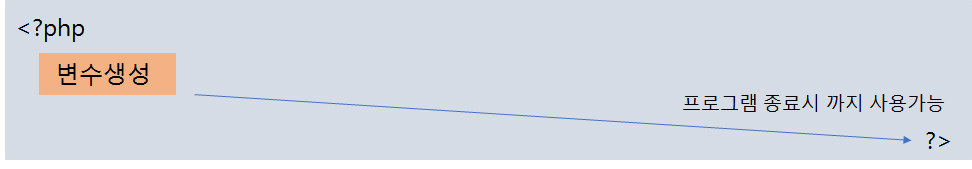
jinyPHP 1권 - 12. 변수 유효범위

**12 변수 유효 범위**

이번 장에서는 변수의 유효 범위(scope)에 대해서 알아보겠습니다. 변수의 유효 범위란 변수의 생성과 접근을 할 수 있는 범위를 말합니다.

프로그램에서 변수를 생성, 사용 선언하게 되면 PHP는 메모리에 변수의 공간을 추가로 할당합니다. 이렇게 생성된 변수는 별도의 해제 작업을 하지 않는 경우 스크립트 실행 종료까지 유지하게 됩니다.



변수의 유효 범위란 변수를 사용할 수 있는 프로그램 안에서의 코딩 공간이라고 이해하면 됩니다.

즉, 변수가 지배하는 땅과 같습니다.

PHP에서는 변수의 영역에 대해서 다음과 같은 세 가지 타입을 가지고 있습니다.

* Global : 글로벌변수
* Local : 지역변수
* Static : 정적변수

**12.1 글로벌변수**

절차적인 간단한 프로그램들은 변수의 영역이라는 개념이 생소합니다. PHP 실행 코드 <?php 다음에 선언하는 변수들은 글로벌변수로 처리를 합니다.

PHP에서는 대부분 변수 선언과 동시에 사용이 가능하고 현재의 스크립트 종료까지 변수는 메모리와 데이터 값이 유효합니다.

|  |
| --- |
| <?php  $x = 5; //글로벌변수  $name = "jiny"; //글로벌변수    ?> |

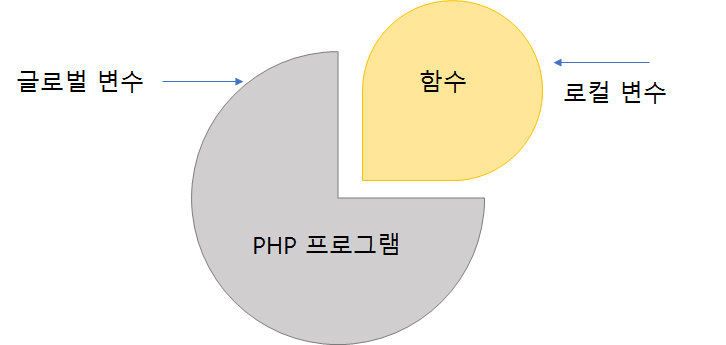
위의 예제에서 정의된 $x 변수는 스크립트 종료까지 메모리에 공간이 유효하게 사용됩니다.

**12.2 로컬변수**

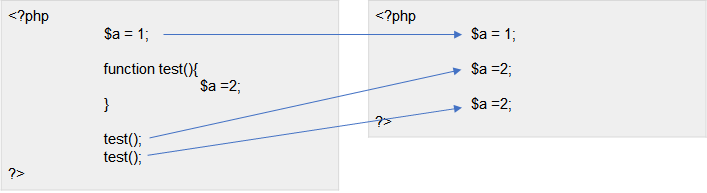
로컬변수의 개념은 함수의 사용과 연관이 있습니다. 함수는 코드의 재사용과 독립성을 보장하는 코드 작성 방법입니다. 또한 함수는 기존 코드들과 분리된 각각의 코드 모듈입니다.

함수는 프로그램에서 여러 번 호출을 통하여 재사용됩니다. 만일, 함수가 호출될 때마다 외부 변수 값에 영향을 받거나, 이전 호출한 함수의 작업된 변수의 영향이 있다고 한다면 함수를 재사용하는 데 많은 어려움이 있을 것입니다.

