Validación de formularios con PHP

Cando un usuario introduce datos nun formulario pode ocorrer que non o faga de xeito adecuado. Por exemplo, pode ocorrer que non cubra todos os campos obrigatorios, ou que introduza nalgún campo valores non válidos. Para non procesar nestes casos os datos recibidos, é necesario que pasen antes por un procedemento de validación.

Comprobar se se recibiu toda a información non opcional

<https://cybmeta.com/isset-is_null-y-empty-diferencias-y-ejemplos-de-uso>

Para comprobar se un campo non se recibiu ou está baleiro, podemos empregar a función empty, que comproba tanto que unha variable exista como que non estea baleira (considera baleira unha variable se contén algún dos valores 0, NULL, FALSE ou a cadea baleira).

A primeira comprobación debería ser se existe algo na variable $\_POST (ou $\_GET, dependendo do método empregado no formulario web). Se está baleira significa que non se recibiu ningún formulario; pode ocorrer se non se accede á páxina que recibe os datos como consecuencia do envío dun formulario (por exemplo porque o usuario tecleara a URL da páxina directamente no seu navegador).

if (empty($\_POST)) exit('Información do formulario non recibida');

A continuación deberemos comprobar que os campos obrigatorios non se deixaron baleiros. Pode facerse por exemplo empregando un procedemento como o seguinte:

// Campos obrigatorios

$obrig = array('nome', 'contrasinal', 'email');

// Comprobamos que ningún dos campos obrigatorios recibidos está baleiro

$erro = false;

foreach($obrig as $campo) {

if (empty($\_POST[$campo])) $erro = true;

}

if ($erro) {

// Non existe algún dos campos obrigatorios

}

Comprobar os tipos de datos da información recibida

O seguinte paso debería ser a comprobación dos tipos de datos recibidos nalgúns controis. Por exemplo, os nomes deberían ser cadeas de texto, e a idade dunha persoa un número enteiro. Para estas comprobacións podemos usar algunhas das funcións de comprobación de tipo, que comezan con 'is\_':

|  |  |
| --- | --- |
| Función de comprobación de tipo |  |
| **is\_array** | Comproba que a variable sexa un array. |
| **is\_​bool** | Comproba que a variable sexa de tipo booleano. |
| **is\_​float, is\_​double, is\_​real** | Comproban que a variable sexa un número real. |
| **is\_​int, is\_integer, is\_​long** | Comproban que a variable sexa un número enteiro. |
| **is\_​numeric** | Comproba que a variable sexa un número ou unha cadea numérica. |
| **is\_scalar** | Comproba que a variable sexa un escalar (enteiro, real, booleano ou cadea de texto). |
| **is\_string** | Comproba que a variable sexa unha cadea de texto. |

Por exemplo:

if (!is\_string($\_POST['nome'])) $erro = true;

Comprobar algúns valores recibidos como datas, URLs ou direccións de email

Nalgúns campos necesitamos facer comprobacións más detalladas. Por exemplo, se queremos que un usuario introduza nun control do formulario web a súa data de nacemento, ou unha dirección de correo, non chega con comprobar que o tipo de datos recibido sexa unha cadea de texto.

Un xeito xenérico de facer comprobacións deste tipo é empregando expresións regulares.

Expresións regulares

PHP ofrece soporte para utilizar expresións regulares ao estilo das que ten a linguaxe Perl por medio da libraría PCRE, que dende a versión 5.3.0 da linguaxe PHP xa non pode ser deshabilitada.

As expresións regulares empréganse xunto con certas funcións para realizar buscas, substitucións ou divisións en cadeas de texto. O patrón que contén a expresión regular encérrase entre delimitadores. O delimitador pode ser calquera carácter alfanumérico menos a barra invertida e o espazo.

