Chapter

01

필터

안정적인 프로그램의 동작을 위해서는 정확한 값의 데이터가 중요합니다. 잘못된 데이터들을 프로그램의 예상치 않은 오동작을 발생시킬 수 있습니다.

프로그램 코드를 작성하다 보면 데이터의 정확한 유효성을 체크하는 작업을 자주 합니다. 하지만 데이터를 일정한 패턴으로 검사하는 것은 그렇게 간단하지 않습니다. PHP는 데이터의 유효한 형식을 분석하고 패턴을 검사할 수 있는 다양한 필터 함수들을 제공합니다.

또한 웹 서비스 용도로 많이 사용하는 PHP 언어의 특성상 데이터는 외부 HTML 폼 입력을 통하여도 많이 입력됩니다. 불특정한 다수의 사람들로부터 데이터를 입력받을 때 이러한 데이터의 유효성과 패턴은 더욱더 중요합니다.

일부 악의적인 사용자가 잘못된 입력으로 프로그램의 오류를 발생시키거나, SQL 쿼리 명령들을 섞어 입력함으로써 데이터베이스의 내용을 해킹할 수도 있습니다.

PHP는 제공되는 필터 함수를 통하여 간단하게 데이터의 패턴을 검사하고 유효성을 확인할 수 있습니다. 필터 함수 기능들은 PHP 5.2.0부터는 기본적으로 설치되어 있습니다.

하지만 필터 기능과 동작은 php.ini의 환경 설정에 의해 약간의 영향을 받습니다. ini 설정은 다음과 같습니다.

filter.default

filter.default 환경 설정은 모든 $ _GET, $ _POST, $ _COOKIE, $ _REQUEST 및 $ _SERVER 데이터를 필터링합니다. 필터 이름을 기본적으로 사용합니다.

filter.default_flags

filter.default_flags 환경 설정은 기본 필터로 설정된 경우 적용할 필터를 지정합니다. 이전 버전과의 호환성을 위해 기본적으로 FILTER_FLAG_NO_ENCODE_QUOTES로 설정됩니다.

01.1 필터의 유용성

필터는 데이터의 값을 패턴 형태로 유효성 검사를 합니다. 필터 기능을 자주 적용하는 분야로는 웹에서 데이터 값을 입력받는 분야일 것입니다. 웹 서비스와 같은 응용 서비스는 다양한 타입의 수많은 외부 데이터를 입력받아 처리합니다.

- HTML 폼 입력값
- 쿠키 값
- API 데이터
- 서버 변수들
- 데이터베이스 결과값

위의 같은 값들은 외부로 입력받을 수 있는 데이터의 유형입니다. 이러한 외부 입력과 연관되는 데이터들을 항상 유효성 검사를 하여 사용하는 것을 권장합니다. 유효성 검사 없이 데이터를 바로 사용할 경우에는 악의적인 사용자로부터 입력받은 데이터 값들이 프로그램의 문제를 발생시키거나 해킹에 노출됩니다.

01.1.1 필터 목록

PHP의 필터들은 다양한 형태의 패턴들을 검사할 수 있습니다. 패턴 검사 유형은 필터 이름으로 구분합니다. 내장 함수 filter_list()는 PHP에서 지원하는 내장 필터 목록을 출력합니다.

| 내장 함수 |

```
array filter_list ( void )
```

내장 함수 filter_list()를 사용할 때는 전달되는 입력 매개변수는 없습니다. 내장된 필터의 목록들을 배열 형태로 반환합니다.

