1부

PHP 사용하기

1장 PHP 훑어보기

2장 데이터 저장하고 불러오기

3장 배열 사용하기

4장 문자열 연산과 정규 표현식

5장 코드 재활용과 함수 작성

6장 객체 지향 PHP

7장 예외 처리

1장 PHP 훑어보기

이 장에서는 PHP 문법과 PHP 구조에 대해 훑어보고자 한다. 만약에 PHP를 사용해본 적이 있다고 하더라도 그동안 모르고 지나쳤던 부분들을 다시 한 번 자세히 알아보자. C 언어와 ASP, 그 외 다른 프로그래밍 언어를 사용해보았다면 좀 더 빠르게 책 내용을 익힐 수 있을 것이다.

이 책에서는 전자상거래를 구축한 경험을 바탕으로 작성한 실제 예제를 통하여 PHP를 알아본다. 다른 프로그래밍 교재들이 기본적인 문법을 온라인 매뉴얼과 다를 바 없는 간단한 예제로 함수와 문법을 나열하는 방식으로 가르치는 데 비해, 우리는 구체적으로 뭔가를 만들어서 실행해봄으로써 언어가 어떻게 쓰이는지 알아보는 좀 더 합리적인 방식을 채택했다.

예제들을 입력하거나 이 책에 수록된 CD-ROM에서 불러와서, 바꿔보고 해석해보기도 하면서 이들을 고쳐 어떻게 활용할 수 있는지 알아보자.

이 장에서는 온라인 상품 주문 폼을 가지고 변수, 연산자, 표현식(expressions)이 PHP에서 어떻게 쓰이는지 알아보고, 데이터 변수형과 연산자 우선순위도 알아보자. 또한 폼 변수에 접근해서 고객 주문의 총합이나 세금을 계산하기 위해 폼 변수를 사용하는 방법을 알아볼 것이다.

그리고 PHP를 사용해서 입력 값을 확인하는 방법도 만들어보고, 불리언 값의 개념과 if, else, '?'와 같은 연산자와 switch 문을 사용하는 예제를 살펴보자.

마지막으로, 반복되는 HTML 테이블을 만들기 위해 PHP 반복문을 사용해보자.

다음은 이번 장의 주요 주제를 정리한 것이다.

- HTML에서 PHP 사용하기
- 폼 변수 다루기

- 식별자
- 사용자 정의 변수
- 변수에 값 대입하기
- 변수형
- 상수
- 변수의 범위
- 연산자와 우선순위
- 표현식
- 변수 함수
- if, else, switch와 같은 조건문
- while, do, for와 같은 반복문

PHP 사용하기

예제를 실행해보기 위해서는 PHP를 사용할 수 있는 웹 서버가 필요하다. 예제를 제대로 익히기 위해서는, 실행시키고 코드를 바꿔볼 실험 장소가 필요하다.

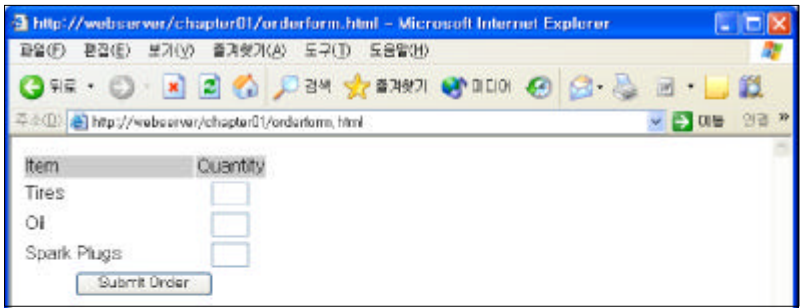
만약 컴퓨터에 PHP가 설치되어 있지 않다면 직접 설치하거나 관리자에게 설치해달라고 요청하자. PHP를 설치하는 방법은 부록 A "PHP, MySQL 설치 가이드"에 나와있다. 이 책에 수록된 CD-ROM에서 유닉스나 윈도우즈 NT에서 PHP를 설치할 때 필요한 모든 정보를 찾을 수 있다.

예제 : Bob's Auto Parts

HTML 폼을 처리하기 위해서 PHP와 같은 서버 스크립트 언어를 많이 사용한다. Bob's Auto Parts라는 부품 회사에서 사용할 주문 폼을 PHP로 만들어보자. 이 장에서 사용되는 Bob의 예제는 CD-ROM의 chapter01 디렉토리에 있다.

주문 폼

Bob이 파는 부품에 대한 주문 폼을 만들어 보았다([그림 1.1] 참조). 웹을 서핑하는 동안 흔히 볼 수 있는 간단한 주문 양식이다. 우선 Bob은 고객이 주문한 것이 무엇인지, 총액은 얼마인지 또 그 주문에 따른 세금은 얼마나 되는지 알고 싶어한다.



[그림 1.1] Bob의 주문 폼 초기 화면, 아직은 물품과 개수 칸만 있다.

이 폼을 생성하는 HTML의 일부가 [리스트 1.1]에 나와있다.

[리스트 1.1] orderform.html — 주문 폼을 만드는 HTML

```
<form action="processorder.php" method="post">
<table border="0">
<tr bgcolor="#cccccc">
  <td width="150">Item</td>
  <td width="15">Quantity</td>
</tr>
<tr>
  <td>Tires</td>
  <td align="center"><input type="text" name="tireqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td>Oil</td>
  <td align="center"><input type="text" name="oilqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td>Spark Plugs</td>
  <td align="center"><input type="text" name="sparkqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td colspan="2" align="center"><input type="submit" value="Submit Order"></td>
</tr>
</table>
</form>
```

첫번째로 살펴볼 부분은 폼의 action이다. 이 부분에 고객의 주문을 처리할 PHP 스크립트(앞으로 만들 것이다)의 이름을 적어 주었다. action의 값은 대부분 고객이 확인 버튼을 눌렀을 때 실행되는 페이지의 URL이다. 이 URL로 사용자가 폼에 입력한 정보를 보내는데, method 속성이 get이면 URL의 뒷부분에 붙여서 보내고 post이면 다른 패킷으로 보낸다.

두번째로 살펴볼 부분은 tireqty, oilqty, sparkqty라는 폼 필드(field)의 이름이다. PHP 스크립트에서도 같은 이름을 쓰기 때문에 폼 필드에 의미 있는 이름을 써야 기억하기 쉽다. HTML 편집기는 필드 이름을 의미 없는 field23과 같이 만들어주는데 폼에 들어올 데이터와 관련된 이름을 사용하는 편이 훨씬 좋다.

적절한 표준 방법을 채택해서 사이트 내의 모든 필드 이름을 같은 방법으로 짓는다면 기억하기 훨씬 편하다. 예를 들면, 단어를 요약해서 쓴다든가 공백 대신에 '_'을 넣는 등의 방법이 있다.

폼 처리하기

action 값에 들어간 processorder.php 스크립트를 만들어서 폼을 처리해보자. 텍스트 편집기를 열어서 다음과 같은 코드를 입력해보자.

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
</body>
</html>
```

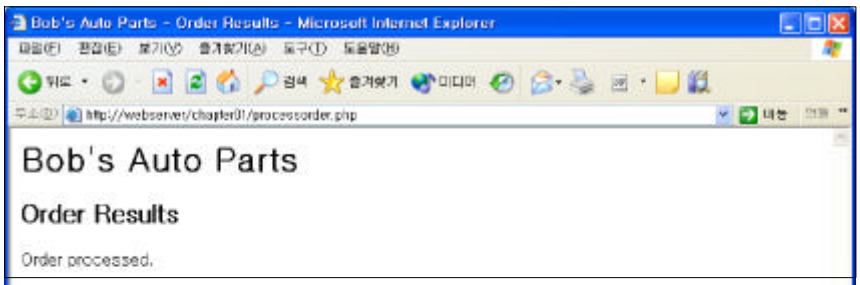
입력한 코드들은 아직까지는 단순한 HTML이다. 이제 그 안에 PHP 코드를 삽입해보자.

HTML에서 PHP 사용하기

<h2> 부분 아래에 다음 코드를 삽입하자.

```
<?php
  echo '<p>Order processed.</p>';
?>
```

파일을 저장하고 Bob의 폼을 모두 채운 뒤에 확인 버튼을 눌러 보자. [그림 1.2]와 비슷한 화면을 볼 수 있다.



[그림 1.2] PHP의 echo를 사용하면 브라우저에 글자가 출력된다.

방금 전에 삽입한 PHP 코드가 어떻게 됐는지 확인해보기 위해서 브라우저에서 소스 보기를 해보자. 그러면 다음과 같은 코드가 제시된다.

```
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
<p>Order processed.</p></body>
</html>
```

PHP 인터프리터가 스크립트를 읽어서 결과물로 바꿔놓기 때문에 실제 PHP 코드는 보이지 않는다. PHP가 브라우저가 이해할 수 있는 HTML 형식으로 바뀌기 때문에 사용자의 브라우저는 PHP를 이해할 필요가 없다.

이 예제는 서버측 스트립트를 잘 보여주는 예이다. PHP는 웹 서버에서 실행하는 서버 스크립트로, 웹 브라우저에서 실행하는 JavaScript나 다른 클라이언트 기술과는 다르다.

processorder.php의 네 가지 구성 요소를 살펴보자.

- HTML
- PHP 태그
- PHP 문
- 공백

여기에 하나 더 추가하자면,

■ 주석

대부분은 HTML이 차지하고 있다.

PHP 태그 사용하기

앞 예제에서 사용한 PHP 코드는 <?php로 시작해서 ?>로 끝났다. 다른 HTML 태그처럼 '<'로 시작해서 '>'로 끝난다. 이런 기호들(<?php와 ?>)을 PHP 태그라고 한다. 이런 기호들로 웹 서버는 PHP 시작과 끝을 인식할 수 있다. PHP 태그 사이에 있는 모든 문장은 PHP로 인식되고 PHP 태그 밖에 있는 것은 일반 HTML로 인식된다. 따라서 PHP 태그를 사용하여 HTML을 빠져나올 수 있다.

다양한 태그 스타일 중에서 선택하여 사용할 수 있다. 쓸 수 있는 여러 가지 태그를 살펴보자.

PHP 태그 스타일

사용할 수 있는 PHP 태그에는 총 네 가지 방식이 있다. 다음 예제 코드들은 모두 동일한 기능을 한다.

■ XML 스타일

```
<?php echo '<p>Order processed.</p>'; ?>
```

예제에서 사용한 태그 스타일로 가장 선호되는 방식이다. 서버가 이 방식을 지원하지 않도록 할 수 없기 때문에, 모든 서버에서 사용할 수 있다. 만약 여러 곳에서 사용할 응용 프로그램을 만들고 있다면 이 방식을 사용하는 편이 좋다. 이 스타일은 XML 문서에서도 사용할 수 있다. 사이트에서 XML을 지원할 생각이라면, 꼭 이 스타일의 태그를 사용해야 한다.

■ 짧은 스타일

```
<? echo '<p>Order processed.</p>'; ?>
```

SGML 방식을 따르는 가장 간단한 스타일이다. 이 스타일을 사용하려면 config 파일의 short_open_tags를 활성화시키거나, 짧은 스타일 태그를 사용할 수 있도록 PHP를 컴파일해

야 한다. 이것에 대한 정보는 부록 A에 자세히 나와있다. 이 방식은 대부분의 서버가 지원하고 있지만, XML 문서 선언에 영향을 주기 때문에 이 방식을 지원하지 않을 수도 있다.

■ 스크립트 스타일

```
<SCRIPT LANGUAGE='php'> echo '<p>Order processed.</p>' </SCRIPT>2
```

가장 긴 스타일로 JavaScript나 VBScript를 사용해봤다면 익숙할 것이다. HTML 편집기를 쓰면 다른 태그 스타일을 사용할 수 없을 경우가 있다. 그럴 때 이 방법을 사용하면 된다.

■ ASP 스타일

```
<% echo '<p>Order processed.</p>'; %>
```

ASP나 ASP.NET과 같은 스타일로 config 파일에서 asp_tags를 활성화해 놓았다면 사용할 수 있다. 편집기를 ASP나 ASP.NET 방식으로 사용하고 있다든가, 이미 ASP나 ASP.NET으로 만든 스크립트를 PHP에서 사용하고 싶을 때 가능하다. 그러나 기본적으로는 이 태그 방식은 활성화되어 있지 않다.

PHP 문

PHP 인터프리터는 PHP 태그 사이의 PHP 문을 가지고 동작한다. 앞 예제에서는 단지 다음의 한 문장 형식이 사용되었다.

