 되돌려 줍니다.

```
첫 번째 행 : 조명진
두 번째 행 : brown1234
```

그러나 결과가 두 개의 행이 존재하더라도 하나의 행만을 보여주면 이 또한 의미가 없어집니다. 그래서 두 행의 결과를 보고 순서를 조정해주면 비밀번호 항목을 먼저 보이게 할 수 있습니다.

```
SELECT name FROM users WHERE userid = 'brown'
UNION
SELECT userpw FROM users WHERE userid = 'brown'
ORDER BY 1 -- '
```

이렇게 SQL 쿼리문을 변경하면 검색 항목이 정렬되기 때문에 비밀번호인 brown1234가 먼저 출력됩니다. 만약 이름이 abc와 같아서 비밀번호인 brown1234보다 정렬이 먼저 된다면 ORDER BY 1 DESC로 수정합니다. 그러면 비밀번호 항목을 볼 수 있습니다.

따라서 SQL Injection을 막기 위해서는 SQL 쿼리문을 만드는 데 사용되는 변수들이 쿼리문에 어떠한 영향을 줄 수 있는지를 잘 생각해서 숫자 형식의 값은 숫자만 입력할 수 있게 하고 문자 형식의 경우에는 작은따옴표처럼 SQL 쿼리에 영향을 줄 수 있는 문자가 들어가는지 확인해야 합니다.

Section 07

업로드 공격

파일 업로드 공격은 사용자가 웹 쉘처럼 공격이 가능한 웹 애플리케이션을 업로드해서 시스템을 2차적으로 공격하는 기법을 말합니다. 이 공격은 다른 공격과 달리 시스템을 공격하기 위한 발판이 되는 공격으로 다른 어떤 취약점보다도 위험하다고 할 수 있습니다.

기본적으로 업로드 공격을 하려면 해당 사이트에 업로드가 가능한 웹 애플리케이션이 존재해야 하며 이를 통해 업로드한 파일에 웹으로 접근할 수 있어야 합니다. 그리고 마지막으로 업로드된 파일이 서버 스크립트로 동작할 수 있어야 합니다. 이러한 세 가지 조건을 모두 만족해야 파일 업로드 공격이 가능한데 대부분의 경우 앞의 두 가지는 가능하지만 세 번째 항목에서 불가능한 경우가 많습니다.

개발자의 입장에서 반대로 생각해보면, 업로드 기능을 부득이하게 만들어야 하는 경우 웹을 통해서 접근하지 못하게 하는 방법을 택할 수 있습니다. 웹에서 접근할 수 있는 홈 디렉토리가 아닌 그 상위의 디렉토리에 업로드를 위한 디렉토리를 만들고, 거기에 파일을 저장하게 하면 웹을 통해서 직접 URL로 접근할 수 없게 됩니다. 그 대신 다운로드를 구현할 때 일반적인 링크 방식으로는 다운로드가 불가능합니다. 왜냐하면 웹에서 접근할 수 없는 디렉토리의 파일을 다운로드하는 것이기 때문입니다. 그래서 해당 디렉토리의 파일을 PHP를 통해서 읽고 저장하게끔 별도의 다운로드 프로그램을 작성해야 하는데 이 때문에 뒤에서 배울 다운로드 취약점 공격이 가능해질 수 있습니다. 반면, 이 방법을 버리고 웹으로 접근할 수 있는 디렉토리에 파일을 저장한다면 PHP 파일과 같은 서버 스크립트 파일을 업로드하여 실행하는 업로드 공격을 받을 수 있습니다. 이러한 이유로 일반적으로 웹에서 접근할 수 있는 디렉토리에 파일을 저장하려고 서버 스크립트 파일의 업로드를 원천적으로 차단하는 방법을 사용하고 있습니다. 업로드할 때 파일의 확장자를 판단하여 서버 스크립트 파일이면 업로드가 되지 않도록 막는 방법입니다. 그러나 확장자 검증 코드에 허점이 있거나 단순히 웹 페이지에서 자바스크립트로 확장자를 검증하는 경우 여전히 업로드 공격을 받을 수 있습니다.