Patróns

O patrón dunha expresión regular indica as características que debe cumprir un texto para que se considere válido. Dentro dos patróns, pódense empregar algúns caracteres especiais como os seguintes:

|  |  |
| --- | --- |
| Carácter | Significado |
| **\** | Escapa calquera dos caracteres especias para poder empregalo dentro do patrón perdendo o seu significado. |
| **^** | Fixa o comezo da cadea. |
| **$** | Fixa o final da cadea. |
| **.** | Un carácter calquera. |
| **?** | O carácter anterior é opcional; pode figurar 0 ou 1 veces. |
| **+** | Repite o carácter anterior 1 ou máis veces. |
| **\*** | Repite o caracter anterior 0 ou máis veces. |
| **|** | Separa dúas expresións alternativas. |
| **[ ]** | Abranguen un conxunto ou rango de caracteres. |
| **{ , }** | Repetición do carácter ou rengo de caracteres anterior un número mínimo e/ou máximo de veces. |

Por exemplo:

|  |  |
| --- | --- |
| Patrón | Textos válidos para o patrón |
| **^<b>.\*</b>$** | Calquera texto que comeza coa etiqueta <b> e remata con </b>. |
| **^a|A$** | Letra a minúscula ou A maiúscula. |
| **a|A** | Calquera texto que conteña a letra a minúscula ou letra A maiúscula. |
| **^[oO]?la** | Calquera texto que comece por "ola" ou "Ola". |

A sintaxe completa dos patróns que podemos empregar nas expresións regulares podemos consultala na URL http://www.php.net/manual/es/reference.pcre.pattern.syntax.php.

Funcións

Entre as funcións da linguaxe PHP que soportan expresións regulares temos:

|  |  |
| --- | --- |
| Función |  |
| **preg\_match** | Compara unha cadea de texto con unha expresión regular. |
| **preg\_replace** | Realiza substitucións de expresións regulares en cadeas de texto. |
| **preg\_split** | Divide unha cadea de texto mediante unha expresión regular. |

A función "preg\_match" é a que empregaremos para validar que o texto recibido dun formulario web se corresponde co esperado. Por exemplo, poderiamos empregar o seguinte código para validar un nome de usuario composto soamente por entre 6 e 20 letras e números:

$erro = false;

if (empty($\_POST['usuario'])) {

$erro = true;

}

else {

$usuario = trim($\_POST['usuario']);

$regex = '/^[a-z0-9]{6,20}$/';

if (!preg\_match($regex, $usuario)) {

$erro = true;

}

}

if ($erro) {

// O nome de usuario non é válido

}

Aínda que as expresións regulares son un bo método para validar algunhas cadeas sinxelas, como o nome dun usuario, os patróns a manexar vólvense moi complexos cando se trata de analizar contidos menos sinxelos, como é o caso dunha URL, unha dirección de correo electrónico ou unha data.

No caso particular da validación das datas, podemos empregar a función "checkdate", que recibe o mes, o día e o ano dunha data (por ese orde), e comproba a súa validez. Para separar os compoñentes dunha cadea de texto que represente a unha data, podemos botar man da función "explode".

Por exemplo, supoñendo que o usuario introduza unha data no formato "DD/MM/AAAA", podemos comprobar a súa validez co seguinte código:

$data = explode('/', $\_POST['data'], 3);

if ((count($data) == 3) && checkdate($data[1], $data[0], $data[2])) {

// data válida

}

else {

$erro = true;

}

Para outros contidos existe en PHP unha extensión, Filter, que se inclúe por defecto dende a versión 5.2. A extensión Filter permite, entre outros, validar de xeito sinxelo certos tipos de información de uso común.

Extensión Filter

A extensión Filter emprégase principalmente con dous obxectivos, que se corresponden cos dous tipos principais de filtros que manexa:

* **Filtrado**: permite validar se os datos cumpran certos requisitos, pero non os modifica.
* **Saneamento**: modifica (pero non valida) os datos eliminando certos conxuntos de caracteres non permitidos.

Entre as funcións que inclúe a extensión, a máis empregada para aplicar un filtro a unha expresión é "filter\_var". Indicámoslle a variable que contén a expresión e o filtro a aplicar, e devolve:

* FALSE cando a expresión non é válida ou falla o filtro.
* No caso da comprobación correcta dun filtro de validación, a mesma expresión comprobada.
* No caso da execución correcta dun filtro de saneamento, a expresión modificada.