따라서 함수에서 사용되는 변수는 외부의 변수들과 단절하고 독립성이 필요합니다. 따라서 함수 내부에서 선언한 변수는 함수 안에서만 사용을 할 수 있도록 독립성을 유지하도록 설계되었습니다. 이렇게 함수 내부에서 독립된 선언과 사용을 할 수 있도록 하는 변수를 로컬변수라고 합니다.



위의 그림처럼 PHP에서 생성하는 변수들은 대부분 글로벌변수로 작성됩니다. 또한 함수에서 작성 및 사용되는 변수는 로컬변수로 인식합니다.



위의 그림처럼 <?php 다음에 처음으로 선언된 $a는 글로벌변수입니다. 또한 test() 함수 안에 동일한 이름의 변수$a는 외부의 $a와 다른 변수입니다. 함수 안에서만 사용 가능한 변수가 로컬변수입니다.

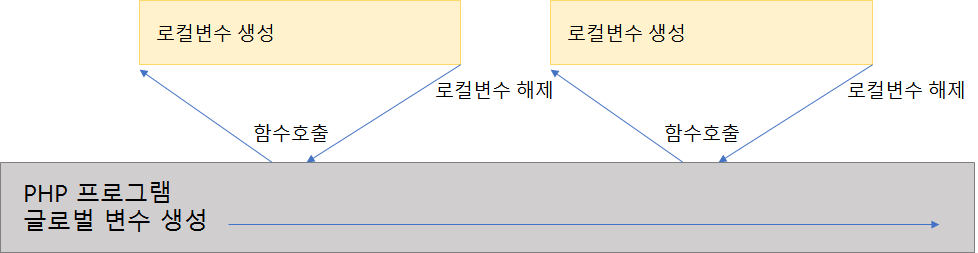
이 프로그램을 예를 실행하면 한 개의 글로벌 변수와 두 번의 로컬변수가 생성됩니다. 하지만 세 개의 변수명은 모두 같습니다. 변수명이 같을지라도 PHP는 세 개의 메모리 영역을 할당하게 됩니다. 즉, 서로 다른 변수입니다.

**12.2.1 휘발성 변수**

로컬변수는 주로 함수 내에서 선언한 변수를 말합니다. 또한 함수 내에서 생성된 로컬변수는 휘발성을 띠고 있습니다.

휘발성이란 변수를 사용하고 나서 더 이상 필요 없을 경우에 자동으로 없어지는 효과를 말합니다.

로컬변수도 함수를 호출할 때 함수 내에서 사용되는 변수 메모리를 할당합니다. 또한 함수가 종료되면 로컬변수들은 자동으로 사라지게 됩니다. 함수 호출 처리가 끝났기 때문에 자동으로 없어지는 것입니다.



하지만 글로벌변수는 선언과 동시에 프로그램 종료까지 살아 있습니다. 그렇지만 로컬변수는 생성과 소멸을 반복합니다.

**12.2.2 변수 접근**

글로벌변수와 로컬변수는 서로 상호 영향을 받지 않도록 독립되어 있습니다. 심지어 변수의 이름이 같아도 서로 다른 메모리를 할당합니다. PHP 프로그램 안에서도 서로 각각의 값을 읽고 쓸 수 없도록 격리되어 있습니다.

이러한 로컬변수의 특성 때문에 함수를 재사용하고 기존 코드와 영향 없이 동작이 가능합니다.

**예제 파일 scope-01**.php

|  |
| --- |
| <?php  **$x = 5;** //글로벌변수  function test() {  // 함수 내에서는 함수 외부에 있는 글로벌변수를 읽을 수 없습니다.  echo "함수 안에서는 외부 변수를 사용할 수 없습니다. **x = $x**</p>";  }  test();  echo "외부에서는 글로벌변수를 사용할 수 있습니다. x = $x</p>";  ?> |

**결과**

함수 안에서는 외부 변수를 사용할 수 없습니다. x =

외부에서는 글로벌변수를 사용할 수 있습니다. x = 5

위의 예제는 함수 내에서 외부 글로벌변수의 접근을 실험합니다. 함수 내에서는 외부의 글로벌변수를 기본적으로 접근하여 사용할 수 없습니다. 함수 내에서 사용하고 있는 $x는 로컬변수로 생성된 별개의 새로운 변수 입니다.

또한 함수 안에서 사용한 로컬변수는 함수가 끝날 때 휘발성으로 자동 사라지게 됩니다.