자바스크립트 검증 회피

일반적으로 업로드를 구현하기 위해서 파일 확장자를 검증합니다. 그런데 간혹 자바스크립트를 이용하여 파일 확장자를 검사하고 PHP 코드로는 확장자 검증을 하지 않는 경우가 있습니다. 이 경우의 착각은 절대로 값이 스크립트를 통과하지 않고서는 전달될 수 없다고 생각하는 것에 있습니다. 그러나 웹 프록시를 이용하면 자바스크립트 검증 부분을 쉽게 통과할 수 있습니다. 웹 프록시의 종류는 여러 가지가 있지만 많이 사용하는 Paros를 사용하여 자바스크립트의 확장자 검증을 회피해

보도록 하겠습니다.

일단 자바스크립트의 파일 확장자 검증 부분을 살펴봅시다.

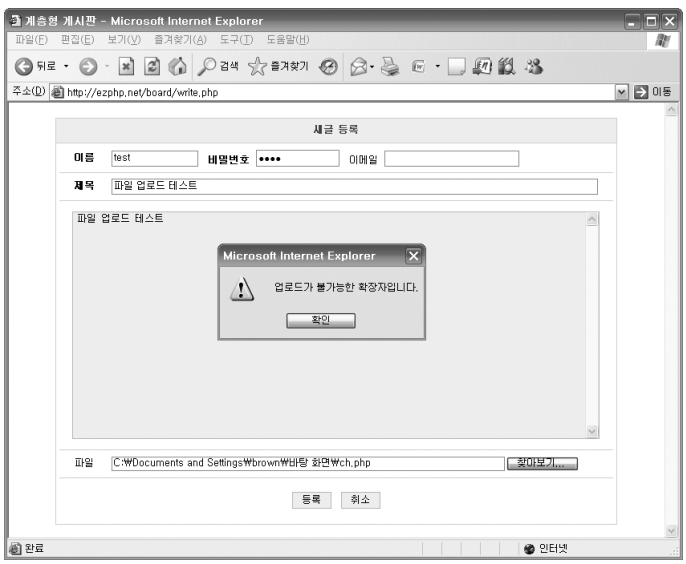
```
<script>

function CheckExt() {
var filename = fm.upfile.value;
if (filename.match(/\.(php|php3|html|htm|cgi|pl)$/i))
{
  alert('업로드가 불가능한 확장자입니다.');
  return false;
}
else return true;
}

</script>
```

자바스크립트의 match 함수를 사용하면 정규식을 이용하여 쉽고 빠르게 파일을 검증할 수 있습니다. 해당 정규식을 간단히 설명하면 다음과 같습니다.

```
/\.(php|php3|html|htm|cgi|pl)$/i
```

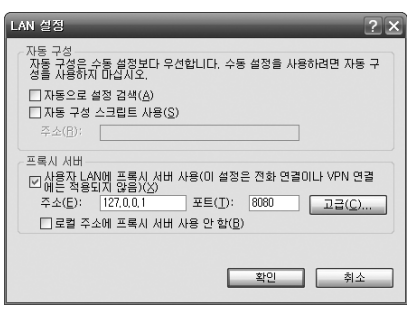
처음의 /는 정규식을 시작임을 나타내고 \.은 확장자 앞에 오는 점을 의미합니다. 정규식에서 점(.)은 임의의 한 문자를 나타내는 말입니다. 따라서 이처럼 순수하게 기호로 점의 의미를 가지려면 앞에 역슬래시를 붙여주어야 합니다. 그다음에 오는 괄호는 정규식 내의 패턴을 그룹화하기 위한 기호로 바(|)와 같이 사용하여 (A|B)는 A 또는 B를 의미합니다. 따라서 (php|php3|html|htm|cgi|pl)의 의미는 나열된 확장자 중에서 하나를 의미합니다. 그리고 달러($) 표시는 라인이나 문자열의 끝을 표시합니다. 즉, 달러 표시를 붙여주므로 해서 위의 확장자로 끝나는 파일 이름을 의미하게 됩니다. 마지막으로 정규식의 끝을 의미하는 슬래시(/)를 붙여주고 대소문자 구분을 하지 않기 위하여 마지막에 i를 붙여주면 정규식이 완성됩니다.

이러한 스크립트를 사용하여 파일을 업로드해보면 다음과 같이 경고 메시지가 출력되는 것을 확인할 수 있습니다.