또한 내장된 필터명은 개발자들이 알기 쉽게 정의한 이름입니다. 이름들은 내부적으로 처리하는 고유의 ID 값들로 다시 매칭되어 있습니다. 내장 함수 filter_id()는 입력된 필터명에 대해서 고유의 상수 ID 값을 확인할 수 있습니다.

| 내장 함수 |

```
int filter_id ( string $filtername )
```

예제 파일 | **filter-01.php**

```
1  <?php
2     foreach (filter_list() as $id =>$filter) {
3        echo "필터명 =" . $filter . ', ID=' . filter_id($filter) . '<br>';
4     }
5
6   ?>
```

화면 출력

```
필터명 =int, ID=257
필터명 =boolean, ID=258
필터명 =float, ID=259
필터명 =validate_regexp, ID=272
```

```
필터명 =validate_domain, ID=277
필터명 =validate_url, ID=273
필터명 =validate_email, ID=274
필터명 =validate_ip, ID=275
필터명 =validate_mac, ID=276
필터명 =string, ID=513
필터명 =stripped, ID=513
필터명 =encoded, ID=514
필터명 =special_chars, ID=515
필터명 =full_special_chars, ID=522
필터명 =unsafe_raw, ID=516
필터명 =email, ID=517
필터명 =url, ID=518
필터명 =number_int, ID=519
필터명 =number_float, ID=520
필터명 =magic_quotes, ID=521
필터명 =callback, ID=1024
```

위의 실습 코드는 내장된 필터의 목록과 필터명에 대한 고유한 ID 값을 확인하는 예제입니다. 이처럼 다양한 형태의 데이터 유효성을 검사할 수 있는 필터들을 제공합니다.

01.1.2 유효성 검사

내장 함수 filter_var()는 입력된 데이터를 지정한 필터 형태의 패턴으로 데이터 유효성을 검사합니다.

| 내장 함수 |

```
mixed filter_var ( mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options ]]
```

함수의 매개변수 입력값으로 2개의 값을 전달받습니다. 첫 번째 데이터 값은 검사 유형입니다. filter_var() 함수는 지정된 필터로 단일 변수만 필터링합니다.

01.2 필터 상수

PHP는 filter_var() 함수를 통하여 필터링하기 위한 패턴 유형을 상수명으로 제공합니다. 알기 쉬운 패턴 코드들을 상수 이름으로 정의하여 제공하는 것이 코드를 작성하는 측면에서 매우 유용합니다.

01.2.1 정수 확인: FILTER_VALIDATE_INT

내부 필터 함수 filter_var()는 입력되는 데이터 값이 정수형 타입인지를 확인합니다. 함수를 호출할 때 두 번째 인자값으로 필터 유형 FILTER_VALIDATE_INT를 사용합니다. 값의 정수 여부 판단은 논리값으로 반환합니다.

예제 파일 | **filter-02.php**

```
<?php
   $int = 100;

   if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
      echo("정수가 유효합니다");
   } else {
      echo("정수가 아닙니다");
   }

?>
```

화면 출력

```
정수가 유효합니다
```

위의 실험은 입력 변수 $int의 데이터 값이 정수값인지를 필터를 통하여 확인합니다. 만일 정수의 값이면 ture 반환값을 출력합니다.

수학에서 정수는 양수, 0, 음수 세 가지로 구분합니다. filter_var()는 양수와 음수만 정수값을 판별할 수 있습니다. 즉, 0의 값은 정수로 인식하지 못합니다. 따라서 이와 관련된 부가적인 코드를 함께 작성해야 합니다. 정수값 필터 사용 시 === 0 조건과 || 연결을 결합하여 사용하는 것을 권장합니다.

예제 파일 | filter-03.php

```
<?php
   $int = 0;

   if (filter_var($int, FILTER_VALIDATE_INT) === 0 ||
      !filter_var($int, FILTER_VALIDATE_INT) === false) {
      echo("Integer is valid");
   } else {
      echo("Integer is not valid");
   }

?>
```

화면 출력

```
Integer is valid
```

위 실험은 0의 정수값을 같이 판별을 하는 예제 코드입니다.