**예제 파일 scope-02**.php

|  |
| --- |
| <?php    $a = "jiny";  $b = "lee";  echo "글로벌 변수를 출력합니다. <br>";  echo $a . " " . $b;  echo "<br>";    // 함수 정의  function usersName(){  // 로컬변수 선언  $title = "jinyPHP";  if ($a) {  echo "a =". $a . "<br>";  } else {  echo "a 값이 없습니다. <br>";  }  if ($b) {  echo "b =".$b . "<br>";  } else {  echo "b 값이 없습니다. <br>";  }  }  // 함수를 호출  echo "함수를 호출합니다.<br>";  usersName();  echo "<br>";  echo "함수 안에 title 변수를 확인합니다.<br>";  if ($title) {  echo "title =". $title . "<br>";  } else {  echo "title 값이 없습니다. <br>";  }  ?> |

**결과**

글로벌 변수를 출력합니다.

jiny lee

함수를 호출합니다.

a 값이 없습니다.

b 값이 없습니다.

함수 안에 title 변수를 확인합니다.

title 값이 없습니다.

기존의 PHP는 변수를 선언과 동시에 PHP 전체에서 선언한 변수를 사용을 할 수 있었습니다. 하지만 함수 안에서는 독립된 공간이기 때문에 외부 글로벌변수를 사용할 수 없습니다. PHP의 변수는 로컬변수와 글로벌변수 등으로 서로 독립되어 분리가 됩니다.

**12.2.3 글로벌 키워드**

앞 절에서 설명한 것과 같이 글로벌변수와 로컬변수는 서로 독립되어 간섭을 주지 않습니다. 메모리의 할당도 서로 다릅니다. 심지어 접근 또한 제한되어 있습니다.

함수들은 외부의 데이터를 입력받아 처리를 하고 값을 반환합니다. 하지만 프로그램 작성을 하다 보면 함수에게 전달하는 모든 값들을 매개변수로 전부 지정하여 사용하는 것은 어렵습니다. 이런 경우 공통의 외부 글로벌변수를 사용하면 편리할 수 있습니다. 공통의 변수를 함수 외부에서 사용하기 위해서는 함수 내에서 글로벌변수 선언키워드를 이용하여 지정하면 됩니다.

PHP는 외부 글로벌변수를 함수 내에서 예외적으로 접근 사용할 수 있는 **global 키워드**를 제공합니다. 함수 내에서 아래와 같이 선언을 하면,

|문법|

global $변수명;

함수 내에서 외부의 글로벌변수를 접근하여 사용할 수 있습니다.

**예제 파일 scope-03**.php

|  |
| --- |
| <?php    $a = "jiny";  $b = "lee";  echo "글로벌 변수를 출력합니다. <br>";  echo $a . " " . $b;  echo "<br>";    // 함수 정의  function usersName(){  // 글로벌 키워드를 통하여 외부 변수를 함수 내부에서 사용 가능하도록 만듭니다.  **global $a, $b;**  if ($a) {  echo "a =". $a . "<br>";  } else {  echo "a 값이 없습니다. <br>";  }  If ($b) {  echo "b =".$b . "<br>";  } else {  echo "b 값이 없습니다. <br>";  }  }  // 함수를 호출  echo "함수를 호출합니다.<br>";  usersName();  ?> |

**결과**

글로벌변수를 출력합니다.

jiny lee

함수를 호출합니다.

a =jiny

b =lee

위의 예제는 글로번변수 접근에 대한 실험입니다. global 키워드로 선언한 외부 글로벌변수는 값을 함수 내에서 가지고 올 수 있습니다. 또한 함수 내에서 값을 변경할 수도 있습니다.

**12.3 PHP static 키워드**

함수 안에 정의된 로컬변수는 휘발적인 특성이 있습니다. 즉, 함수가 종료되면 함수 내에서 사용된 변수들은 전부 자동 소멸하게 됩니다.

하지만, 함수 내의 로컬변수의 자동 소멸을 방지할 수 있는 방법 또한 있습니다. static 키워드는 함수를 재사용을 하면서 함수 내에서 선언한 변수가 소멸하지 않도록 할 수 있습니다. 아래와 같이 함수 내에서,

|문법|

static $변수명;

으로 선언을 하면, 함수가 종료된 후에도 해당 변수를 소멸하지 않고 변수를 남겨 놓을 수 있습니다.