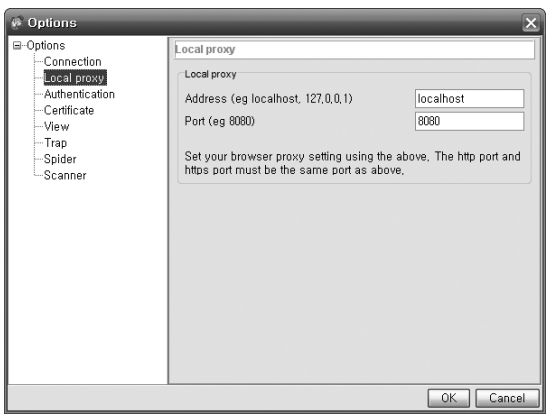
[그림 15-23] 자바스크립트를 이용한 확장자 검증

이제 자바스크립트 검증을 회피해보겠습니다. 우선 업로드하고자 하는 파일의 이름을 업로드가 가능한 확장자로 변경합니다. ch.php 파일의 이름을 ch.txt와 같이 업로드가 가능한 확장자로 변경한 후, 다음과 같이 인터넷 익스플로러가 웹 프록시를 사용할 수 있도록 설정을 변경합니다. 설정 방법은 인터넷 익스플로러의 메뉴 → 도구 → 인터넷옵션 → 연결 → LAN 설정입니다.



[그림 15-24] 웹 프록시 설정

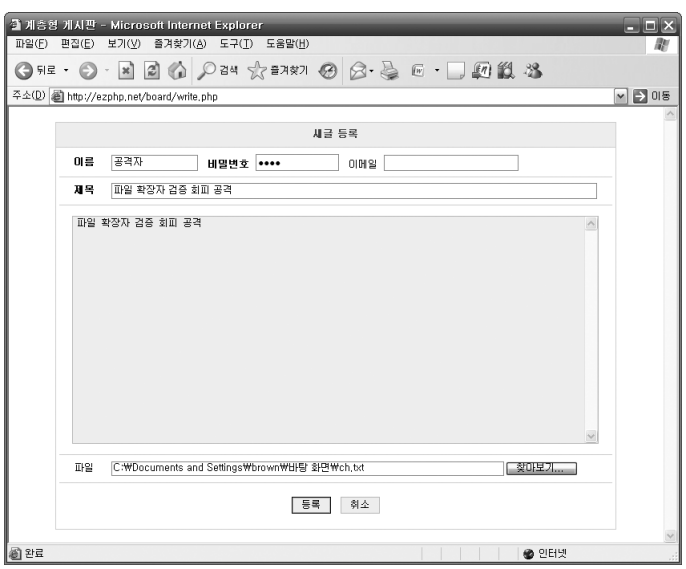
프록시의 포트는 설정에 따라 변경될 수 있습니다. Paros의 경우에 Tool->Options를 통해서 다음과 같이 설정할 수 있습니다.



[그림 15-25] Paros 웹 프록시 설정

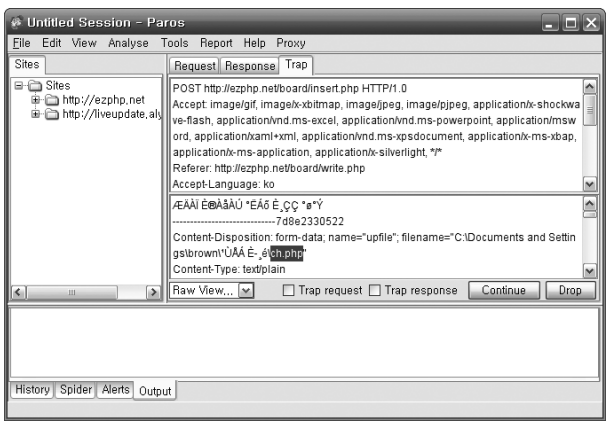
이렇게 웹 프록시를 등록하고 나면 인터넷 익스플로러를 실행하고 파일 업로드 페이지로 이동합니다. 그러면 Paros 웹 프록시의 왼쪽 Sites창에는 해당 사이트가 트리 형태로 보이는데 이때 오른쪽의 Trap창을 선택하여 아래와 같이 Trap request와 Trap response 항목을 체크합니다. 이는 HTTP를 통한 요청과 응답을 모두 감시하겠다는 의미입니다.

이제 모든 준비가 되었으니 파일 이름을 변경한 ch.txt 파일을 업로드합니다. 그러면 파일 확장자가 업로드 가능한 형태이기 때문에 자바스크립트의 검증을 무사히 통과합니다.