```
echo '<p>Order processed.</p>';
```

예제에서 살펴보았듯이 echo는 문자열을 브라우저에서 출력한다. [그림 1.2]를 보면 브라우저에 Order processed.라는 문장이 나타난 것을 확인할 수 있다.

echo 문 마지막에 찍힌 세미콜론(;)은 마침표(.)가 문장의 끝을 나타내듯이 C 언어나 Java에서처럼 PHP 문의 끝을 나타낸다.

초보자들이 세미콜론을 빼먹는 실수를 자주 하긴 하지만 쉽게 발견해서 고칠 수 있다.

공백

줄바꿈 문자, 스페이스, 탭과 같은 공간 문자를 공백(whitespace)이라고 부른다. 브라우저가 HTML에 있는 공백을 무시하듯 PHP 처리기도 PHP 코드의 공백을 무시한다. 다음 두 개의 HTML 코드를 살펴보자.

```
<h1>Welcome to Bob's Auto Parts!</h1><p>What would you like to order today?</p>
```

와

```
<h1>Welcome              to Bob's
Auto Parts!</h1>
<p>What would you like
 to order today?</p>
```

위 두 HTML 코드는 브라우저에 동일한 결과를 나타낸다. 하지만 HTML, PHP 코드의 가독성을 높이기 위해서 공백을 적절히 사용하는 것이 좋다. PHP 문 사이에 공백이 있을 필요는 없지만 각기 다른 줄에 쓰는 편이 훨씬 읽기에 편하다. 예를 들어,

```
echo 'hello ';
echo 'world';
```

와

```
echo 'hello ';echo 'world';
```

위 두 경우는 동일하지만 처음 코드가 읽기 더 편하다.

주석

주석은 코드에 대한 설명을 코드 속에 적어놓은 것이다. 주석에는 스크립트의 목적, 누가 작성한 것인지, 왜 이런 방식으로 만들어졌는지, 언제 마지막으로 수정되었는지 등에 대한 정보가 포함된다. 아주 간단한 PHP 스크립트를 제외하고는 모두 주석이 있다.

PHP 인터프리터는 주석을 무시한다. 특히 PHP 파서(parser)는 주석을 공백처럼 처리한다.

PHP는 C 언어, C++ 및 쉘 스크립트 스타일의 주석을 모두 지원한다.

다음은 PHP 스크립트의 처음에 나올 만한 C 언어 스타일의 여러 줄 주석의 예이다.

```
/* 작성자: Bob Smith
   마지막 수정일: 4월 10일
   고객 주문을 처리한다.
*/
```

여러 줄 주석은 /*로 시작해서 */로 끝난다. C 언어와 달리 여러 줄 주석은 중첩될 수 없다.

한 줄 주석을 달고 싶으면 C++ 스타일로 써보자.

```
echo '<p>Order processed.</p>'; // 주문을 출력한다.
```

혹은 쉘 스크립트 스타일로도 쓸 수 있다.

```
echo '<p>Order processed.</p>'; # 주문을 출력한다.
```

둘 다 주석 기호(//, #) 다음부터 그 줄의 끝이나 PHP 태그까지 주석으로 처리된다.

다음 예에서 닫는 태그 전에 있는 문장인 here is a comment는 주석이다. 하지만 닫는 태그 다음에 있는 here is not은 닫는 태그 밖에 있기 때문에 HTML로 처리된다.

```
// here is a comment ?> here is not
```

동적으로 컨텐츠 추가하기

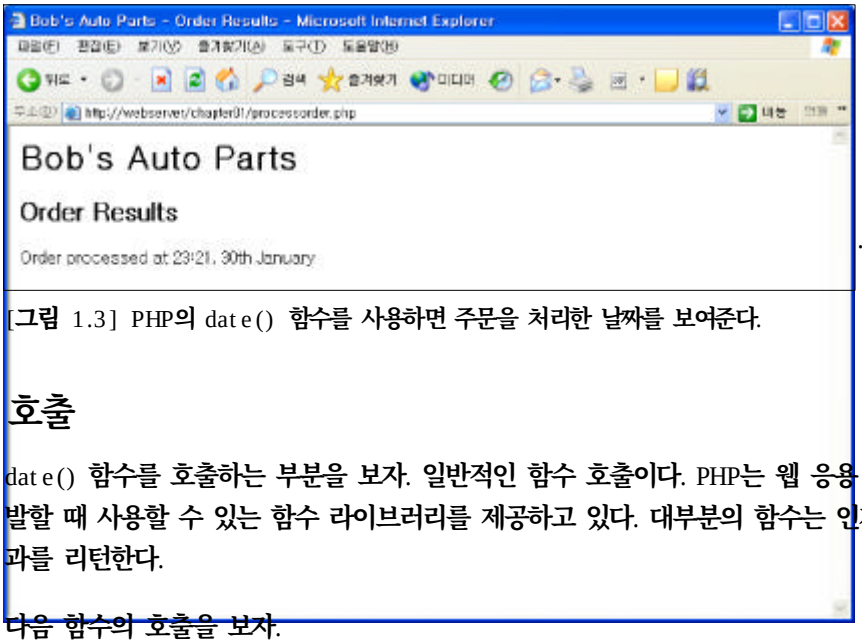
지금까지는 단순한 HTML도 할 수 있는 기능만을 다루었다.

서버 스크립트의 사용 목적은 사이트의 사용자에게 동적인 컨텐츠를 제공하기 위해서이다. 사용자의 요구나 시간에 따라 컨텐츠가 달라지게 하여 사용자들이 사이트를 다시 방문할 수 있도록 유도하는 스크립트를 만들어보자. PHP를 사용하면 쉽게 할 수 있다.

간단한 예제부터 시작해보자. processorder.php를 다음 코드로 바꾼다.

```
<?php
  echo '<p>Order processed at ';
  echo date('H:i, jS F');
  echo '</p>';
?>
```

이 코드는 PHP에서 제공하는 date() 함수를 사용하여 고객의 주문이 처리되고 있는 시간을 알려준다. 실행되는 시기에 따라 시간은 달라지겠지만 스크립트를 실행한 결과는 [그림 1.3]과 비슷할 것이다.



[그림 1.3] PHP의 date() 함수를 사용하면 주문을 처리한 날짜를 보여준다.

함수 호출

date() 함수를 호출하는 부분을 보자. 일반적인 함수 호출이다. PHP는 웹 응용 프로그램을 개발할 때 사용할 수 있는 함수 라이브러리를 제공하고 있다. 대부분의 함수는 인자를 받아서 결과를 리턴한다.

다음 함수의 호출을 보자.

```
date('H:i, jS F')
```

한 쌍의 괄호 안에 문자열을 넣어 함수로 전달할 수 있다. 이런 것을 함수의 인자 혹은 파라미터라고 부른다. 함수는 특정 결과값을 만들기 위해 인자를 사용한다.

date() 함수

date() 함수는 건네준 인자를 결과 형식을 나타내는 포맷 문자열로 사용한다. 문자열의 문자들은 날짜와 시간을 나타낸다. H는 24시간 형식의 "시간"을 나타내고 두 자리를 맞추기 위해 0을 앞에 붙이기도 한다. i는 두 자리로 나타낸 "분"을 가리킨다(01분 등으로 표시). j는 이번 달의 날짜를 나타내는데 0은 붙이지 않는다. S는 서수접미사를 나타내고(이 경우 "th"), F는 이번 달의 이름을 가리킨다(1월이라면 "Jan"으로 표시하지 않고 "January"로 표시한다).

date()가 지원하는 포맷을 모두 살펴보고 싶다면 20장 "날짜와 시간 다루기"를 보자.

폼 변수 다루기

주문 폼에서 필요한 것은 고객의 주문을 모아오는 일이다. 고객이 입력한 정보를 모으는 일은 PHP에서 아주 쉬운 일이기는 하지만 정확한 방법은 PHP 버전과 php.ini 파일에 설정한 것에 따라 다를 수 있다.

폼 변수

PHP 스크립트에서 폼 필드 값은 PHP 변수처럼 접근할 수 있다. PHP에서는 변수가 $ 부호로 시작하기 때문에 쉽게 알아볼 수 있다($ 기호를 빼먹지 말자!).

PHP 버전과 세팅에 따라서 폼 데이터를 변수로 접근하는 방법은 세 가지가 있다. 공식적인 이름이 없으므로 일단 짧은, 중간, 긴 스타일이라고 부르자. 각각의 폼 필드는 PHP 스트립트에서 입력되어야만 이번 스크립트에서 쓸 수 있다.

tireqty라는 필드는 다음과 같이 접근할 수 있다.