이렇게 변수의 자동소멸을 유보한 변수는 함수의 재호출 사용 시 기존의 작업된 변수의 내용을 가지고 와서 계속 연산을 이어 갈 수 있습니다.

**예제 파일 scope-04**.php

|  |
| --- |
| <?php  function increment(){  // static 변수는 처음 한 번 선언 시에만 초기화 됩니다.  **static $total = 0;**  return ++$total;  }  echo increment() . "<br>";  echo increment() . "<br>";  echo increment() . "<br>";  ?> |

**결과)**

1

2

3

위의 예제는 정적변수의 실험입니다. increment() 함수 안에 정의한 정적변수 $total은 함수가 종료 후에도 자동으로 소멸하지 않습니다. 할당된 변수의 값을 계속 가지고 있습니다.

또한 함수를 재호출할 때 기존 값을 이용하여 계속 참조할 수도 있습니다. 정적변수의 초기화는 처음 함수를 호출할 때 한 번만 실행됩니다.

**예제 파일 scope-05**.php

|  |
| --- |
| <?php  function foo()  {  // 정적 변수를 선언합니다.  **static $bar;**  $bar++;  echo "unset 실행 전: $bar, ";    // 변수를 직접 해제합니다.  unset($bar);  $bar = 21;  echo "unset 실행 후: $bar <br>";  }  foo();  // 결과: unset 실행 전: 1, unset 실행 후: 21    foo();  // 결과: unset 실행 전: 2, unset 실행 후: 21    foo();  // 결과: unset 실행 전: 3, unset 실행 후: 21  ?> |

**결과**

unset 실행 전: 1, unset 실행 후: 21

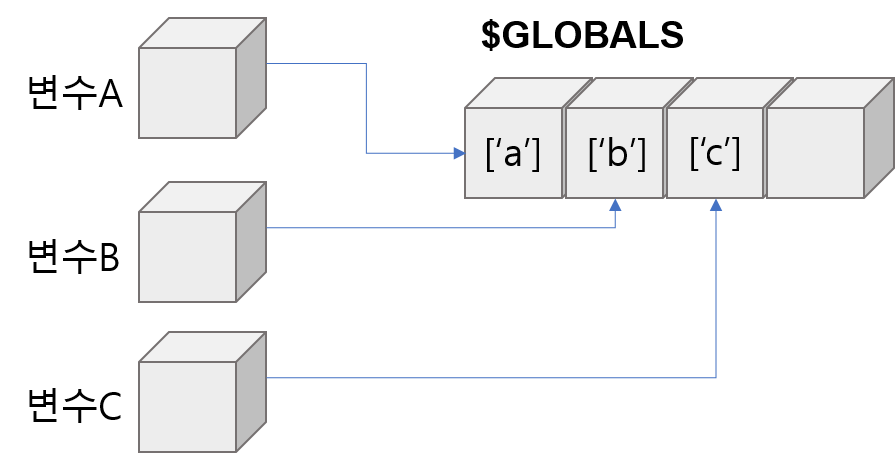
unset 실행 전: 2, unset 실행 후: 21

unset 실행 전: 3, unset 실행 후: 21

**12.4 글로벌 배열**

PHP내에서 생성된 모든 글로벌변수들은 내부에서 슈퍼 배열(supper array)형태로 저장됩니다. 글로벌변수는 이중으로 슈퍼배열과 연결이 되어 있습니다. 변수명을 통하여 접근도 가능하지만 배열로 해당 변수에 데이터를 접근할 수도 있습니다.

**$GLOBALS 예약 변수**명은 배열 객체입니다. PHP에서 생성된 모든 글로벌변수들은 변수의 이름과 값이 배열 형태로 $GLOBAL 변수명에 이중으로 연결되어 있습니다.



즉, $aaa = 10; 은 $GLOBALS[‘aaa’];로도 읽고 쓸 수 있습니다.