[그림 15-26] 파일 확장자 검증 회피 공격

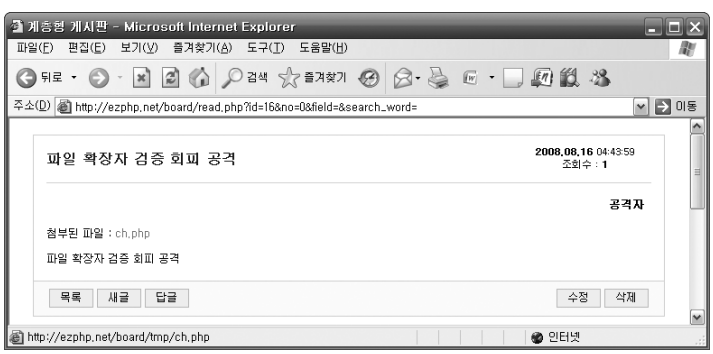
자바스크립트를 무사히 통과하고 나면 Paros 웹 프록시는 사용자가 입력한 폼 값을 가로채서 웹 서버에 전송하지 않고 우리의 응답을 기다립니다.



[그림 15-27] 웹 프록시를 이용한 파일 확장자 변경

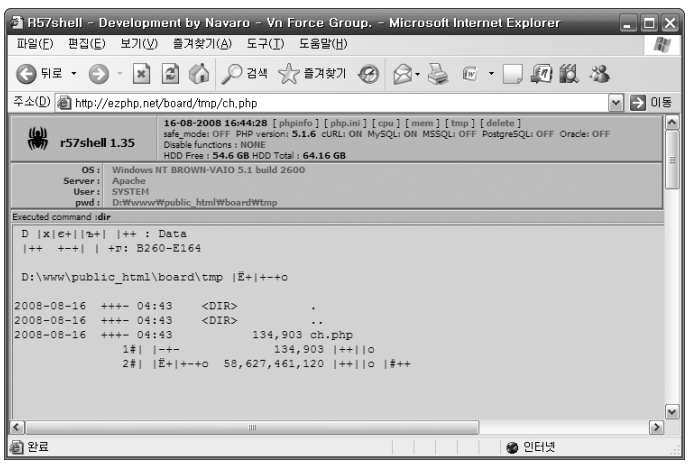
Trap창의 아랫 부분을 살펴보면 우리가 입력한 값들이 들어가 있는 것을 확인할 수 있습니다. 비록 Paros 웹 프록시가 한글을 완벽히 지원하지 않아서 한글이 위의 그림과 같이 깨져보일 수 있지만 사용에는 지장이 없으니 ch.txt 파일명을 찾아서 ch.php 파일로 변경합니다. 이제 가로챈 폼 입력 정보를 웹 서버로 전송하기 위해서 Continue 버튼을 누르면 됩니다. 그런데 더 이상 HTTP 프로토콜을 감시할 필요가 없으므로 Trap 항목은 모두 체크 해제를 하고 Continue 버튼을 누르도록 합시다.

이제 글은 등록되었고 글 읽기 페이지를 살펴보면 다음과 같이 파일이 업로드된 것을 알 수 있습니다.



[그림 15-28] 파일 이름이 변경된 글

첨부된 파일의 경로를 살펴보면 우리가 웹을 통해서 접근할 수 있는 주소임을 알 수 있습니다. 그래서 파일을 클릭해보면 다음과 같이 PHP 스크립트가 잘 실행되는 것을 확인할 수 있습니다.

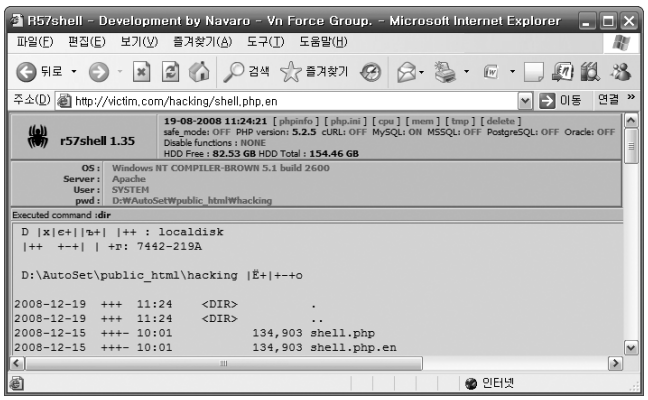


[그림 15-29] 웹 쉘을 실행

이처럼 클라이언트에서 동작하는 자바스크립트는 신뢰할 수 없습니다. 자바스크립트만으로 여러 가지 폼의 값을 검증하는 것은 매우 유용하며 또한 페이지의 변환이 없어서 빠른 결과를 보여주지만 언제나 보조 용도로 사용해야지 업로드 파일과 같은 중요한 값을 판단하는데 전적으로 신뢰하여 사용해서는 안 됩니다. 이러한 경우 반드시 PHP 스크립트로 다시 한번 검증을 해야만 안심할 수 있습니다.