```
$tireqty                          // 짧은 스타일
$_POST['tireqty']                 // 중간 스타일
$HTTP_POST_VARS['tireqty']        // 긴 스타일
```

앞으로는 폼 변수에 접근하기 위해서 중간 스타일(즉 $_POST['tireqty'])을 사용한다. 하지만 필요한 경우 짧은 버전도 사용하였다(PHP 4.2.0버전 이후로는 중간 스타일이 추천되고 있다.)

PHP 코드를 작성할 때에는 다른 스타일을 사용할 수도 있으므로 여러 가지 방법을 살펴보자.

- 짧은 스타일($tireqty)은 편하지만 register_globals를 활성화시켜야 한다. 기본 설정값은 버전에 따라 다르다. 4.2.0 이후로는 비활성화되어 있다. 예전에는 기본적으로 활성화되어 있었기 때문에 많은 PHP 프로그래머들이 짧은 스타일을 사용하였다. 하지만 시간이 지날수록 많은 혼란을 발생시켰고, 짧은 스타일을 사용하면 코드가 안전하지 못할 수도 있었다. 그래서 더 이상은 짧은 스타일을 추천하지 않는다.
- 중간 스타일($_POST['tireqty'])이 추천되는 방식이다. 중간 스타일은 꽤 편하지만 PHP 4.1.0 이후에서만 사용할 수 있기 때문에 더 오래된 시스템에서는 쓸 수 없다.
- 긴 스타일($HTTP_POST_VARS['tireqty'])은 가장 많은 단어를 사용한다. 긴 스타일은 계속 지양되어 왔기 때문에 미래에는 사용되지 않을 것 같다. 예전에는 긴 스타일이 어디에서나 사용할 수 있어 이식성이 높았지만, 성능을 높이기 위해 register_long_arrays를 사용하여 비활성화시킬 수 있다.

짧은 스타일을 사용하면 스크립트에서 사용하는 변수의 이름이 폼 필드의 이름과 같다. 변수를 정의한다든가 변수를 생성할 필요는 없다. 그 변수들은 함수에 인자를 건네주듯이 스크립트로

넘겨진다. 짧은 스타일을 사용한다면 $tireqty와 같이 변수로 사용할 수 있다. 폼 필드 tireqty는 $tireqty 변수를 생성한다.

짧은 스타일을 사용하려면 register_globals를 설정해야 한다. register_globals를 활성화시키기 전에 왜 PHP 개발팀이 이 값을 기본적으로 비활성화해두었는지 생각해보는 쪽이 좋겠다.

변수처럼 바로 접근하는 방식은 매우 편하지만, 프로그램을 작성할 때 폼 변수와 자신이 만든 변수와 섞이는 일이 발생할 수도 있다. 짧은 스타일을 사용하게 되면 사용자들이 직접 입력한 믿을 수 없는 변수와 스크립트에서 만든 변수를 구별할 명시적인 방법이 없어진다.

모든 변수에 기본값을 설정하지 않는다면, 스크립트 사용자들이 변수를 전달하면서 스크립트의 변수와 폼 변수를 섞어버릴 수 있다. 짧은 스타일을 사용하려면 자신의 모든 변수를 초기화하는 것이 좋다.

중간 스타일은 $_POST, $_GET, $_REQUEST 중 한 배열에서 폼 변수를 접근할 수 있다. $_POST나 $_GET 둘 중 하나는 폼 변수 값을 저장하는데, 메소드를 POST로 했을 경우 $_POST에서 폼 변수 값을 구할 수 있고, GET으로 했다면 $_GET에서 얻을 수 있다. 그리고 POST로 보내든 GET으로 보내든 간에 모든 변수는 $_REQUEST에서도 접근할 수 있다.

POST 방식을 선택했을 때 tireqty에 적힌 데이터는 $_POST['tireqty']에 저장되어 있다. GET을 선택했다면 $_GET['tireqty']에 저장되어 있을 것이다. 두 경우 모두 $_REQUEST['tireqty']에는 값이 저장되어 있다.

이런 배열은 소위 수퍼글로벌이라고 불리는 것으로, 변수 범위에 대해 설명할 때 자세히 알아보자.

옛날 PHP 버전을 사용한다면 $_POST나 $_GET에 접근할 수 없다. 4.1.0 버전보다 예전 버전이라면 대신에 $HTTP_POST_VARS와 $HTTP_GET_VARS라는 배열에 폼 변수 값이 저장되어 있다. 이 변수를 사용하는 방식을 긴 스타일이라고 부른다. 앞에서 말했듯이 긴 스타일은 지양되어 왔다. 그리고 $_REQUEST와 같은 역할을 하는 배열도 없다.

긴 스타일을 사용한다면 $HTTP_POST_VARS['tireqty']나 $HTTP_GET_VARS['tireqty']로 폼 변수를 사용할 수 있다.

이 책의 예제들은 PHP 5.0으로 검사하였기 때문에 PHP 4.1.0보다 이전 버전일 경우 동작하지 않을 수도 있다. 가능하다면 최신 버전을 사용하길 바란다.

이제 다른 예제를 살펴보자. 길거나 중간 스타일의 변수 이름은 좀 성가시므로 배열이라는 변수 타입을 사용해보자. 배열은 3장 "배열 사용하기" 에서 자세히 설명하기 전까지는 제대로 설명하지 않겠지만 일단 사용하기 쉬운 복사본을 만들어 사용한다.

한 변수의 값을 다른 변수에 복사하기 위해서 대입 연산자('=')를 사용한다. 다음 코드는 $tireqty라는 새 변수를 생성하여서 $POST['tireqty']의 값을 새 변수에 복사한다.

```
$tireqty = $_POST['tireqty'];
```

처리 스크립트의 처음에 다음 코드들을 삽입해보자. 폼에서 오는 데이터를 처리하는 스크립트들은 모두 첫 부분에 비슷한 코드들을 가지고 있다. 이 코드들은 결과가 출력되지 않기 때문에 HTML 태그 앞에 넣거나 뒤에 넣어도 다를 것이 없다. 하지만 찾기 쉽도록 스크립트의 첫 부분에 넣는 것이 좋다.

```
<?php
   // 짧은 스타일의 변수로 만든다.
  $tireqty = $HTTP_POST_VARS['tireqty'];
  $oilqty = $HTTP_POST_VARS['oilqty'];
  $sparkqty = $HTTP_POST_VARS['sparkqty'];
?>
```

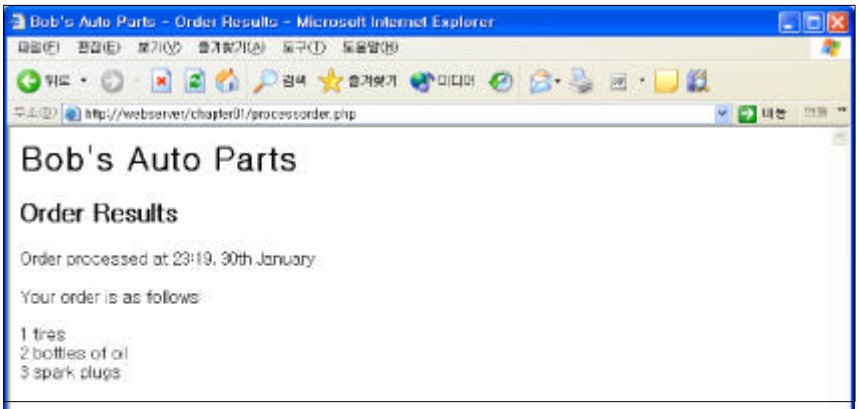
위 코드는 $tireqty, $oilqty, $sparkqty라는 새 변수를 만들어서 POST 방식으로 넘어온 데이터를 저장한다.

스크립트의 기능을 알아보기 위해서 다음과 같은 코드를 추가해보자.

```
echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br />';
echo $oilqty.' bottles of oil<br />';
echo $sparkqty.' spark plugs<br />';
```

이때에 각 폼 필드에서 받아들인 데이터를 검사하지 않았다. 잘못된 데이터를 넣고 어떤 일이 일어나는지 보자. 이 장을 다 읽고 나면 이 스크립트에 데이터 검증 부분을 추가할 수 있을 것이다.

이 파일을 브라우저에 띄우면 [그림 1.4]와 비슷한 화면이 뜰 것이다. 실제 값은 당연히 폼에 입력한 값에 달려 있다.



[그림 1.4] 사용자가 입력한 폼 변수는 processorder.php에서 쉽게 사용할 수 있다.

다음에서 이 예제에서 사용된 재미있는 요소 몇 가지를 알아보자.

문자열 연결

사용자가 입력한 폼 필드의 값 뒤에 설명을 위해 넣은 문장이 화면에 출력되었다. echo 문을 자세히 보면 변수 이름과 문자열 사이에 "."가 찍혀 있는 것을 볼 수 있다.

```
echo $tireqty.' tires<br />';
```

'.'는 문자열 연결 연산자(string concatenation operator)라는 것으로 문자열들을 연결하는 데 쓰인다. echo를 사용해서 출력할 때 자주 사용되는 연산자이다. 이렇게 하면 echo를 여러 번 사용하지 않아도 된다.

배열이 아닌 변수는 겹따옴표 안에 넣어서 출력할 수도 있다(배열 출력은 좀 더 복잡하다. 배열과 문자열을 같이 사용하는 방법은 4장 "문자열 연산과 정규 표현식"에서 알아본다). 예를 들면 다음과 같다.

```
echo "$tireqty tires<br />";
```

이와 같이 해도 첫번째 문장과 동일하다. 둘 중 마음에 드는 것을 사용하면 된다. 이처럼 변수가 변수의 내용물로 치환되는 것을 삽입이라고 한다.

하지만 이것은 겹따옴표(")의 특징으로, 홑따옴표(')는 이렇게 사용할 수 없다. 다음 코드를 실행해보자.

```
echo '$tireqty tires<br />';
```

그러면 화면에 "$tireqty tires
"가 찍히는 것을 볼 수 있다. 겹따옴표를 쓰면 변수 이름이 그 값으로 변환되지만 홑따옴표 안에서는 아무 것도 바뀌지 않은 채 출력된다.

변수와 문자

echo 문에서 연결한 변수와 문자열은 다른 데이터 형을 가지고 있다. 변수는 데이터의 기호이고 문자열은 데이터 자체이다. 실제 데이터는 문자라고 한다. $tireqty는 변수이고 고객이 입력한 값을 저장하고 있지만, tires
는 문자이다. 문자는 보이는 대로의 값을 가진다. 앞에 나왔던 두번째 예제를 기억해보자. PHP는 문자열에 쓰이는 $tireqty를 그에 저장된 값으로 치환한다.

최근에는 문자열을 표시하는 세번째 방법이 추가되었다. Perl에서 사용되던 heredoc 문법(<<<)이 PHP4에 추가되었다. heredoc 문법은 긴 문자열을 깔끔하게 명시할 수 있도록 문자열을 끝나는 끝마무리 문자를 사용자가 명시할 수 있게 한다. 다음 예제는 세 줄짜리 문자열을 생성하여 출력한다.

```
echo <<<theEnd
line 1
line 2
line 3
theEnd
```

theEnd는 무엇으로 바꾸어도 상관없다. 그저 문장 안에 나오지만 않으면 된다. heredoc 문자열을 마치기 위해서는 처음에 썼던 끝마무리 문자를 써두면 된다.

heredoc 문자열도 겹따옴표처럼 삽입 방식을 취한다.

식별자

식별자(identifier)는 변수의 이름을 말한다(함수와 클래스의 이름도 역시 식별자이다. 5장 "코드의 재사용과 함수 작성"에서 함수와 클래스에 대해 알아볼 것이다). 식별자에 관한 몇 가지 간단한 규칙을 알아보자.

- 식별자는 어떤 길이도 가질 수 있으며 문자, 숫자, '_', '$로 만들 수 있다. 하지만 $를 식별자에 사용할 때에는 주의해야 한다. 그 이유는 다음에 "가변 변수" 부분에 설명되어 있다.

- 식별자는 숫자로 시작할 수 없다.
- PHP에서 식별자는 대소문자를 구별한다. $tireqty는 $Tireqty와 같지 않다. 둘을 섞어 쓰는 프로그래밍 오류를 자주 한다. 그러나 함수의 이름은 대소문자를 구별하지 않는다.
- 변수는 함수와 같은 이름을 가질 수 있지만 피하는 것이 좋으며, 다른 함수와 같은 이름을 가지는 함수는 만들 수 없다.

사용자 정의 변수

HTML 폼에서 보낸 변수 이외의 사용자 자신의 변수를 정의해서 사용할 수 있다.

변수는 처음 그 변수에 값을 대입하는 순간에 생기기 때문에 변수를 사용하기 전에 정의하지 않아도 된다.

변수에 값 대입하기

변수에 값을 대입할 때에는 한 변수의 값을 다른 변수에 복사할 때처럼 대입 연산자인 "="를 사용한다. 한 변수의 값을 다른 변수에 복사할 때도 사용할 수 있다. 예를 들어, Bob의 사이트에서 주문한 목록의 수와 총액을 계산해야 한다고 하자. 이 값을 저장하기 위한 두 변수를 만들기 전에 먼저 변수들을 0으로 초기화시켜 보자.

PHP 스크립트 아래에 다음 코드를 추가해보자.