**예제 파일 scope-06**.php

|  |
| --- |
| <?php  $aaa = 10;  $bbb = 20;  $ccc = 30;  print\_r($GLOBALS);  ?> |

**결과**

Array ( [\_GET] => Array ( ) [\_POST] => Array ( ) [\_COOKIE] => Array ( ) [\_FILES] => Array ( ) [GLOBALS] => Array \*RECURSION\* **[aaa] => 10 [bbb] => 20 [ccc] => 30** )

위의 예는 글로벌변수의 구조를 출력하는 예입니다. 위의 예는 세 개의 외부 변수를 선언했습니다. 그 후에 글로벌 배열변수의 값을 출력해 봅니다. 위의 예에서 생성한 세 개의 변수명과 값이 $GLOBAL 변수에 추가된 것을 확인할 수 있습니다.

이러한 글로벌변수의 특성을 이용하여 PHP 내부에서 사용하는 외부변수들을 함수 내에서 global 키워드 대신 $GLOBALS[] 배열도 사용할 수 있습니다. 사용 방법은 $GLOBALS['변수명']로 사용 하고자 하는 변수명을 기입을 하시면 됩니다.

다음 예제는 글로벌변수의 접근입니다.

**예제 파일 scope-07**.php

|  |
| --- |
| <?php  $x = 5;  $y = 7;  function test() {  global $x,$y;  $y = $x + $y;  }  test();  echo $y; // outputs 15  ?> |

**결과**

12

먼저 PHP 안에 외부변수 $x와 $y를 생성합니다. 함수 내에서 함수 밖에 미리 선언된 글로벌 변수 $x, $y를 사용할 수 있도록 **global 키워드를 사용하여 정의합니다.** 이처럼 글로벌 키워드를 정의된 외부변수는 함수 내에서도 외부에 있는 변수의 값을 직접 접근하여 사용할 수 있습니다.

하지만 함수 내부에서 외부 변수들을 일일이 정의하고 사용하는 것은 불편한 점도 있을 것입니다. $GLOBAL 슈퍼변수를 통해 직접 접근하여 사용할 수도 있습니다. 다음 예는 위의 예와 기능은 같지만 변수 처리 접근 방법은 서로 다릅니다.

**예제 파일 scope-08**.php

|  |
| --- |
| <?php  $x = 5;  $y = 10;  function test() {  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];  }  test();  echo $y; // outputs 15  ?> |

**결과**

15

위의 예제는 global 키워드를 사용하지 않고 $GLOBALS 슈퍼변수를 사용하여 외부 $x, $y 값을 직접 사용을 할 수 있습니다.

슈퍼변수는 별도의 선언 없이 어디에 서든지 사용을 할 수 있습니다.

**12.5 슈퍼변수**

앞 절에서 약간의 슈퍼변수 $GLOBALS에 대해서 설명을 했습니다. $GLOBAL 변수는 PHP 언어에서 지원하는 다수의 슈퍼변수 중 하나입니다.

PHP언어는 $GLOBALS 이외에 여러 개의 슈퍼변수를 다음과 같이 제공합니다.

PHP 슈퍼 글로벌변수들

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

이러한 슈퍼변수의 개념은 PHP 4.1.0에서 도입된 기능입니다.

슈퍼변수는 PHP에서 미리 정의된 내장변수입니다. 또한 예약된 변수명입니다. 슈퍼변수는 PHP 소스의 특정 위치에 상관이 없이 함수, 클래스 내에서도 언제든지 사용할 수 있습니다.

**12.6 슈퍼변수: $\_SERVER**

$\_SERVER 슈퍼변수는 서버 정보를 저장합니다. 브라우저를 통하여 PHP 스크립트를 실행할 때 서버의 정보를 PHP에서 확인할 수 있도록 제공하는 배열변수입니다.

PHP는 웹 개발 용도로 많이 사용을 합니다. 또한 웹 서버와 언어에 최적화된 환경을 제공하는 의미도 있습니다. 따라서 이러한 $\_SERVER 같은 슈퍼변수들을 제공하고 있습니다.

다음은 $\_SERVER 에서 사용되는 배열 값들에 대한 상세한 설명입니다:

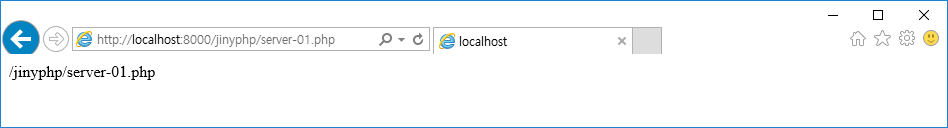
**$\_SERVER['PHP\_SELF']**

PHP\_SELF는 현재 실행하고 있는 PHP 스크립트 파일명을 출력합니다. 이는 HTML폼 ACTION 스크립트 경로를 입력할 때 유용하게 사용을 합니다.