아파치 다국어 지원을 이용한 확장자 검증 회피

아파치 웹 서버는 다국어를 지원하고 있습니다. 여러 가지 언어용 페이지를 만들어두면 웹 브라우저의 언어 설정에 따라서 해당 언어를 보여주는 기능입니다. 이는 HTTP/1.1 규약에 정의된 내용 협상(Content Negotitation) 기능의 한 형태입니다. 내용 협상은 브라우저가 제공하는 정보를 통해서 가장 적합한 자원을 찾아서 보여주는 기능입니다. 이를 지원하기 위해서 영문 페이지는 index.html.en, 프랑스어 페이지는 index.html.fr과 같은 형식으로 확장자의 추가 확장을 지원하고 있습니다. 그러나 이 기능으로 인해서 웹 해킹이 발생할 우려가 있습니다. 일반적으로 확장자를 검증할 때 있어서 가장 마지막에 오는 ".xxx" 형태를 확장자라고 간주하는 경우가 많습니다. 그러나 실제 확장자는 ".php.en" 중에서 ".php"이기 때문에 마지막의 ".xxx" 방식을 택했다면 확장자 검증 부분을 무사히 통과할 수 있습니다.



[그림 15-30] 다국어 지원으로 인한 확장자 검증 통과

많은 사람이 마지막 .en이라는 확장자로 인해서 PHP가 실행되지 않을 것이라고 생각하지만 실제로 위의 경우에서 볼 수 있듯이 PHP로 동작하는 것을 확인할 수 있습니다.

따라서 업로드를 구현하는 경우 마지막 확장자만을 판단할 것이 아니라 다음과 같은 방법으로 확장자가 될 수 있는 모든 값을 검증해야만 합니다.

```
<?
  $filename = 'index.php.en';
  $ext_exp = '\.(php|php3|html|htm|cgi|pl)';

  if (eregi($ext_exp, $filename))
  {
    echo "업로드가 불가능한 확장자입니다.";
  }
  else
  {
    echo "업로드가 가능한 확장자입니다.";
  }
?>
```

자바스크립트를 이용한 확장자 검증 부분에서와 마찬가지로 특정 확장자를 가지면 정규식을 통해서 걸리는 방법입니다. 그러나 자바스크립트 방법과의 차이는 .php나 .html과 같은 형식으로 파일의 이름이 끝나는 것이 아니라 이러한 문자열을 포함하는 파일 이름을 가진 경우에도 검증에서 걸리는 것입니다. 이 방법을 통해서 확장자의 뒤에 .en이나 .fr처럼 확장된 확장자가 붙어도 해당 확장자를 검증할 수 있습니다.

이와 더불어 앞서 배운 자바스크립트 검증 부분도 수정해야 합니다. 기존 자바스크립트 검증 부분은 .php로 끝나는 경우에 대해서 검증을 했기 때문에 .php를 포함하는 파일 이름 검증으로 수정하기 위해서 문자열의 끝을 의미하는 $ 기호를 제거하면 됩니다.

```
<script>

function CheckExt() {
var filename = fm.upfile.value;
if (filename.match(/\.(php|php3|html|htm|cgi|pl)/i))
{
  alert('업로드가 불가능한 확장자입니다.');
  return false;
}
else return true;
}

</script>
```

Section 08

다운로드 취약점 공격

확장자를 검사하여 업로드하게 하는 방법은 기본적으로 허용되는 파일만 업로드시키고 직접 URL로 접근하여 다운로드하게 하겠다는 의도가 깔려 있습니다. 보다 안전한 보안을 유지하기 위하여 특정 행위를 금지하여 제한하기보다는 허용을 통해서 다른 모든 것에 대해 제한하는 것이 바람직합니다. 즉, 특정 확장자를 가진 파일을 업로드 못하게 하는 것이 아니라 특정 확장자를 가진 것만 업로드하게 하는 것이 보다 안전한 것입니다. 그러나 이 방법에는 허용하는 확장자가 금지하는 확장자보다 훨씬 더 많기 때문에 금지를 통한 보안을 택하고 있습니다. 이러다 보니 서버의 설정에 따라서 PHP로 동작하는 확장자가 달라서 이를 확인하고 해당 확장자의 업로드를 금지시켜야 하지만 이를 간과하여 웹 해킹 취약점을 갖게 되는 경우가 많습니다.