01.2.2 정수 확인 및 범위

컴퓨터에서 정수값을 사용할 때 표현할 수 있는 최대치의 범위가 있습니다. 또한 프로그램에서 입력의 임계치 등의 범위가 필요할 경우가 있습니다. 이러한 경우에는 FILTER_VALIDATE_INT 값과 같이 세 번째 매개변수를 통하여 범위 값을 함께 전달할 수 있습니다.

배열값을 응용하여 정수값 판별과 값의 범위 여부 내에 있는지 확인할 수 있습니다.

예제 파일 | filter-04.php

```
<?php
   $int = 122;
   $min = 1;
   $max = 200;

   $range = array("options" => array("min_range"=>$min, "max_
   range"=>$max);

```

```
8       if (filter_var($int, FILTER_VALIDATE_INT, $range)) === false) {
9           echo("변수 값이 유효한 범위 내에 있지 않습니다!");
10      } else {
11          echo("변수 값이 유효한 범위 내에 있습니다!");
12      }
13
14  ?>
```

화면 출력

변수 값이 유효한 범위 내에 있습니다!

위의 실험은 필터 함수 filter_var()를 사용하여 입력되는 데이터가 정수 유형인지와 1에서 200 사이의 값을 갖는지를 확인합니다.

그 외 참고하여 같이 사용할 수 있는 필터 상수는 다음과 같습니다.

- FILTER_VALIDATE_BOOLEAN: 논리값을 검사합니다.
- FILTER_VALIDATE_FLOAT: float의 유효성을 검사합니다.
- FILTER_SANITIZE_NUMBER_FLOAT: 숫자, + − 및 .e를 제외한 모든 문자를 제거합니다.
- FILTER_SANITIZE_NUMBER_INT: 숫자와 + −를 제외한 모든 문자를 제거합니다.

01.2.3 IP 주소 유효성: FILTER_VALIDATE_IP

필터 함수 filter_var()는 인터넷 IP 주소의 유형을 검사할 수 있습니다. 필터 상수명으로 FILTER_VALIDATE_IP를 사용하게 되면 입력한 데이터가 IP 주소인지를 검사할 수 있습니다.

예제 파일 | **filter-05.php**

```
1   <?php
2       $ip = "127.0.0.1";
3
4       if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
5           echo("$ip 는 유효한 IP 주소입니다.");
```

```
    } else {
        echo("$ip 는 유효하지 않은 IP 주소입니다.");
    }

?>
```

화면 출력

127.0.0.1는 유효한 IP 주소입니다.

01.2.4 IPv6 주소 유효성

IP에 대해서 IPv6 주소 타입 및 유효성을 검사할 수 있습니다. IPv6 주소를 검사할 때는 FILTER_VALIDATE_IP와 FILTER_FLAG_IPV6 2개의 값을 함께 이용합니다.

예제 파일 | filter-06.php

```
<?php
    $ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

    if (!filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) === false) {
        echo("$ip 는 유효한 IPv6 address");
    } else {
        echo("$ip 는 유효하지 않은 IPv6 address");
    }

?>
```

화면 출력

2001:0db8:85a3:08d3:1319:8a2e:0370:7334는 유효한 IPv6 address

01.2.5 쿼리스트링과 URL 유효성 검사: FILTER_VALIDATE_URL

검사 유형 FILTER_VALIDATE_URL과 FILTER_FLAG_QUERY_REQUIRED 2개의 값을 통하여 URL 유효성과 이후 입력되는 GET 쿼리스트링 데이터 값의 유효성까지 함께 확인할 수 있습니다.

예제 파일 | **filter-07.php**

```
<?php
   $url = "http://www.hojin.io";

   if (!filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_
   REQUIRED) === false) {
      echo("$url 유효한 URL");
   } else {
      echo("$url 유효하지 않은 URL");
   }

?>
```

화면 출력

```
http://www.hojin.io 유효하지 않은 URL
```

다음 실험은 filter_var() 함수를 사용하여 문자열을 삭제합니다. 다음과 같이 문자열에서 모든 HTML 태그와 ASCII 값이 127보다 큰 모든 문자를 제거합니다.