**예제 파일** server-01.php

|  |
| --- |
| <?php  echo $\_SERVER['PHP\_SELF'];  ?> |

**출력 화면**



**$\_SERVER['GATEWAY\_INTERFACE']**

사용 중인 CGI (Common Gateway Interface)의 버전을 반환합니다.

**$\_SERVER['SERVER\_ADDR']**

호스트 서버의 IP주소를 알 수 있습니다.

**$\_SERVER['SERVER\_NAME']**

호스트 서버의 이름(도메인)을 알 수 있습니다.

**$\_SERVER['SERVER\_SOFTWARE']**

서버의 소프트웨어 환경. 서버 식별 문자열 (예: Apache / 2.2.24)을 반환합니다.

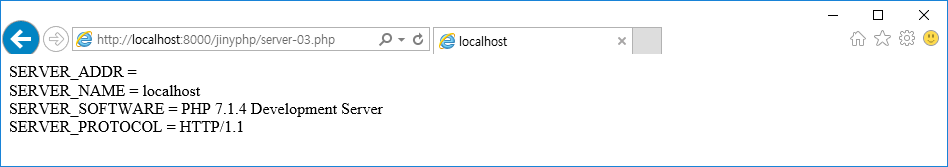
**$\_SERVER['SERVER\_PROTOCOL']**

프로토콜의 이름, 버전, 정보를 반환합니다(예: HTTP / 1.1).

**예제 파일** server-02.php

|  |
| --- |
| <?php  echo "SERVER\_ADDR = ".$\_SERVER['SERVER\_ADDR'];  echo "<br>";  echo "SERVER\_NAME = ".$\_SERVER['SERVER\_NAME'];  echo "<br>";  echo "SERVER\_SOFTWARE = ".$\_SERVER['SERVER\_SOFTWARE'];  echo "<br>";  echo "SERVER\_PROTOCOL = ".$\_SERVER['SERVER\_PROTOCOL'];  ?> |

**출력 화면**



**$\_SERVER['REQUEST\_METHOD']**

페이지 접속 방식을 POST/GET 을 반환합니다.

**$\_SERVER['REQUEST\_TIME']**

접속 타임 스탬프를 반환합니다

**$\_SERVER['REQUEST\_URI']**

현재 페이지 주소에서 도메인을 제외한 값을 반환합니다.

**$\_SERVER['HTTP\_USER\_AGENT']**

사이트에 접속한 클라이언트 프로그램 정보를 반환합니다.

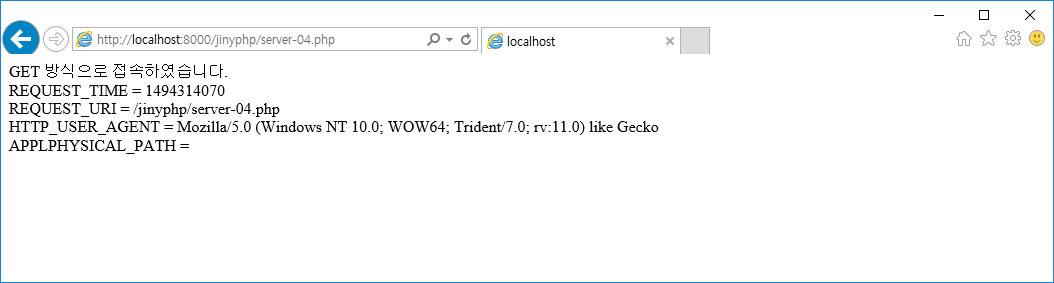
**$\_SERVER['APPLPHYSICAL\_PATH']**

현재 페이지의 실제적인 주소를 반환합니다.