그래서 보다 안전한 업로드를 위해서 URL을 통해 직접 접근할 수 없는 디렉토리에 파일을 저장하고 그것을 다운로드할 수 있는 다운로드 애플리케이션을 개발하는 경우가 많아졌습니다. 또한 여기에다 파일의 이름을 MD5 해시 문자열 등을 이용하여 변경하고 원래 파일의 이름을 데이터베이스 등에 기록해두는 방법으로 보다 안전한 업로드를 구현할 수 있습니다. 그런데 이렇게 URL을 통해서 접근할 수 없는 다른 디렉토리에도 접근할 수 있는 다운로드 애플리케이션은 또 다른 웹 해킹 공격을 받을 수 있습니다. 가장 대표적인 해킹 방법은 서버 시스템의 계정 정보 파일(/etc/passwd)을 취득하여 서버 시스템의 사용자 권한을 획득하는 것입니다.

```
http://victim.com/hacking/download.php?filename=abc.hwp
```

예를 들어 다운로드의 링크가 다음과 같다면 이 URL을 통해서 /etc/passwd 파일의 상대적인 위치를 대략적으로 알 수 있습니다. 일반적으로 리눅스 시스템에서 웹 서버를 구축하면 홈 디렉토리는 /var/www/html이 됩니다. 이를 바탕으로 hacking 디렉토리를 추가하면 대략 네 단계 정도 상위 디렉토리가 루트 디렉토리가 될 것이라고 예상할 수 있습니다.

```
http://victim.com/hacking/download.php?filename=../../../../etc/passwd
```

위의 URL과 같이 다운로드 파일을 상대적인 주소로 적어주면 다운로드 프로그램은 이에 따라 웹에서 접근할 수 없는 디렉토리의 파일을 다운받을 수 있게 해줍니다.

[그림 15-31] 다운로드 취약점을 이용한 passwd 파일 획득

이처럼 프로그램을 이용한 다운로드를 구현하면 예상치 못하게 시스템 파일을 다운로드할 수 있게 됩니다. 이 경우 해커는 /etc/passwd 파일을 통해서 사용자의 계정을 확인하고 사전식 무작위 대입 공격을 통해서 쉬운 형식의 패스워드를 사용하는 계정을 획득할 수 있습니다. 이 계정을 통해 SSH나 Telnet을 이용하여 접속하고 exploit 공격을 이용하여 root 권한을 얻게 됩니다.

이처럼 업로드만큼이나 파일 다운로드 기능도 매우 위험합니다. 따라서 파일 다운로드를 구현할 때는 반드시 상대 경로를 통해서 시스템 파일에 접근할 수 없게 만들어야 합니다.

```
if( eregi( "\.\.|/", $filename ) echo "상대 경로로 접근할 수 없습니다.";
```

Section 09

웹 해킹에 대한 보완 대책

웹 해킹은 프로그래머의 보안에 대한 인식결여 때문에 발생합니다. 대부분의 웹 해킹은 단순한 아이디어로 쉽게 취약점을 해결할 수 있습니다. 그럼에도 불구하고 많은 프로그래머가 짧은 개발기간을 핑계로 수많은 취약점을 무시하며 지내온 것이 사실입니다. 지금까지 운 좋게 해킹을 당하지 않고 잘 넘어왔다고 해서 앞으로도 영원히 그러리란 법은 없습니다. 프로그래머의 작은 실수가 우리의 사이트를 믿고 사용하는 수많은 사용자에게 흉기가 되어 되돌아갈 수 있다는 것을 명심하기 바랍니다. 더불어 수많은 웹 해킹 테크닉을 언제 배워 언제 보완하느냐고 생각하시는 분이 있다면 이것 하나만 기억하기 바랍니다.

"사용자로부터 어떤 값을 얻어와야 할 필요가 있다면,
그 입력이 내가 원하는 형식에 맞는지 반드시 검증하라."

숫자를 원한다면 사용자가 입력한 값이 숫자가 맞는지, 파일 이름을 원한다면 혹 해당 파일이 아닌 다른 파일을 임의로 접근할 수 있지는 않은지 모두 검증을 해야 합니다. 이것만 유념해서 프로그램을 작성한다면 대부분의 웹 해킹은 차단할 수 있을 것입니다.