```
$totalqty = 0;
$totalamount = 0.00;
```

위 두 줄은 변수를 생성하고 값을 대입한다. 그리고 변수에 변수를 대입할 수도 있다.

```
$totalqty = 0;
$totalamount = $totalqty;
```

변수형

변수의 형(type)은 저장된 데이터의 종류에 따라 정해진다. PHP가 정의하는 데이터 형은 점점 많아지고 있다. 종류가 다른 데이터는 다른 데이터 형으로 저장된다.

PHP의 데이터 형

PHP는 다음 기본 데이터 형을 지원한다.

- **정수형** — 모든 숫자에 사용된다.
- **실수형**(double이라고도 불린다.) — 실수를 나타낼 때 사용한다.
- **문자열** — 문자의 나열을 나타낸다.
- **불리언** — 참, 거짓의 값을 가진다.
- **배열** — 같은 형의 여러 데이터를 저장한다.
- **객체** — 클래스의 인스턴스를 저장한다.

특별한 NULL과 resource라는 두 가지 타입이 더 정의되어 있다. 값을 가지고 있지 않거나 특별한 값인 NULL을 가지는 변수는 NULL 형이다. 특정 함수(데이터베이스 함수 등)에서는 resource 형의 변수를 리턴하기도 한다. 이들은 외부 자원을 나타낸다(데이터베이스 연결 등). 하지만 resource 변수를 직접 다룰 기회는 흔치 않다. 그 대신 resource 변수를 리턴받아 다른 함수의 파라미터로 넘기게 된다.

형 강도

PHP는 형 강도(type strength)가 매우 약하다. 대부분의 프로그래밍 언어에서 변수는 미리 어떤 형으로 선언하고 한 가지 형의 데이터만을 저장할 수 있지만 PHP에서는 변수에 저장된 값에 따라 형이 결정된다.

$total_qty와 $total_amount를 만들어서 초기값을 넣어보자.

```
$total_qty = 0;
$total_amount = 0.00;
```

$total_qty에는 정수인 0을 대입했기 때문에 $total_qty는 정수형이 된다. 같은 방식으로 $total_amount는 실수형이 된다.

좀 이상해 보이겠지만 다음 코드를 추가해보자.

```
$total_amount = 'Hello';
```

추가한 후에는 $total_amount는 문자열이 된다. PHP는 변수에 저장된 값이 바뀔 때마다 그 데이터 형으로 변수의 형도 같이 바뀐다.

형이 자유자재로 바뀌는 것은 매우 유용하게 쓰인다. PHP에서는 변수에 데이터를 넣을 때마다 변수의 형이 그에 맞게 변하기 때문에 변수를 불러왔을 때 저장한 값 그대로 찾을 수 있다.

형 변환

형 변환(type casting)을 사용하면 데이터를 다른 형의 데이터로 전환할 수 있다. C 언어의 형 변환처럼 변수 앞에 괄호를 넣고 그 안에 바꾸고 싶은 형을 적으면 된다.

두 변수를 형 변환을 사용해서 정의해보자.

```
$totalqty = 0;
$totalamount = (double) $totalqty;
```

두번째 줄은 "$totalqty의 값을 가져와서 실수로 변환한 다음에 $totalamount에 저장하라"라는 뜻이다. $totalamount 변수는 실수형이 되지만 $totalqty는 여전히 정수형이다.

가변 변수(variable variables)

PHP는 가변 변수라는 변수를 지원한다. 가변 변수를 사용하면 변수의 이름을 동적으로 바꿀 수 있다.

PHP는 이런 영역에서 매우 자유롭다. 대부분의 언어에서 변수의 값을 바꿀 수 있을지는 몰라도 변수의 형을 바꾸기는 힘들다. 더군다나 변수의 이름을 바꾸기는 더욱 어렵다.

한 변수의 이름을 다른 변수가 값으로 가지는데, 다음 예를 살펴보자.

```
$varname = 'tireqty';
```

$tireqty 대신에 이제 $$varname을 써도 된다. $tireqty의 값도 바꾸어보자.

```
$$varname = 5;
```

위와 같이 하는 것은 다음과 동일하다.

```
$tireqty = 5;
```

불명확해 보일지 모르겠지만 일단은 그냥 넘어가자. 리스트를 가지고 폼 변수를 따로따로 쓰는 것보다 가변 변수를 사용하여 재귀 순환 방식으로 처리하는 것이 더 좋은 방법이라는 것만 알아두자.

상수

앞에서 언급했듯이 변수 안의 값은 바뀔 수 있다. 변수처럼 값을 저장할 수는 있지만 한 번 값이 결정되면 스크립트 내에서 그 값을 바꿀 수 없는 것을 상수라고 한다.

예제에서처럼 각 아이템의 가격을 상수로 저장할 수 있다. define 함수를 사용하여 상수를 정의한다.

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

위 코드를 스크립트에 추가해보자. 이제 고객 주문의 총합을 계산하기 위한 상수를 모두 갖추었다.

C 언어의 관례를 따라 상수의 이름을 모두 대문자로 만들면 상수와 변수를 쉽게 구별할 수 있다(물론 C나 PHP에서 꼭 그래야 하는 것은 아니다). 이런 관례를 지키면 코드의 유지보수에 편리하다.

변수는 $를 붙이지만 상수는 상수의 이름만 적어주면 된다. 한 번 상수를 사용해보자.

```
echo TIREPRICE;
```

정의한 대로 값이 화면에 출력된다. phpinfo() 함수를 호출하면 기본 상수에 관한 간략한 설명을 볼 수 있다.

```
phpinfo();
```

phpinfo()는 PHP에 정의되어 있는 변수와 상수에 대한 유용한 정보를 제공해준다. 앞으로 그들 중 일부를 알아볼 것이다.

변수와 상수의 중요한 차이점 중 하나는 상수는 불리언, 정수형, 실수형과 문자열 데이터만을 저장할 수 있다는 점이다. 이들은 스칼라 값이라고 불리는 데이터 형이다.

변수의 범위

변수의 **범위**(scope)란 스크립트 안에서 특정 변수를 사용할 수 있는 구간을 말한다.

PHP에는 기본적으로 여섯 가지 범위 규칙이 있다.

- 수퍼글로벌 변수는 스크립트 전역에서 사용할 수 있다.
- 한번 선언된 상수는 스크립트 전역에서 사용할 수 있다. 즉 함수의 안과 밖 모두에서 사용할 수 있다.
- 전역 변수는 스크립트 내에서 정의된 변수로 스크립트 내에서 사용할 수 있지만 함수 안에서는 사용할 수 없다.
- 함수 안에서 정의된 변수는 함수 내에서만 사용할 수 있다.
- 함수 안에서 전역으로 정의된 변수는 함수 밖에서는 사용할 수 없지만 매 사용 시마다 값이 저장되어 다음에 사용할 수 있다(5장에서 이 개념을 좀 더 자세히 설명한다).
- 함수 안에서 사용된 변수는 함수가 끝나면 삭제된다.

PHP 4.1에서 $_GET과 $_POST와 그 외 특별한 변수의 범위는 수퍼글로벌이며 함수 안과 밖 모두에서 사용할 수 있다.

수퍼글로벌의 전체 리스트는 다음과 같다.

- $GLOBALS — 모든 전역 변수의 배열(global 키워드처럼 함수 안에서 글로벌 변수에 접근할 수 있다. $GLOBALS['myvariable']처럼 접근할 수 있다.)
- $_SERVER — 서버 환경 변수의 배열
- $_GET — GET 메소드로 넘어온 변수의 배열
- $_POST — POST 메소드로 넘어온 변수의 배열
- $_COOKIES — 쿠키 변수의 배열
- $_FILES — 파일 업로드와 관련된 변수의 배열
- $_ENV — 환경 변수의 배열
- $_REQUEST — 사용자가 입력한 변수의 배열
- $_SESSION — 세션 변수의 배열

위 변수들에 대해서는 이 책을 진행하면서 적절한 상황에서 언급하기로 한다.

변수의 범위에 대해서는 함수를 알아볼 때 좀 더 자세히 살펴보자. 지금 스크립트에서 사용하는 변수는 전역 변수이다.

연산자

연산자(operator)는 값과 변수를 사용해서 어떤 일을 하는 기호를 말한다. 연산자를 사용해서 고객 주문의 총합과 세금을 계산해보자.

대입 연산자인 '='와 문자열 연결 연산자인 '.'는 이미 앞에서 언급했지만, 이번에는 연산자 전체를 살펴보자.

연산자는 한 개나 두 개 혹은 세 개의 인자를 가지고 그 중 대부분은 두 개의 인자를 가진다. 예를 들어, 대입 연산자는 저장될 곳은 '=' 왼쪽에, 저장할 값은 오른쪽에 두어 모두 두 개의 인자를 가진다. 이런 인자를 **피연산자**(operand)라 한다.

산술 연산자

산술 연산자(arithmetic operator)는 수학적 연산자와 비슷하게 생겼다. PHP의 산술 연산자를 [표 1.1]에 정리하였다.

[표 1.1] PHP 산술 연산자

연산자	이름	예
+	덧셈	$a + $b
-	뺄셈	$a - $b
*	곱셈	$a * $b
/	나눗셈	$a / $b
%	모듈러(나머지)	$a % $b

연산 결과는 대입 연산자를 사용해서 저장할 수 있다.

```
$result = $a + $b;
```

덧셈과 뺄셈은 $a와 $b 변수에 저장된 값을 더하기/빼기한다.

뺄셈 기호인 '-'는 음수 값을 가리키는 단항 연산자(피연산자를 하나 가지는 연산자를 말한다)로도 쓰인다. 다음과 같이 사용할 수 있다.

```
$a = -1;
```

곱셈과 나눗셈 역시 생각하는 대로 동작하는데, 수학적인 기호와 달리 '*'가 곱셈 연산자이고 '/'가 나눗셈 연산자라는 것을 기억해두자.

모듈러 연산자는 $a를 $b로 나눈 나머지를 리턴한다. 예를 들면 다음과 같다.

```
$a = 27;
$b = 10;
$result = $a%$b;
```

$result에 저장되는 값은 27을 10으로 나눈 나머지로 7이 된다.

산술 연산자는 정수와 실수형에서만 쓰일 수 있으며 문자열을 산술 연산하려고 시도하면 문자열을 숫자로 바꾸어서 실행한다. 만약에 문자열에 "e"나 "E"가 들어있다면 실수형으로 변환하고 그 외에는 정수형으로 변환한다. PHP는 문자열 처음에서 숫자를 찾아서 그것을 값으로 인식하는데, 만약에 문자열에 숫자가 없다면 문자열의 값은 0이 된다.

문자열 연산자

문자열 연산자는 앞에서 살펴본 문자열 연결 연산자 "."밖에 없다. 문자열 연결 연산자는 두 문자열을 앞뒤로 붙여서 새로운 문자열을 만들어낸다.