**예제 파일** server-03.php

|  |
| --- |
| <?php  if($\_SERVER['REQUEST\_METHOD'] == "GET"){  echo "GET 방식으로 접속하였습니다.<br>";  } else if($\_SERVER['REQUEST\_METHOD'] == "POST"){  echo "POST 방식으로 접속하였습니다.<br>";  } else {  }  echo "REQUEST\_TIME = ".$\_SERVER['REQUEST\_TIME'];  echo "<br>";  echo "REQUEST\_URI = ".$\_SERVER['REQUEST\_URI'];  echo "<br>";  echo "HTTP\_USER\_AGENT = ".$\_SERVER['HTTP\_USER\_AGENT'];  echo "<br>";  echo "APPLPHYSICAL\_PATH = ".$\_SERVER['APPLPHYSICAL\_PATH'];  echo "<br>";  ?> |

**출력 화면**



**$\_SERVER['QUERY\_STRING']**

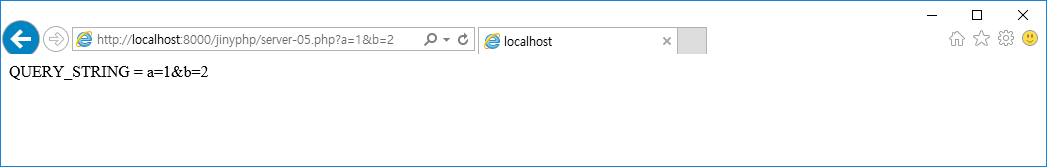
쿼리 문자열을 통해 페이지에 액세스한 경우 쿼리 문자열을 반환합니다.

**예제 파일** server-04.php

|  |
| --- |
| <?php  echo "QUERY\_STRING = ".$\_SERVER['QUERY\_STRING'];  echo "<br>";  ?> |

**출력 화면**

스크립트 다음에 쿼리를 같이 입력합니다. server-05.php?a=1&b=2



**$\_SERVER['HTTP\_ACCEPT']**

현재 요청에 대한 헤더를 반환합니다.

**$\_SERVER['HTTP\_ACCEPT\_CHARSET']**

현재 요청(예: utf-8, ISO-8859-1)에서 Accept\_Charset 헤더를 반환합니다.

**$\_SERVER['HTTP\_ACCEPT\_ENCODING']**

인코딩 방식 예)gzip, deflate

**$\_SERVER['HTTP\_ACCEPT\_LANGAGE']**

언어 예)ko

**$\_SERVER['HTTP\_HOST']**

현재 요청에서 Host 헤더를 반환합니다.

**$\_SERVER['HTTP\_REFERER']**

현재 페이지로 오기 전의 페이지 URL을 반환합니다. <a>, <form> 태그 등으로 넘어올 때 표시됩니다.

**$\_SERVER['SERVER\_ADMIN']**

웹 서버 설정 파일에서 SERVER\_ADMIN 지시어에 주어진 값을 반환합니다. (스크립트가 가상 호스트에서 실행되는 경우 가상 호스트에 대해 정의된 값이 됩니다.)

**$\_SERVER['SERVER\_PORT']**

서버 포트를 확인합니다.

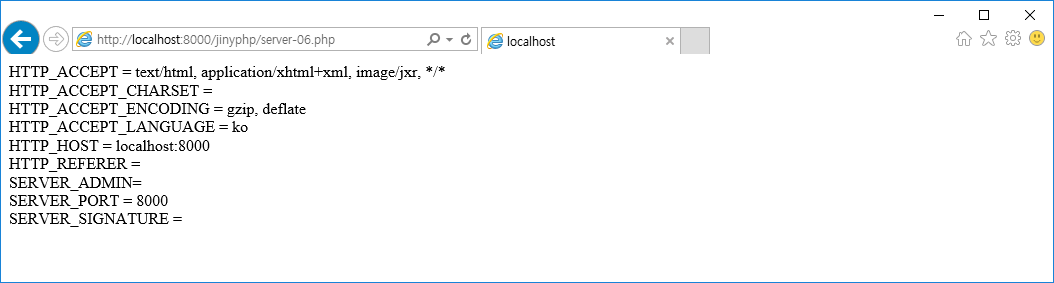
**$\_SERVER['SERVER\_SIGNATURE']**

서버 생성 페이지에 추가되는 서버 버전 및 가상 호스트 이름을 반환합니다.