Validación de expresións

Algúns dos filtros de validación que incorpora a extensión son os seguintes:

|  |  |
| --- | --- |
| Filtro de validación |  |
| **FILTER\_VALIDATE\_EMAIL** | Válido cando a expresión correspóndese cunha dirección de correo electrónico. |
| **FILTER\_VALIDATE\_IP** | Válido cando a expresión correspóndese cunha dirección IP. |
| **FILTER\_VALIDATE\_URL** | Válido cando a expresión correspóndese cunha URL. |

Por exemplo, podemos empregar o seguinte código para validar unha dirección de correo electrónico recibida:

$email = $\_POST['email'];

if (filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

// dirección de email válida

}

else {

$erro = true;

}

Saneamento de expresións

Algúns dos filtros de saneamento que incorpora a extensión son os seguintes:

|  |  |
| --- | --- |
| Filtro de validación |  |
| **FILTER\_SANITIZE\_EMAIL** | Elimina os caracteres que non poden formar parte dunha dirección de correo electrónico. |
| **FILTER\_SANITIZE\_MAGIC\_QUOTES** | Escapa os caracteres comiña simple ('), comiña dobre ("), barra invertida (\) e NUL (o byte NULL), engadíndolles diante unha barra invertida. É semellante a empregar a función addslashes. |
| **FILTER\_SANITIZE\_SPECIAL\_CHARS** | Substitúe os caracteres comiña simple ('), comiña dobre ("), menor (<), maior (>), ampersand (&), e os de código ASCII menor de 32, pola súa entidade HTML equivalente. |
| **FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS** | Semellante ao anterior, e ademais rexeita unha expresión cando detecta un carácter non válido polo conxunto de caracteres empregado (o definido na directiva default\_charset). |
| **FILTER\_SANITIZE\_URL** | Elimina os caracteres que non poden formar parte dunha URL. |

Por exemplo, se queremos sanear un termo introducido polo usuario nun formulario podemos facer:

$busca = $\_POST['email'];

$busca\_san = filter\_var($busca, FILTER\_SANITIZE\_SPECIAL\_CHARS);

Validación dos datos dos formularios con PHP

Unha vez recibido un dato, normalmente é necesario comprobar se o dato é do tipo esperado (número, texto, email, data, etc) para poder procesalo sen erros. Tal como no caso da verificación, a validación de formularios tamén se pode levar a cabo no lado do cliente utilizando Javascript. Neste caso estudarase cómo facer a validación de formularios con PHP. PHP ofrece distintas funcións para levar a cabo as comprobacións necesarias para a validación.

Cómpre ter en conta que algunha das funcións que se van a tratar a continuación son soamente soportadas por versións recentes de PHP, polo tanto, se o código a desenvolver puidese ser implantado en servidores con versións antigas de PHP, debería ser necesario comprobar se o servidor soporta unha función antes de usala. Para isto PHP conta ca función function\_exists(). Véxase un exemplo de uso desta función.

|  |  |  |
| --- | --- | --- |
| Código do control no formulario | Código de exemplo.php | Información recibida |
| <form action="exemplo.php">  <p>Idade:<input type="text" name="idade" />(\*)</p>  <p><input type="submit" value="Submit" /></p>  </form> | <?php  header('Content-Type: text/html; charset=UTF-8');  echo"<pre>";  print\_r($\_REQUEST);  echo"</pre>";  $idade= (isset($\_REQUEST['idade']))  ? $\_REQUEST['idade']  : "";  if ($idade == "")  print "<p>Non se escribiu ningún nome</p>";  else{  if (function\_exists("filter\_var")){  if (filter\_var($idade, FILTER\_VALIDATE\_INT))  print "<p>A idade é o número $idade</p>";  else  print "A idade ten que ser un número enteiro";  }else  print "A función <strong>filter\_var()<strong/> non está dispoñible neste servidor";  }  ?> | Array  (  [idade] => 35 anos  )  A idade ten que ser un número enteiro |
| Array  (  [idade] => 35  )  A idade é o número 35 |