예제 파일 | **filter-08.php**

```
<?php
   $str = "<h1>Hello World!</h1>";

   $aaa = filter_var($str, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_
   HIGH);
   echo $aaa;

?>
```

화면 출력

```
Hello World!
```

01.2.6 이메일 주소 검사

내장 함수 filter_var()를 통하여 이메일의 유효성을 검사할 수 있습니다. 이메일의 유효성을 검사할 때는 필터 상수 FILTER_VALIDATE_EMAIL을 사용합니다.

예제 파일 | **filter-09.php**

```php
<?php
    $email = "infohojin@naver.com";

    // 이메일에 포함된 부정한 글자들은 제거합니다.
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);

    // e-mail 유효성
    if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
        echo("$email is a valid email address");
    } else {
        echo("$email is not a valid email address");
    }

?>
```

화면 출력

```
james.lee@example.com is a valid email address
```

그 외 참고하여 같이 사용할 수 있는 필터 상수는 다음과 같습니다.

- FILTER_SANITIZE_EMAIL: 부정한 글자를 제거
- FILTER_VALIDATE_EMAIL: 전자메일 주소 유효성 검사

01.2.7 문자열 체크: FILTER_SANITIZE_STRING

입력한 값을 재출력하는 기능에서 잘못 입력되는 html 코드들은 오동작을 발생하게 됩니다. 내장 함수 filter_var()를 통하여 입력된 문자열에서 HTML 코드를 제거할 수 있습니다.

예제 파일 | **filter-10.php**

```php
<?php
    $str = "<h1>Hello World!</h1>";
    $newstr = filter_var($str, FILTER_SANITIZE_STRING);
    echo $newstr;
?>
```

화면 출력

```
Hello World!
```

필터 상수 FILTER_SANITIZE_STRIPPED은 FILTER_SANITIZE_STRING의 별칭입니다.

01.2.8 URL 유효성 검사

필터 상수 FILTER_SANITIZE_URL은 입력값의 URL 유효성 검사를 확인할 수 있습니다.

예제 파일 | **filter-11.php**

```php
<?php
    $url = "https://www.hojin.io";

    // Remove all illegal characters from a url
    $url = filter_var($url, FILTER_SANITIZE_URL);

    // Validate url
    if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
        echo("$url is a valid URL");
    } else {
        echo("$url is not a valid URL");
    }

?>
```

화면 출력

```
https://www. hojin.io is a valid URL
```

01.2.9 정규표현

필터 상수 FILTER_VALIDATE_REGEXP는 입력된 데이터의 정규표현식 유효성을 검사합니다.

01.2.10 특수문자

다음과 같은 필터 상수를 통하여 입력되는 데이터의 특수문자들을 처리할 수 있습니다.

- FILTER_SANITIZE_ENCODED: 특수문자를 제거/인코딩합니다.
- FILTER_SANITIZE_SPECIAL_CHARS: 특수문자를 제거합니다.
- FILTER_UNSAFE_RAW: 별도의 작업 없이 선택적으로 특수문자만 제거/인코딩합니다.
- FILTER_SANITIZE_FULL_SPECIAL_CHARS

표현 문자	숫자 표현	문자 표현	설명
–	&#00;–&#08;	–	사용하지 않음
space		–	수평 탭
space
–	줄 삽입
–	&#11;–&#31;	–	사용하지 않음
space		–	여백
!	!	–	느낌표
"	"	"	따옴표
#	#	–	숫자 기호
$	$	–	달러
%	%	–	백분율 기호
&	&	&	Ampersand
'	'	–	작은따옴표
(	(	–	왼쪽 괄호
)	)	–	오른쪽 괄호
*	*	–	아스트릭
+	+	–	더하기 기호
,	,	–	쉼표
–	-	–	Hyphen
.	.	–	마침표
/	/	–	Solidus (slash)