```
$a = "Bob's ";
$b = 'Auto Parts';
$result = $a.$b;
```

$result에 저장된 값은 "Bob's Auto Parts"이다.

대입 연산자

대입 연산자는 이미 언급한 "="이고, 이 연산자는 "~에 -를 대입한다"라고 읽는다.

```
$totalqty = 0;
```

이와 같은 경우 "$totalqty에 0을 대입한다"라고 읽으면 된다. 뒤에서 비교 연산자를 다룰 때 왜 이렇게 읽어야 하는지 설명할 것이다. 이 연산자를 "같다"라고 읽으면 비교 연산자와 헷갈리게 된다.

대입 연산자의 리턴값

대입 연산자도 다른 연산자들처럼 값을 리턴한다.

```
$a + $b
```

위와 같은 표현식의 값은 $a와 $b의 합이 된다.

마찬가지로 다음 표현식의 값은 0이 된다.

```
$a = 0
```

대입 연산자도 값을 리턴하기 때문에 다음과 같은 것도 가능하다.

```
$b = 6 + ($a = 5);
```

$b의 값은 11이 되며, 대입 연산자 왼쪽에 대입되는 값이 대입 연산문이 리턴하는 값이 된다.

괄호 안의 표현식은 더 높은 우선순위를 가지기 때문에 표현식의 값을 사용할 수 있으며 괄호는 수학에서와 같은 방식으로 쓰인다.

복합 대입 연산자

식을 간단하게 만들기 위해서 대입 연산자와 그 외 연산자를 결합해서 쓰는 경우도 있다. 변수에다 연산을 취한 다음 그 값을 같은 변수에 대입하려 할 때 사용한다.

```
$a += 5 ;
$a = $a + 5 ;
```

위 두 문장은 같은 식이다. 복합 대입 연산자는 산술 연산자와 문자열 연결 연산자와 함께 쓸 수 있다.

복합 대입 연산자의 목록이 [표 1.2]에 나와있다.

[표 1.2] PHP 복합 대입 연산자

연산자	사용법	동일식
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= $b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

전 ·후 증가와 감소 연산자

전 · 후 증가(++)와 감소(--) 연산자는 "+="와 "-=" 연산자와 비슷하지만 조금 다르다.

모든 증가 연산자는 두 가지 작업을 한다. 일단 값을 증가시킨 뒤, 다시 대입한다.

```
$a = 4;
echo ++$a;
```

두번째 것은 변수 앞에 "++"를 쓰는 전 증가 연산자를 사용하였다. 그 결과 $a는 1만큼 증가하여 값이 5가 되어 5를 출력하게 된다(실제로 $a에 저장된 값이 달라진다. $a +1을 반환하는 것과는 다르다는 것에 주의하자).

하지만 $a 뒤에 "++"를 쓰게 되면(후 증가 연산자) 결과가 조금 달라진다.

```
$a = 4;
echo $a++;
```

이번 경우 일단 $a의 값을 출력한 후에 그 값을 증가시키게 된다. 출력되는 값은 4이겠지만 그 다음 줄부터 $a의 값은 5가 된다.

"--" 연산자도 "++" 연산자와 비슷한 역할을 하지만 "--"는 값을 1만큼 감소시킨다.

참조 연산자

대입 연산자와 결합해서 쓸 수 있는 참조 연산자인 '&'에 대해 알아보자. 한 변수 값을 다른 변수에 대입하면 첫번째 변수의 값을 복사하여 메모리 어딘가에 저장한다.

```
$a = 5;
$b = $a;
```

이렇게 하면 $a의 값을 복사하여 $b에 저장하게 된다. 만약 $a의 값이 변경되어도 $b의 값에는 전혀 영향을 미치지 않는다.

```
$a = 7; // $b는 여전히 5이다.
```

이때 '&' 연산자를 사용하면 좀 달라진다.

```
$a = 5;
$b = &$a;
$a = 7; // $a와 $b 모두 7이다.
```

참조는 교묘하게 사용할 수 있다. 참조는 포인터라기보다는 별명에 가깝다는 것을 기억하자. $a와 $b는 같은 메모리 위치를 가리키고 있다. unset()을 사용하여 둘의 관계를 떼어놓을 수 있다.

```
unset($a);
```

이와 같은 연산은 $b의 값(7)을 바꾸지는 못하지만 $a와 메모리에 저장되어 있는 7이라는 값 사이의 연결을 제거한다.

비교 연산자

비교 연산자는 두 값을 비교한다. 비교 연산자는 두 값을 비교해서 true나 false의 값을 리턴한다.

등위 연산자

등위 연산자 "=="는 두 값이 같은지 비교할 때 사용한다.

```
$a == $b
```

이 식은 $a와 $b의 값이 같은지 비교한다. 만약 같다면 true를, 다르다면 false를 리턴한다.

등위 연산자는 대입 연산자인 "="와 혼동하기 쉬운데다 바꿔 써도 대부분 0이 아닌 값은 true로, 0이면 false로 인식한다. 따라서 다음과 같이 한 후 $a = $b로 연산을 하면 true가 리턴된다.

```
$a = 5;
$b = 7;
```

왜냐하면 $a = $b의 리턴값은 대입 연산자에서 설명했듯이 왼쪽에 대입되는 값이 7로서 0이 아니기 때문에 true로 인식한다. 만약 $a == $b였다면 false가 리턴되었을 것이다. 이러한 에러는 문법상으로는 잘못이 없기 때문에 매우 고치기 힘들다. 따라서 대입 연산자와 등위 연산자를 사용할 때에는 항상 주의를 기울여야 한다.

기타 비교 연산자

PHP에서 사용할 수 있는 등위 연산자 이외에 여러 가지 비교 연산자를 살펴보자. [표 1.3]에 사용 가능한 비교 연산자가 모두 나와있다. identical 연산자('===')는 두 피연산자의 값이 같고 같은 형을 가질 때에만 true를 리턴한다. 예를 들어 0=='0'은 true이지만, 0==='0'은 false이다. 왜냐면 앞의 0은 정수이나 뒤의 '0'은 문자열이기 때문이다.

[표 1.3] PHP 비교 연산자

연산자	이름	사용법
==	등위	$a == $b
===	동일	$a === $b
!=	같지 않다	$a != $b
!==	동일하지 않다	$a !== $b
<>	같지 않다	$a <> $b
<	작다	$a < $b
>	크다	$a > $b
<=	같거나 작다	$a <= $b
>=	같거나 크다	$a >= $b

논리 연산자

논리 연산자들은 논리적 조건들을 연결하는 데 쓰인다. 예를 들어, 만약 $a의 값이 0과 100 사이에 있는지 알아보고자 한다. 이럴 때에는 $a >= 0과 $a <= 100을 AND 연산자로 연결해서 사용할 수 있다.

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

PHP에서 사용할 수 있는 논리 연산자에는 AND, OR, XOR(exclusive OR), NOT이 있다.

논리 연산자와 사용법이 [표 1.4]에 정리되어 있다.

[표 1.4] PHP 논리 연산자

연산자	이름	사용법	결과
!	NOT	!$b	$b가 false이면 true를 리턴한다.
&&	AND	$a && $b	$a와 $b 모두 true여야 true를 리턴한다.
||	OR	$a || $b	$a와 $b 중 하나라도 true이면 true를 리턴한다.
and	AND	$a and $b	&&와 같다.
or	OR	$a or $b	||와 같다.

and와 or 연산자는 "&&"와 "||" 연산자보다 우선순위가 낮다. 우선순위에 대해서는 뒤에서 알아보기로 하자.

비트 연산자

비트 연산자를 사용하면 정수를 비트들의 나열로 처리할 수 있다. PHP에서는 별로 사용할 일이 없긴 하지만 [표 1.5]에 비트 연산자를 나열해 보았다.

[표 1.5] PHP 비트 연산자

연산자	이름	사용법	결과
&	비트 단위 AND	$a & $b	$a와 $b 모두 1인 비트만 1로 바꾼다.
|	비트 단위 OR	$a | $b	$a와 $b 중 하나라도 1인 비트를 1로 바꾼다
~	비트 단위 NOT	~$a	$a의 0인 비트를 1로, 1인 비트를 0으로 바꾼다.
^	비트 단위 XOR	$a ^ $b	$a와 $b 중 하나만 1인 비트를 1로 바꾼다.
<<	왼쪽 시프트	$a << $b	$a의 비트를 $b 값만큼 왼쪽으로 옮긴다.
>>	오른쪽 시프트	$a >> $b	$a의 비트를 $b 값만큼 오른쪽으로 옮긴다.

기타 연산자

앞에서 언급한 연산자 이외에도 다른 연산자들이 더 있다.

쉼표 연산자 ','는 함수 인자와 같이 상품들을 열거할 때 사용된다.

new와 '->'는 클래스를 새로 생성하거나, 클래스의 멤버에 접근할 때 사용되는데, 이것은 6장에서 자세히 알아볼 것이다.

여기서는 몇 가지 다른 연산자들을 간단히 알아보자.

삼항 연산자

"?:"이라는 연산자는 C 언어에서의 사용 방법과 같다.

```
조건 ? 참일 때의 값 : 거짓이었을 때의 값
```

삼항 연산자는 if-else 문과 비슷한데, 좀 복잡하므로 예를 통해 알아보자.

```
($grade > 50 ? 'Passed' : 'Failed');
```

이 표현식은 만약 $grade의 값이 50을 넘지 않으면 'Failed'라는 값을 리턴하고 50을 넘으면 'Passed'를 리턴한다.

에러 억제 연산자

에러 억제 연산자(error suppression operator) '@은 어떠한 표현식 앞에서도 쓸 수 있다. 예를 들어보자.

```
$a = @(57/0);
```

'@가 없다면 위 문장은 "0으로 나누기"에 의한 에러가 발생하지만 '@를 앞에 써주면 에러는 무시되고 넘어간다.

만약 에러를 이런 방식으로 억제한다면 에러를 처리하는 코드를 작성해야 한다. 만약 PHP에서 track_errors를 설정해 놓았다면 에러 메시지를 전역 변수인 $php_errormsg에 저장한다.

실행 연산자

실행 연산자(execution operator)는 ``와 ``로 이루어진 한 쌍의 연산자로 ``는 홑따옴표(')와 다른 기호로 키보드에서는 '~' 아래에 있다.

PHP에서 서버의 커맨드라인에서 실행하고 싶은 것이 있다면 `` 사이에 명령어를 쓰면 된다. 그러면 ``의 결과값이 표현식의 리턴값이 된다.

유닉스와 같은 환경이라면 다음과 같이 입력해보자.

```
$out = `ls -la`;
echo '<pre>'.$out.'</pre>';
```

혹은 윈도우즈 서버를 사용하고 있다면 다음과 같이 해보자.

```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

그러면 C 디렉토리의 파일 리스트가 화면에 보이게 된다. 이들은 18장 "파일 시스템과 서버와의 연동"에서 사용해볼 것이다.