**예제 파일** server-05.php

|  |
| --- |
| <?php  echo "HTTP\_ACCEPT = ".$\_SERVER['HTTP\_ACCEPT'];  echo "<br>";  echo "HTTP\_ACCEPT\_CHARSET = ".$\_SERVER['HTTP\_ACCEPT\_CHARSET'];  echo "<br>";  echo "HTTP\_ACCEPT\_ENCODING = ".$\_SERVER['HTTP\_ACCEPT\_ENCODING'];  echo "<br>";  echo "HTTP\_ACCEPT\_LANGUAGE = ".$\_SERVER['HTTP\_ACCEPT\_LANGUAGE'];  echo "<br>";  echo "HTTP\_HOST = ".$\_SERVER['HTTP\_HOST'];  echo "<br>";  echo "HTTP\_REFERER = ".$\_SERVER['HTTP\_REFERER'];  echo "<br>";  echo "SERVER\_ADMIN= ".$\_SERVER['SERVER\_ADMIN'];  echo "<br>";  echo "SERVER\_PORT = ".$\_SERVER['SERVER\_PORT'];  echo "<br>";  echo "SERVER\_SIGNATURE = ".$\_SERVER['SERVER\_SIGNATURE'];  echo "<br>";  ?> |

**출력 화면**



**$\_SERVER['HTTPS']**

https 프로토콜 여부를 확인합니다.

**예제 파일** server-06.php

|  |
| --- |
| <?php  if($\_SERVER['HTTPS'] == "on"){  echo "https://";  } else {  echo "http://";  }  ?> |

**$\_SERVER['REMOTE\_ADDR']**

사용자 클라이언트가 접속한 IP 주소를 반환합니다. 예) 192.168.0.3

**$\_SERVER['REMOTE\_HOST']**

사용자 클라이언트가 접속한 호스트명를 반환합니다.

**$\_SERVER['REMOTE\_PORT']**

사용자 클라이언트가 접속한 포트를 반환합니다.

**예제 파일** server-07.php

|  |
| --- |
| <?php  echo "REMOTE\_ADDR = ".$\_SERVER['REMOTE\_ADDR'];  echo "<br>";  echo "REMOTE\_HOST = ".$\_SERVER['REMOTE\_HOST'];  echo "<br>";  echo "REMOTE\_PORT = ".$\_SERVER['REMOTE\_PORT'];  echo "<br>";  echo "REMOTE\_ADDR = ".$\_SERVER['REMOTE\_ADDR'];  echo "<br>";  ?> |

**$\_SERVER['PATH\_TRANSLATED']**

현재 스크립트에 대한 파일 시스템 기반 경로를 반환합니다.

**$\_SERVER['SCRIPT\_FILENAME']**

현재 실행중인 스크립트의 절대 경로 이름을 반환합니다.

**$\_SERVER['SCRIPT\_NAME']**

현재 스크립트의 경로를 반환합니다.

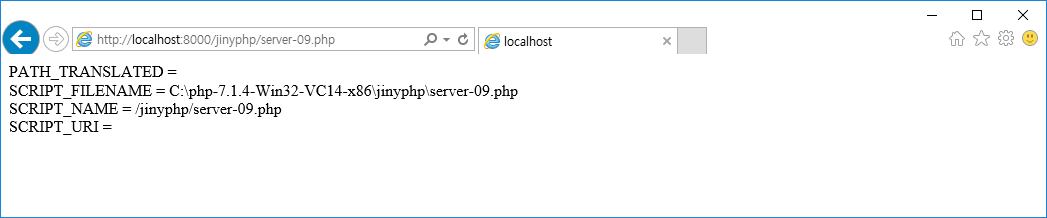
**$\_SERVER['SCRIPT\_URI']**

현재 페이지의 URI를 반환합니다.

**예제 파일** server-08.php

|  |
| --- |
| <?php  echo "PATH\_TRANSLATED = ".$\_SERVER['PATH\_TRANSLATED'];  echo "<br>";  echo "SCRIPT\_FILENAME = ".$\_SERVER['SCRIPT\_FILENAME'];  echo "<br>";  echo "SCRIPT\_NAME = ".$\_SERVER['SCRIPT\_NAME'];  echo "<br>";  echo "SCRIPT\_URI = ".$\_SERVER['SCRIPT\_URI'];  echo "<br>";  ?> |

**출력 화면**



이 책에서 설명하지 못하는 기능들과 슈퍼변수들은 공식 사이트 php.net의 정보를 확인하기를 바랍니다.