표현 문자	숫자 표현	문자 표현	설명
0 – 9	0–9	–	0부터 9까지
:	&#58;	–	콜론
;	&#59;	–	세미콜론
〈	&#60;	&lt;	보다 작은
=	&#61;	–	등호
〉	&#62;	&gt;	보다 큰
?	&#63;	–	물음표
@	&#64;	–	Commercial at
A – Z	A–Z	–	A부터 Z까지
[	&#91;	–	왼쪽 대괄호
₩	&#92;	–	백슬래시
]	&#93;	–	오른쪽 대괄호
^	&#94;	–	탈자 부호
_	&#95;	–	수평선
`	&#96;	–	Acute accent
a – z	a–z	–	a부터 z까지
{	&#123;	–	왼쪽 중괄호
|	&#124;	–	수직선
}	&#125;	–	오른쪽 중괄호
~	&#126;	–	꼬리표
–	&#127;–Ÿ	–	사용하지 않음
			Non-breaking space
¡	&#161;	&iexcl;	거꾸로된 느낌표
¢	&#162;	&cent;	센트 기호
£	&#163;	&pound;	파운드
¤	&#164;	&curren;	현재 환율
¥	&#165;	&yen;	엔
¦	&#166;	&brvbar;	끊어진 수직선
§	&#167;	&sect;	섹션 기호

표현 문자	숫자 표현	문자 표현	설명
¨	¨	¨	움라우트
©	©	©	저작권
ª	ª	ª	Feminine ordinal
«	«	«	왼쪽 꺾인 괄호
¬	¬	¬	부정
			Soft hyphen
®	®	®	등록상표
&hibar;	¯	¯	Macron accent
°	°	°	Degree sign
±	±	±	Plus or minus
²	²	²	Superscript two
³	³	³	Superscript three
´	´	´	Acute accent
µ	µ	µ	Micro sign (Mu)
¶	¶	¶	문단 기호
·	·	·	Middle dot
¸	¸	¸	Cedilla
¹	¹	¹	Superscript one
º	º	º	Masculine ordinal
»	»	»	오른쪽 꺾인 괄호
¼	¼	¼	4분의 1
½	½	½	2분의 1
¾	¾	¾	4분의 3
¿	¿	¿	거꾸로된 물음표
A	À	À	Capital A, grave accent
A	Á	Á	Capital A, acute accent
A	Â	Â	Capital A, circumflex accent
A	Ã	Ã	Capital A, tilde
A	Ä	Ä	Capital A, dieresis or umlaut mark

표현 문자	숫자 표현	문자 표현	설명
Å	&#197;	&Aring;	Capital A, ring (Angstrom)
Æ	&#198;	&AElig;	Capital AE diphthong (ligature)
Ç	&#199;	&Ccedil;	Capital C, cedilla
È	&#200;	&Egrave;	Capital E, grave accent
É	&#201;	&Eacute;	Capital E, acute accent
Ê	&#202;	&Ecirc;	Capital E, circumflex accent
Ë	&#203;	&Euml;	Capital E, dieresis or umlaut mark
Ì	&#204;	&Igrave;	Capital I, grave accent
Í	&#205;	&Iacute;	Capital I, acute accent
Î	&#206;	&Icirc;	Capital I, circumflex accent
Ï	&#207;	&Iuml;	Capital I, dieresis or umlaut mark
Ð	&#208;	&ETH;	Capital Eth, Icelandic
Ñ	&#209;	&Ntilde;	Capital N, tilde
Ò	&#210;	&Ograve;	Capital O, grave accent
Ó	&#211;	&Oacute;	Capital O, acute accent
Ô	&#212;	&Ocirc;	Capital O, circumflex accent
Õ	&#213;	&Otilde;	Capital O, tilde
Ö	&#214;	&Ouml;	Capital O, dieresis or umlaut mark
×	&#215;	&times;	Multiply sign
Ø	&#216;	&Oslash;	width="130"Capital O, slash
Ù	&#217;	&Ugrave;	Capital U, grave accent
Ú	&#218;	&Uacute;	Capital U, acute accent
Û	&#219;	&Ucirc;	Capital U, circumflex accent
Ü	&#220;	&Uuml;	Capital U, dieresis or umlaut mark
Ý	&#221;	&Yacute;	Capital Y, acute accent
Þ	&#222;	&THORN;	Capital Thorn, Icelandic
ß	&#223;	&szlig;	Small sharp s, German (sz ligature)
à	&#224;	&agrave;	Small a, grave accent
á	&#225;	&aacute;	Small a, acute accent