배열 연산자

배열 연산자 중 배열 요소 연산자인 "[]"를 사용하면 배열의 요소에 접근하여 사용할 수 있게 된다. 특정 배열 구성에서는 "=>" 연산자를 사용할 수도 있다. 이들은 3장에서 더 자세하게 다룰 것이다.

배열에 접근하는 연산자가 꽤 있으나 3장에서 자세하게 다룰 것이므로 여기서는 일단 배열 연산자에 대해 간단하게 언급하고 넘어간다.

[표 1.6] PHP의 배열 연산자

연산자	이름	사용법	결과
+	합집합	$a + $b	$a와 $b가 가지고 있는 모든 값을 반환한다.
==	등위	$a == $b	$a와 $b가 같은 요소를 가지고 있으면 true를 반환한다.
===	동일	$a === $b	$a와 $b가 같은 순서로 같은 요소를 가지고 있으면 true를 반환한다.
!=	같지 않다	$a != $b	$a와 $b가 같지 않으면 true를 반환한다.
<>	같지 않다	$a <> $b	$a와 $b가 같지 않으면 true를 반환한다.
!==	동일하지 않다	$a !== $b	$a와 $b가 동일하지 않으면 true를 반환한다.

[표 1.6]에 보면 스칼라 변수가 사용하는 연산자와 동일한 배열 연산자를 찾을 수 있다. 스칼라 형에서는 '+'가 덧셈이지만 배열에서는 합집합을 하는 연산자로서 '+'라는 산술적 의미와는 상관없이 데이터 형에 맞는 연산을 처리한다. 배열은 스칼라 형과 비교할 수 없다.

형 연산자

형 연산자는 instanceof 하나이다. 이 연산자는 객체 지향 프로그램에서 사용한다(객체 지향 프로그램은 6장에서 언급한다). instanceof 연산자는 특정 클래스의 객체인지 알아볼 때 사용할 수 있다.

```
class sampleClass{};
$myObject = new sampleClass();
if ($myObject instanceof sampleClass)
echo "myObject is an instance of sampleClass"
```

연산자 사용하기 : 폼에서 받은 값 처리하기

이제 PHP 연산자를 어떻게 사용하는지 알았으므로 Bob의 주문 폼을 가지고 총합과 세금을 계산해보자.

일단 다음 코드를 PHP 스크립트의 아래쪽에 덧붙여보자.

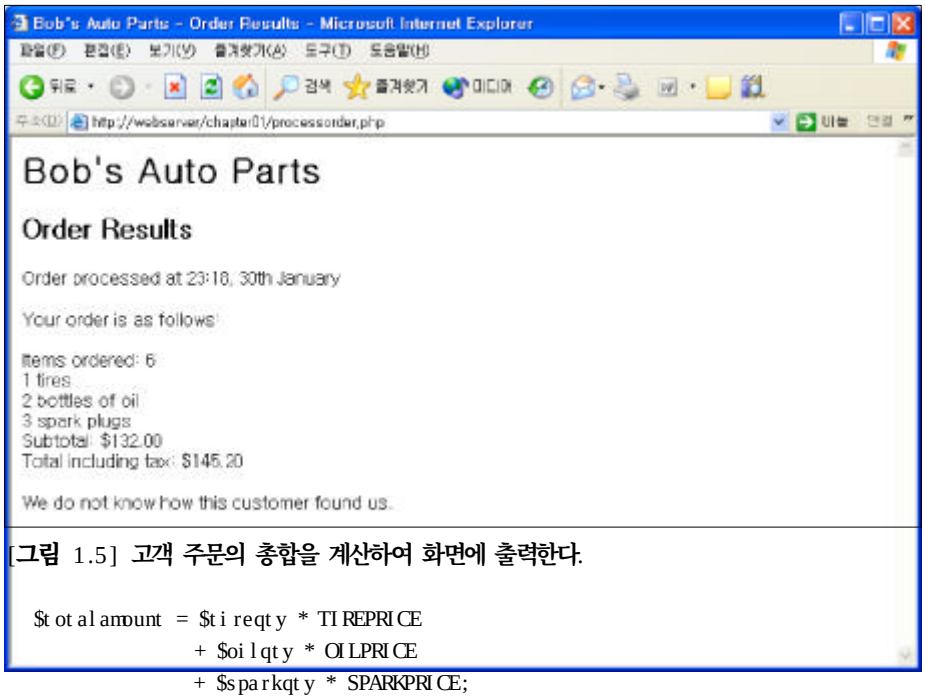
```
$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

echo 'Subtotal: $'.number_format($totalamount,3).'<br />';

$taxrate = 0.10; // 세금이 10%이다.
$totalamount = $totalamount * (1 + $taxrate);
echo 'Total including tax: $'.number_format($totalamount,2).'<br />';
```

브라우저에서 '새로 고침'을 눌러보자. [그림 1.5]와 비슷한 결과가 나왔을 것이다.

이번에 덧붙인 코드에는 여러 가지 연산자를 사용하였다. '+'와 '*' 연산자를 사용해서 총합을 구하였고 '.'를 사용하여 출력할 형태를 정하였다.

그리고 number_format() 함수를 사용하여 총합을 소수점 이하 두 자리 숫자로 표현하여 보았다. 이 함수는 PHP의 Math 라이브러리에 포함되어 있다.

계산 과정을 자세히 보면 우리가 알고 있는 상식적인 순서대로 계산이 되어 있다는 것을 알 수 있다.



[그림 1.5] 고객 주문의 총합을 계산하여 화면에 출력한다.

```
$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;
```

총 가격은 제대로 나온 것 같은데 어떻게 곱셈이 덧셈보다 먼저 계산되었는지 궁금하지 않은가? 그 이유는 연산자마다 우선순위가 있기 때문이다.

우선순위와 결합 순서 : 수식 처리

연산자들은 계산되는 순서를 지정한 우선순위(precedence)를 가지고 있다.

또한 연산자들이 같은 우선순위일 때 오른쪽이 먼저 계산될 것인지 왼쪽이 먼저 계산될 것인지를 결정하는 결합 순서라는 것도 가지고 있다. 순서는 주로 왼쪽에서 오른쪽(짧게 왼쪽이라고 한다), 오른쪽에서 왼쪽(짧게 오른쪽이라고 한다) 혹은 상관없음이 있다.

[표 1.7]에 연산자의 우선순위와 결합 순서가 나와있다.

이 표는 맨 위에 가장 낮은 우선순위 연산자부터 해서 아래로 갈수록 우선순위가 높아진다.

[표 1.7] 연산자 우선순위

결합 순서	연산자
왼쪽	,
왼쪽	or
왼쪽	xor
왼쪽	and
오른쪽	print
왼쪽	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
왼쪽	? :
왼쪽	||
왼쪽	&&
왼쪽	|
왼쪽	^
왼쪽	&
없음	== != === !==
없음	< <= > >=
왼쪽	<< >>
왼쪽	+ - .
왼쪽	* / %
오른쪽	! ~ ++ -- (int) (double) (string) (array) (object) @
오른쪽	[ ]
없음	new
없음	( )

지금까지 우선순위가 제일 높은 연산자를 언급한 적은 없지만 괄호 '(,)'라는 것을 잘 알고 있을 것이다. '(', ')'는 그 안에 든 것이 무엇이든 간에 우선순위를 높여준다. 따라서 원하는 경우 우선순위 법칙을 바꿀 수 있다.

예제의 마지막 부분을 다시 떠올려보자.

```
$totalamount = $totalamount * (1 + $taxrate);
```

이 식에서 괄호를 제거해보자.

```
$totalamount = $totalamount * 1 + $taxrate;
```

이 식을 계산하면 곱셈 연산자가 덧셈 연산자보다 더 높은 우선순위를 가지므로 먼저 계산되어 원하는 값과 다른 값이 나온다. 이때 괄호를 사용하면 1+$taxrate를 먼저 계산할 수 있다.

표현식에서 원한다면 많은 괄호를 사용할 수 있다. 가장 내부에 위치한 괄호가 먼저 계산된다.

또 앞에서 언급하지 않은 다른 연산자가 표에 나와있다. print는 echo와 같은 역할을 하여 둘 다 출력을 생성한다.

이 책에서는 주로 echo를 사용하지만 print가 더 좋다면 print를 사용해도 좋다. 함수는 괄호 안에 든 인자를 받으므로 print나 echo는 진짜 함수가 아니다. 그래서 둘 다 연산자로 취급한다.

print를 함수처럼 호출하면 1을 반환한다. 만약 좀 더 복잡한 표현식 안에서 출력을 생성하려 할 때 유용하지만 print는 대체로 echo보다 느리다.

변수와 관련된 함수

변수와 연산자에 대한 설명을 마치기 전에 PHP에서 사용할 수 있는 변수와 관련된 함수들을 살펴보자. PHP는 변수를 다루고 검사하는 다양한 방법을 제공하는 함수들의 라이브러리를 가지고 있다.

변수의 데이터 형을 검사하고 설정하기

가장 널리 쓰이는 함수는 gettype()과 settype()으로 이 함수들의 프로토타입을 알아보자(프로토타입은 어떤 형의 인자를 받아들여서 어떤 형의 데이터를 리턴하는지 알려준다).

```
string gettype(mixed var);
bool settype(mixed var, string type);
```

gettype()을 사용하기 위해 변수를 넘겨주면 이 함수는 데이터형을 결정하여 "boolean", "integer", "double", "string", "array", "object", "resource" 혹은 NULL을 반환한다. gettype()은 표준 데이터 형이 아닌 경우 "unknown"을 반환한다.

settype()을 사용할 때에는 변수와 바꾸고 싶은 형을 문자열로 넘겨주어야 한다.

NOTE >>

이 책과 php.net 문서는 mixed라는 데이터 형을 사용한다. 실제로 그런 데이터 형은 없지만 PHP는 데이터 형 처리에 매우 유연하기 때문에 많은 함수들이 다양한(혹은 어떤 것이든) 데이터 형을 받아들일 수 있다. 다양한 데이터 형을 받아들일 수 있는 인자의 경우 mixed로 표시할 수 있다.

이 함수를 다음과 같이 사용할 수 있다.

```
$a = 56;
echo gettype($a).'<br />';
settype($a, 'double');
echo gettype($a).'<br />';
```

gettype()이 처음 호출되었을 때 $a는 정수형이었지만 settype()이 호출된 뒤에는 실수형으로 바뀌었다.

PHP는 그 외에도 몇몇 형 검사 함수를 제공하고 있는데, 모두 변수를 인자로 받아들여 true나 false를 리턴한다.

- is_array()
- is_double(), is_float(), is_real() (모두 같은 함수)
- is_long(), is_int(), is_integer() (모두 같은 함수)
- is_string()
- is_object()
- is_resource()
- is_null()
- is_scalar() — 변수가 스칼라 변수인지 확인한다. 즉 정수형, 불리언형, 문자열이나 실수형인지 확인한다.
- is_numeric() — 변수가 숫자나 혹은 숫자 문자열인지 확인한다.
- is_callable() — 변수에 저장된 값이 호출할 수 있는 함수의 이름인지 확인한다.

변수 상태 검사

PHP에서 변수의 상태를 검사하는 방법은 여러 가지가 있다. 맨 먼저 isset()의 프로토타입을 알아보자.

```
boolean isset(mixed var);
```

이 함수는 변수의 이름을 인자로 받아서 만약 이 변수가 존재한다면 true를, 아니면 false를 리턴한다. ‘,’로 연결된 변수들을 인자로 주면 isset()은 인자로 받은 모든 변수가 존재해야 true를 반환한다.