Existe un conxunto de funcións is\_ a partir da versión 4 de PHP e posteriores. Son funcións booleanas que determinan se o valor que se lles pasa como parámetro é dun tipo determinado, devolvendo verdadeiro ou falso. Este grupo de funcións recóllense na seguinte táboa:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Función | Tipo de datos | Alias (función equivalente) |
| existencia | isset($valor) | Definida |  |
| is\_null($valor) | NULL |  |
| números | is\_bool($valor) | Booleano |  |
| is\_numeric($valor) | Número (pode ter signo, parte decimal, e pode estar expresado en notación decimal, exponencial ou hexadecimal). |  |
| is\_int($valor) | Enteiro | is\_integer($valor)  is\_long($valor) |
| is­\_float($valor) | Float | is\_double($valor)  is­\_real($valor) |
| cadeas | is\_string($valor) | Cadea |  |
| outros | is\_scalar($valor) | Escalar (entero, float, cadea ou booleano) |  |
| is­\_array($valor) | Array |  |
| is\_callable($valor) | Función |  |
| is­\_object($valor) | Obxecto |  |
| is­\_resource($valor) | Recurso |  |

Estas funcións comproban o tipo para valores de variables independentemente da súa orixe, por exemplo poden ser valores almacenados nun array. Cando os valores veñen dun formulario, estes son sempre cadeas, polo que as funcións is\_int() e is\_float() devolverán sempre false. No caso dos formularios, usarase a función is\_numeric() para comprobar se o dato é un número.

No exemplo que segue amósase o uso dalgunhas destas funcións.

|  |  |  |
| --- | --- | --- |
| Código do control no formulario | Código de exemplo.php | Información recibida |
| <form action="exemplo.php">  <p>Estatura:<input type="text" name="estatura" /></p>  <p><input type="submit" value="Submit" /></p>  </form> | <?php  header('Content-Type: text/html; charset=UTF-8');  echo"<pre>";  print\_r($\_REQUEST);  echo"</pre>";  $estatura= (isset($\_REQUEST['estatura']))  ? $\_REQUEST['estatura']  : "";  if (is\_numeric($estatura))  print "<p>Escribiuse o número $estatura.</p>";  else  print "<p>Non se escribiu ningún número no campo estatura.</p>";  ?> | Array  (  [estatura] => 1X70  )  Non se escribiu ningún número no campo estatura. |
| Array  (  [estatura] => 1.70  )  Escribiuse o número 1.70 |

Cando en algún dos campos se introduce unha data, para poder validala, unha forma sinxela é mediante o uso de dúas funcións: explode() e checkdate(). Supoñendo unha data almacenada nunha cadea, cun formato determinado dd/mm/aaaa (debería informarse deste formato ao usuario no formulario para que introduza así a data), aplicaráselle a función explode indicando / como carácter de separación, obtendo así un array de tres elementos: día, mes e ano. Comprobarase que o devandito array ten tres elementos, e a continuación usarase a función checkdate() pasándolle como parámetros o mes, o día e o ano (nesa orde). Desta maneira, a función checkdate devolverá verdadeiro se a data é válida, e falso no caso contrario. Adicionalmente, poderíanse facer outras comprobacións se, por exemplo, desexárase que o ano fose posterior ao 2000. O seguinte exemplo ilustra este proceso de validación de datas.

|  |  |  |
| --- | --- | --- |
| Código do control no formulario | Código de exemplo.php | Información recibida |
| <form action="exemplo.php">  <p>Data de nacemento:<input type="text" name="dataNacemento" />dd/mm/aaaa</p>  <p><input type="submit" value="Submit" /></p>  </form> | <?php  header('Content-Type: text/html; charset=UTF-8');  echo"<pre>";  print\_r($\_REQUEST);  echo"</pre>";  $dataNacemento= (isset($\_REQUEST['dataNacemento']))  ? $\_REQUEST['dataNacemento']  : "";  if ($dataNacemento!=""){  $dataArray=explode("/", $dataNacemento);  if (count($dataArray)==3){  if(checkdate($dataArray[1], $dataArray[0], $dataArray[2]))  print "<p>Escribiuse a data de nacemento $dataNacemento.</p>";  else  print "<p>Algún dos valores día, mes e ano non é válido.</p>";  }else  print "<p>A data introducida non ten un formato válido.</p>";  }else  print "<p>Non se escribiu ningunha no campo data de nacemento.</p>";  ?> | Array  (  [dataNacemento] => 24/05/2013  )  Escribiuse a data de nacemento 24/05/2013. |
|  | Array  (  [dataNacemento] => 24-05-2013  )  A data introducida non ten un formato válido. |
| Array  (  [dataNacemento] => 24/17/2013  )  Algún dos valores día, mes e ano non é válido. |