표현 문자	숫자 표현	문자 표현	설명
a	&#226;	&acirc;	Small a, circumflex accent
a	&#227;	&atilde;	Small a, tilde
a	&#228;	&auml;	Small a, dieresis or umlaut mark
a	&#229;	&aring;	Small a, ring
æ	&#230;	&aelig;	Small ae diphthong (ligature)
c	&#231;	&ccedil;	Small c, cedilla
e	&#232;	&egrave;	Small e, grave accent
e	&#233;	&eacute;	Small e, acute accent
e	&#234;	&ecirc;	Small e, circumflex accent
e	&#235;	&euml;	Small e, dieresis or umlaut mark
i	&#236;	&igrave;	Small i, grave accent
i	&#237;	&iacute;	Small i, acute accent
i	&#238;	&icirc;	Small i, circumflex accent
i	&#239;	&iuml;	Small i, dieresis or umlaut mark
ð	&#240;	&eth;	Small eth, Icelandic
n	&#241;	&ntilde;	Small n, tilde
o	&#242;	&ograve;	Small o, grave accent
o	&#243;	&oacute;	Small o, acute accent
o	&#244;	&ocirc;	Small o, circumflex accent
o	&#245;	&otilde;	Small o, tilde
o	&#246;	&ouml;	Small o, dieresis or umlaut mark
÷	&#247;	&divide;	Division sign
ø	&#248;	&oslash;	Small o, slash
u	&#249;	&ugrave;	Small u, grave accent
u	&#250;	&uacute;	Small u, acute accent
u	&#251;	&ucirc;	Small u, circumflex accent
u	&#252;	&uuml;	Small u, dieresis or umlaut mark
y	&#253;	&yacute;	Small y, acute accent
þ	&#254;	&thorn;	Small thorn, Icelandic

표현 문자	숫자 표현	문자 표현	설명
y	ÿ	ÿ	Small y, dieresis or umlaut mark

01.2.11 함수 적용

다음과 같은 필터 상수들은 함수를 별도의 함수를 적용하거나 호출할 수 있습니다.

- FILTER_SANITIZE_MAGIC_QUOTES: addslashes() 함수를 적용합니다.
- FILTER_CALLBACK: 데이터 필터링을 위한 사용자 정의 함수를 호출합니다.

01.3 필터 함수

PHP는 필터 기능을 좀 더 다양하게 처리하기 위해서 추가로 몇 가지 내장 함수들을 제공합니다. 추가 함수들을 이용하면 복잡한 필터 처리를 보다 쉽게 구현할 수 있습니다.

| 내장 함수 |

```
bool filter_has_var ( int $type , string $variable_name )
```

내장 함수 filter_has_var()는 지정된 유형의 변수가 있는지 확인합니다. 타입의 종류는 다음과 같습니다.