만약 어떤 변수를 삭제하고 싶다면 unset()이라는 함수를 사용하면 된다.

```
void unset(mixed var);
```

unset()을 사용하면 인자로 받은 변수의 존재 자체를 없애고 true를 리턴한다.

마지막으로 empty()라는 함수가 있는데, 이 함수는 변수가 존재하고, 비어 있지 않으며, 0이 아닌 값을 가지고 있다면 true를 리턴하고, 아니면 false를 리턴한다.

```
boolean empty(mixed var);
```

자, 이제 이 함수들을 사용해보자.

스크립트에 다음과 같은 코드를 잠시 덧붙여보자.

```
echo 'isset($tireqty): '.isset($tireqty).'<br />'
echo 'isset($nothere): '.isset($nothere).'<br />'
echo 'empty($tireqty): '.empty($tireqty).'<br />'
echo 'empty($nothere): '.empty($nothere).'<br />'
```

그런 다음 브라우저에서 ‘새로고침’을 누르고 결과를 살펴보자.

$tireqty 안에 값이 있든 없든 간에 isset()은 true를 리턴하겠지만, 만약 필드에 어떤 값을 넣느냐에 따라 empty()의 리턴값은 달라진다.

$nothere는 없는 변수이기 때문에 isset()이 false를 리턴하겠지만 empty()는 true를 리턴한다.

이런 함수들을 사용하면 사용자가 입력해야 하는 필드들을 검사할 수 있다.

변수형 변환

함수를 사용하여 변수를 "형 변환(type casting)"을 해보자. 세 가지 형 변환 함수가 있다.

```
int intval(mixed var[, int base]);
float floatval(mixed var);
string strval(mixed var);
```

이 함수들은 인자로 받은 변수의 값을 적절한 형으로 변환해서 리턴한다. intval() 함수는 문자열을 변환할 때 사용할 값을 명시할 수 있도록 한다(예로 들면 16진수 문자열을 정수로 전환한다라고 명시할 수 있다).

제어 구조

프로그램이나 스크립트에서 실행되는 순서를 다룰 수 있게 하는 구조를 제어 구조라고 한다. 제어 구조는 조건 구조(branching)와 반복 구조 혹은 루프로 나눌 수 있다. 그 각각을 살펴보기로 하자.

조건식 만들기

사용자의 입력에 민감하게 반응하려면 스크립트가 어떤 상황에 대한 결정을 내릴 수 있어야 한다. 결정을 내릴 수 있는 방법을 "조건"이라고 부르자.

if 문

결정을 내리기 위해 if 문을 사용할 수 있다. if 문을 사용하기 위해서는 조건을 주어야 하는데, 만약 그 조건이 true라면 if 문에 뒤따르는 코드들이 수행된다. if 문의 조건문은 괄호'('와 ')'로 둘러싸여 있어야 한다.

예를 들어, 사용자가 타이어와 기름, 스파크 플러그를 주문하지 않았다면 아마도 사용자가 실수로 확인 버튼을 누른 것이라고 생각할 수 있다. 그럴 경우 "주문이 처리중입니다"라는 것보다 다른 메시지를 내보내는 것이 더 좋다.

만약 사용자가 아무 것도 사지 않았다면 "이전 페이지에서 아무 것도 안 사셨군요!"라고 말해보자. if 문을 사용해서 구현해보자.

```
if( $totalqty == 0 )
echo 'You did not order anything on the previous page!<br />';
```

조건으로 사용한 것은 $totalqty == 0이다. 여기서 등위 연산자가 대입 연산자와 다르다는 것을 다시 한 번 기억해두자.

만약 $totalqty == 0이 true이면 $totalqty가 0이라는 것이다. 만약 $totalqty가 0이 아니라면 조건은 false가 된다. 만약 조건이 true이면 echo 문이 실행된다.

코드 블록

if 문과 같은 조건문 다음에 실행할 코드가 여러 문장이 될 수도 있다. 이러한 경우에는 if 문을 또다시 쓰는 대신에 여러 문장을 블록으로 묶어서 사용한다. 블록으로 묶을 문장들을 '{', '}'로 감싸준다.

```
if( $totalqty == 0 )
{
  echo '<font color=red>';
  echo 'You did not order anything on the previous page!<br />';
  echo '</font>';
}
```

세 문장이 '{', '}'로 묶여 한 블록의 코드가 되었다. 만약 조건이 true이면 세 문장 모두 실행되고 false라면 당연히 세 문장 모두 실행되지 않는다.

NOTE >>

> 이미 말했듯이 PHP는 여러분이 어떻게 코드를 작성하든 신경 쓰지 않지만 그래도 가독성을 높이려면 코드에 들여쓰기를 하는 것이 좋다. 들여쓰기를 하면 if 조건이 맞을 경우 어떤 코드가 실행되는지 한 눈에 알아볼 수 있다. 앞 예제에서는 블록 안의 문장을 들여쓰기 하였다.

else 문

조건이 맞을 때에만 실행하는 것이 아니라 여러 가지 코드를 상황에 따라 골라서 실행하고 싶은 경우가 있다.

else 문을 사용하면 if 문이 false가 되었을 때 실행할 코드를 정의할 수 있다. 만약 Bob의 고객이 아무 것도 사지 않았다면 그에 대해 경고하는 메시지를 보내고 싶다고 하자. 그 경우 고객이 주문을 했다면 경고 대신에 고객이 주문한 물품을 보여주자.

다음과 같이 코드를 수정하면 경고나 주문 물품을 보여줄 수 있다.

```
if( $totalqty == 0 )
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  echo $tireqty.' tires<br />';
  echo $oilqty.' bottles of oil<br />';
  echo $sparkqty.' spark plugs<br />';
}
```

if 문 안에 if 문을 다시 사용한다면 좀 더 복잡한 상황도 처리할 수 있다. 코드를 다음과 같이 작성하면 $totalqty == 0일 때에만 아무 것도 사지 않았다고 경고하며 각각 주문한 물품의 수에 따라 무엇을 몇 개를 샀는지 화면에 표시해준다.

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  if ( $tireqty>0 )
    echo $tireqty.' tires<br />';
  if ( $oilqty>0 )
    echo $oilqty.' bottles of oil<br />';
  if ( $sparkqty>0 )
    echo $sparkqty.' spark plugs<br />';
}
```

elseif 문

두 개 이상의 결정을 내려야 할 때도 있다. 이때는 elseif 문을 사용해서 각각의 상황을 처리할 수 있다. 프로그램이 나열된 조건문을 차근차근 읽어가면서 조건이 참이면 그 조건에 맞는 코드를 수행한다.

타이어를 많이 사는 사람에게 할인 혜택을 준다고 하자. 그 할인율은 다음과 같다.

- 10개 이하 — 할인율 0%
- 10~49개 — 5%
- 50~99개 — 10%
- 100개 이상 — 15%

if와 elsief 문을 사용하여 위의 할인율에 맞게 코드를 만들어보자. 두 개의 조건을 연결하기 위해서 AND 연산자(&&)를 사용한다.

```
if( $tireqty < 10 )
  $discount = 0;
elseif( $tireqty >= 10 && $tireqty <= 49 )
  $discount = 5;
elseif( $tireqty >= 50 && $tireqty <= 99 )
  $discount = 10;
elseif( $tireqty >= 100 )
  $discount = 15;
```

elseif는 else if와 같기 때문에 어느 것을 사용하더라도 상관없다.

elseif 문을 연속해서 사용할 때에는 연속된 if 문 중에서 하나만 실행된다는 데 주의하자. 위의 예제는 어떤 상황에서도 한 조건만이 참이 되기 때문에 상관없지만, 만약 조건문 두 개가 동시에 참이 될 수 있다면 그 중에서 먼저 검사되는 조건문의 코드만이 실행된다.

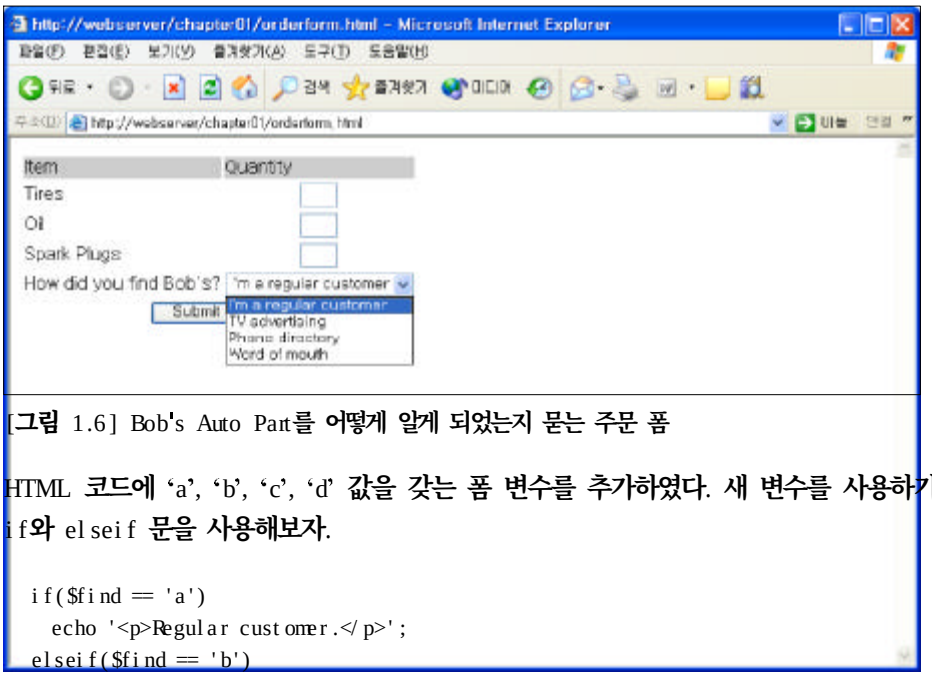
switch 문

switch 문은 if 문과 비슷한 방식으로 사용할 수 있지만 조건문에 한 가지 값 이상을 받아들일 수 있다. if 문에서는 조건이 true가 아니면 false였지만, switch 문에서는 조건이 간단한 형(정수형, 실수형, 문자열)으로만 바뀔 수 있다면 어떤 형태의 값이어도 된다. 각각의 조건에 대한 처리는 case 문에서 수행되며 어떤 case 문에서도 맞지 않은 값을 처리하기 위해서 default case 문을 두는 경우도 있다.

Bob이 만약 어떤 방법으로 광고를 하는 것이 좋을지 알고 싶다면 주문 폼에 다음과 같은 코드를 추가시켜 알아볼 수도 있다. 폼은 [그림 1.6]과 비슷할 것이다.

```
<tr>
  <td>How did you find Bob's</td>
  <td><select name="find">
```

```
            <option value = "a">I'm a regular customer
            <option value = "b">TV advertising
            <option value = "c">Phone directory
            <option value = "d">Word of mouth
          </select>
      </td>
    </tr>
```



[그림 1.6] Bob's Auto Part를 어떻게 알게 되었는지 묻는 주문 폼

HTML 코드에 'a', 'b', 'c', 'd' 값을 갖는 폼 변수를 추가하였다. 새 변수를 사용하기 위해서 if와 elseif 문을 사용해보자.