- INPUT_GET
- INPUT_POST
- INPUT_COOKIE
- INPUT_SERVER 또는 INPUT_ENV

예제 파일 | **filter-12.php**

```
1   <?php
```

```
    echo "INPUT_GET = ";
    echo filter_has_var(INPUT_GET, 'aaa') ? 'Yes' : 'No';

?>
```

화면 출력

```
INPUT_GET = Yes
```

위의 실습은 타입 유형의 변수 존재 유무를 확인합니다. filter-12.php?aaa=test 형태로 GET 데이터를 추가하여 스크립트를 호출합니다. GET 방식으로 데이터를 전달과 변수명이 aaa이기 때문에 filter_has_var() 함수는 결과를 '참'으로 출력합니다.

| 내장 함수 |

```
mixed filter_input ( int $type , string $variable_name [, int $filter = FILTER_DEFAULT [,
mixed $options ]] )
```

내장 함수 filter_input()은 양식 입력 등 외부 변수를 읽어와서 선택적으로 필터링합니다.

예제 파일 | **filter-13.php**

```
<?php
    $search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_
    SPECIAL_CHARS);
    $search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_
    ENCODED);

    echo "입력한 검색 키워드는 $search_html 입니다.<br>";
    echo "<a href='?search=$search_url'>재검색</a>";

?>
```

위의 실험은 filter-13.php?search=jiny 형태로 GET 데이터를 추가하여 스크립트를 호출합니다.

| 내장 함수 |

mixed **filter_input_array** (int $type [, mixed $definition [, bool $add_empty = true]])

내장 함수 filter_input_array()는 필터를 처리하기 위한 배열 정보를 이용하여 다수의 외부 변수를 일괄적으로 필터링합니다. 이 함수는 filter_input()을 반복적으로 호출하지 않고도 많은 값을 검색할 때 유용합니다.

예제 파일 | **filter-14.php**

```
1  <?php
2      error_reporting(E_ALL | E_STRICT);
3      $args = array(
4          'name'   => FILTER_SANITIZE_ENCODED,
5          'age' => FILTER_VALIDATE_INT,
6          'country'   => FILTER_SANITIZE_ENCODED
7      );
8      $myinputs = filter_input_array(INPUT_GET, $args);
9      var_dump($myinputs);
10
11 ?>
```

화면 출력

```
array(3) { ["name"]=> string(4) "jiny" ["age"]=> int(18) ["country"]=>
string(5) "korea" }
```

위의 실험은 filter-14.php?name=jiny&age=18&country=korea 형태로 GET 데이터를 추가하여 스크립트를 호출합니다.

| 내장 함수 |

mixed **filter_var_array** (array $data [, mixed $definition [, bool $add_empty = true]])

내부 변수 filter_var_array()는 여러 개의 변수를 필터링합니다. 이 함수는 filter_var()를 반복적으로 호출하지 않고도 많은 값을 검색하는 데 유용합니다. 다음 예제는 공식 사이

트에 있는 예제입니다.

예제 파일 | **filter-15.php**

```
<?php
    error_reporting(E_ALL | E_STRICT);
    $data = array(
        'product_id'    => 'libgd<script>',
        'component'     => '10',
        'versions'      => '2.0.33',
        'testscalar'    => array('2', '23', '10', '12'),
        'testarray'     => '2',
    );

    $args = array(
    'product_id'   => FILTER_SANITIZE_ENCODED,
    'component'    => array('filter'    => FILTER_VALIDATE_INT,
                            'flags'     => FILTER_FORCE_ARRAY,
                            'options'   => array('min_range' => 1, 'max_
                            range' => 10)
                           ),
    'versions'     => FILTER_SANITIZE_ENCODED,
    'doesnotexist' => FILTER_VALIDATE_INT,
    'testscalar'   => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_REQUIRE_SCALAR,
                           ),
    'testarray'    => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_FORCE_ARRAY,
                           )

                      );

    $myinputs = filter_var_array($data, $args);

    var_dump($myinputs);
    echo "\n";

?>
```

화면 출력

```
array(6) { ["product_id"]=> string(17) "libgd%3Cscript%3E" ["component"]=>
array(1) { [0]=> int(10) } ["versions"]=> string(6) "2.0.33" ["doesnotexist"]=>
NULL ["testscalar"]=> bool(false) ["testarray"]=> array(1) { [0]=> int(2) } }
```