```
if($find == 'a')
  echo '<p>Regular customer.</p>';
elseif($find == 'b')
  echo '<p>Customer referred by TV advert.</p>';
elseif($find == 'c')
  echo '<p>Customer referred by phone directory.</p>';
elseif($find == 'd')
  echo '<p>Customer referred by word of mouth.</p>';
else
  echo '<p>We do not know how this customer found us.</p>';
```

switch 문으로도 똑같이 해보자.

```
switch($find)
{
  case 'a' :
    echo '<p>Regular customer.</p>';
```

```
    break;
  case 'b' :
    echo '<p>Customer referred by TV advert.</p>';
    break;
  case 'c' :
    echo '<p>Customer referred by phone directory.</p>';
    break;
  case 'd' :
    echo '<p>Customer referred by word of mouth.</p>';
    break;
  default :
    echo '<p>We do not know how this customer found us.</p>';
    break;
}
```

(두 예제들은 모두 $_POST 배열에서 원하는 값을 $find로 복사하였다고 가정하였다.)

switch 문은 if와 elseif 문과는 조금 다른 방식으로 실행된다. if 문은 단 하나의 코드 블록만을 실행하지만 switch 문 내의 case 문이 실행되면 break 문이 나오기 전의 문장들을 모두 실행한다. break 문을 사용하지 않으면 case 문이 true가 되는 시점부터 모든 코드를 실행한다. break 문이 나오게 되면 switch 블록 밖의 코드로 이동하게 된다.

조건문 비교하기

조건문에 익숙하지 않으면 "뭐가 제일 좋은 거야"하고 불평하게 된다.

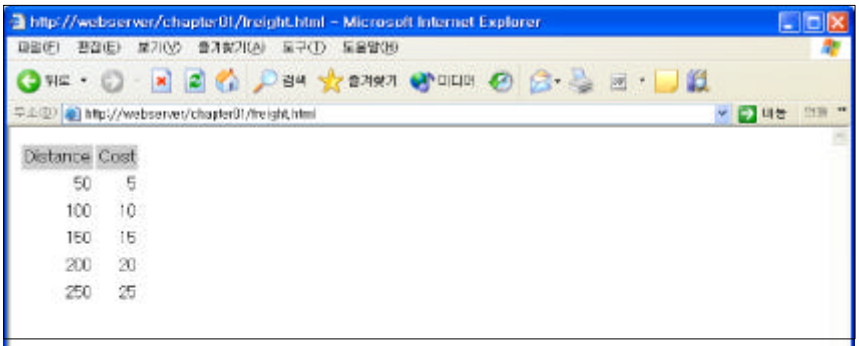
항상 좋은 조건문이란 없다. else, elseif, switch 문을 써서 만들 수 있는 조건문이라면 if 문을 여러 개 써서 만들 수도 있다. 상황에 맞춰 가장 적절하다고 생각하는 조건문을 사용하자. 경험이 늘면 어떤 것을 사용해야 할지 잘 알 수 있게 된다.

반복문

컴퓨터는 인간과 달리 단순하고 반복적인 일을 즐기기 때문에 프로그램에서 똑같은 일을 여러 번 해야 할 때 반복문(루프문)을 사용하는 것이 좋다.

주문 배송에 드는 배송비도 같이 보여주자. 배달해야 하는 거리에 따라 배송비가 비례한다는 것을 간단한 식을 사용해서 보여주도록 하자.

[그림 1.7]과 같은 배송비 표를 만들어보자.



[그림 1.7] 거리에 따른 배송비를 나타내는 표

[리스트 1.2]가 이 표의 HTML인데 꽤 길고 반복되는 부분이 많다.

[리스트 1.2] freight.html — 배송비

```
<html>
<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<tr>
  <td align="right">50</td>
  <td align="right">5</td>
</tr>
<tr>
  <td align="right">100</td>
  <td align="right">10</td>
</tr>
<tr>
  <td align="right">150</td>
  <td align="right">15</td>
</tr>
<tr>
  <td align="right">200</td>
  <td align="right">20</td>
</tr>
<tr>
  <td align="right">250</td>
  <td align="right">25</td>
</tr>
```

```
</table>
</body>
</html>
```

지루하게 직접 코드를 쳐 넣기보다는 싸고 지칠줄 모르는 컴퓨터가 작성하도록 해보자. 반복문을 사용하면 한 문장이나 블록을 반복해서 실행시킬 수 있다.

while 문

PHP에서 while 문이 가장 간단한 루프로, if 문처럼 조건문을 사용한다. while 문은 if 문과 달리 while의 조건문이 true인 동안 반복 실행된다.

while 문은 조건문이 언제쯤 false가 될지 확실히 알 수 없을 때에 주로 사용하며, 일정하게 반복된다면 for 문을 주로 사용한다.

while 문의 기본 구조는 다음과 같다.

```
while( condition ) expression;
```

while 문을 사용하여 1에서부터 5까지 화면에 출력해보자.

```
$num = 1;
while ($num <= 5 )
{
  echo $num."<br />";
  $num++;
}
```

매 반복마다 조건문을 검사하여 false이면 while 블록은 수행되지 않고 루프를 빠져나간다.

while 문으로 [그림 1.7]과 같은 배송비 표를 만들어보자.

[리스트 1.3] freight.php — PHP로 Bob의 배송비 표 만들기

```
<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Cost</td>
```

```
</tr>
<?
$distance = 50;
while ($distance <= 250 )
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
  $distance += 50;
}
?>
</table>
</body>
</html>
```

스크립트가 만드는 HTML의 가독성을 높이기 위해서 줄바꿈 문자와 공백을 넣어주는 것이 좋다. 앞에서도 말했지만 브라우저는 공백을 무시하지만 사람들이 읽는 데는 중요한 요소이다.

**[리스트 1.3]을 보면 문자열 내에 줄바꿈 문자(\n)가 있다. 만약 겹따옴표(") 문자열 내에 줄바꿈 문자가 있다면 HTML에서
로 바뀐다.**

for와 foreach 루프

while 문을 사용할 때 카운터를 쓴다면 for 문을 사용하여 깔끔하게 구현해보자.

for 문의 기본 구조는 다음과 같다.

```
for( expression1; condition; expression2)
     expression3;
```

- ***expression1*은 시작할 때 한 번 실행된다. 여기서 카운터의 초기값을 설정해줄 수 있다.**
- ***condition*은 반복을 하기 전에 검사하며 조건이 false를 리턴하면 루프를 빠져나간다.**
- ***expression2*는 매 반복의 마지막에 실행되는데 여기서 카운터의 값을 조정한다.**
- ***expression3*은 반복해서 실행할 코드 부분이다. 이 표현식은 코드 블록으로 이루어진다.**

[리스트 1.3]에서 while을 사용했던 부분을 for 문으로 바꾸어보자.

```
<?php
for($distance = 50; $distance <= 250; $distance += 50)
{
```

```
    echo "<tr>\n  <td align='right'>$distance</td>\n";
    echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
  }
  ?>
```

while 문을 사용하든 for 문을 사용하든 하는 일은 동일하지만 for 문을 사용하는 것이 좀 더 깔끔해 보인다.

두 루프 방식은 거의 동일하다. 어느 것도 더 낫거나 더 나쁘지 않다. 주어진 상황에 맞는 루프를 사용하자.

덧붙여서 for 문에서 가변 변수를 사용하여 폼 필드를 사용할 수도 있다. 예를 들어, 폼 필드의 이름을 name1, name2, name3과 같은 방식으로 정했다고 해보자.

```
for ($i=1; $i <= $numnames; $i++)
{
  $temp= "name$i";
  echo $$temp.'<br />'; // 혹은 실행하고 싶은 코드를 작성한다.
}
```

변수의 이름을 동적으로 생성해서 순서대로 폼 필드를 사용할 수 있다.

foreach 문은 for 문을 조금 바꾸어서 배열을 사용하기 쉽게 만든 것인데 지금은 그냥 넘어가고 3장에서 자세히 알아보자.

do..while 문

do...while 문은 다른 루프와 조금 다르다. 먼저 기본 구조를 살펴보자.

```
do
  expression;
while( condition );
```

do...while 문은 while 문과 달리 조건을 마지막에 검사한다. 따라서 do...while 문에서는 블록이 적어도 한 번은 실행된다.

설사 조건이 절대 true가 될 수 없다 하더라도 조건을 마지막에 검사하기 때문에 블록을 한 번은 실행하게 된다.

```
$num = 100;
do
{
  echo $num.'<br />';
}
while ($num < 1);
```

제어 구조와 스크립트 빠져나가기

코드를 빠져나가고 싶을 때에는 세 가지 방법을 사용할 수 있다.

만약에 루프를 빠져나가고 싶다면 switch 문에서 언급한 break 문을 사용할 수 있다. break 문을 루프에서 사용하면 스크립트는 루프를 빠져나와 루프 다음에 있는 문장을 실행한다.

만약에 지금 루프를 끝내고 다음 번 루프로 넘어가고 싶다면 continue 문을 사용할 수 있다.

아예 PHP 스크립트를 끝내고 싶다면 exit 문을 사용할 수 있다. exit 문을 사용해서 오류를 검사해볼 수 있다. 앞 예제에 exit 문을 추가해보자.

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
  exit;
}
```

exit 문을 호출하는 순간 PHP를 완전히 끝내게 된다.

대체 제어 구조 문법 사용하기

이제까지 살펴본 제어 구조들에 대해 다른 형태를 사용할 수 있다. '{' 대신에 ':'을 사용하고 '}' 대신에 사용한 제어 구조에 따라 새로운 키워드인 endif, endswitch, endwhile, endfor, endforeach를 사용할 수 있다. 그러나 do...while 루프는 대체 문법이 존재하지 않는다.

예를 들어,

```
if( $totalqty == 0 )
{
   echo 'You did not order anything on the previous page!<br />';
   exit;
```

```
}
```

는 if와 endif를 사용하여 대체 문법으로 바꿀 수 있다.

```
if( $totalqty == 0 ):
   echo 'You did not order anything on the previous page!<br />';
   exit;
endif;
```

declare 사용하기

PHP에는 declare라는 또다른 제어 구조가 있는데 다른 구조에 비해서는 자주 쓰이지 않는 편이다. declare의 일반적인 형태는 다음과 같다.

```
declare (directive)
{
// block
}
```

이 구조는 코드 블록에서 실행 지침(execution directive)을 설정하기 위해 사용된다. 즉, 다음 코드가 실행될 법칙을 정의한다. 지금은, 단 하나의 실행 지침인 ticks만이 구현되어 있다. ticks=n이라는 방식으로 지침을 삽입할 수 있다. 코드 블록 내에 있는 특정 함수 중 n개의 코드 행을 실행시킬 수 있어 디버깅에 매우 유용하다.

declare 제어 구조는 여기서는 이만큼만 언급하고 tick을 사용하는 방법은 24장 "대규모 프로젝트에서 PHP와 MySQL"과 25장 "디버깅"에서 자세하게 알아보자.

다음 장에서는 : 고객 주문 저장하기

이제 고객의 주문을 받아서 사용하는 법을 알았으므로 다음 장에서는 주문을 어떻게 저장하고 어떻게 저장한 값을 불러와서 사용하는지 알아보자.