

UD3 - Programación baseada en linguaxes de marcas con código embebido



Características da linguaxe PHP (II)



Índice

3.10 Validacóns

3.11 Envío de arquivos en formularios

3.12 Seguridade en formularios

3.13 Ficheiros



10

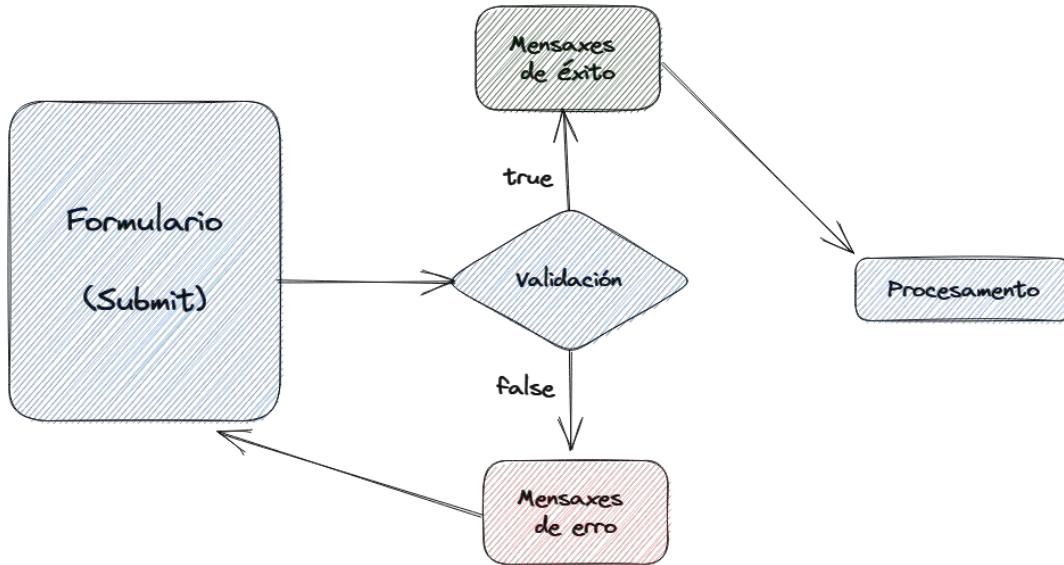
Validaciones



3.10 Validacións



- A **validación no lado de servidor** é algo **obrigatorio** xa que os datos que nos envía o cliente poden vir incorrectos ou mesmo pode alteralos de forma malintencionada, mesmo cando a páxina inclúe validacións en HTML ou Javascript.
- Deste xeito cando unha páxina php recibe datos ten que **validalos ou comprobar que sexan correctos**.
- Se por **exemplo** recibe un campo nome, ten que comprobar que só leve letras, nada de números ou caracteres especiais e se é obligatorio que non veña baleiro.





3.10 Validacións



- PHP dispón de **funcións que axudan a realizar estas validacións**, por exemplo:
 - `empty()` e `isset()`. Comproban se unha variable está baleira ou non definida. No caso de `isset()` sería o equivalente a comprobar que o dato non vén nulo, polo que ainda que haberá que realizar outras comprobacións cos datos recibidos.
 - `die()` e `exit()`. Amosan unha mensaxe, por norma xeral de erro e finalizan o script. Para errores fatais, por exemplo.
 - `trim()`. Elimina os caracteres en branco ao principio e fin dun String.
 - `addslashes()` e `stripslashes()`. Engade ou elimina as barras \ para escapar ' " \ etc.
 - `htmlspecialchars()`, `htmlentities()` e `strip_tags()`. Limpeza de código nos datos de entrada.
- No caso de que haxa algún **erro nos datos**, habería que **indicalo por pantalla** para que o usuario tivera coñecemento do mesmo e puidera solucionalo, volvendo a amosarlle o formulario.
- Idealmente, **os datos do formulario deberían aparecer cubertos de novo**, para que o usuario só tivera que arranxar os que estean mal.



3.10 Validaciones



Característica	<code>htmlspecialchars()</code>	<code>htmlentities()</code>
Alcance da conversión	Só converte caracteres especiais básicos (ex.: &, <, >, " , ')	Converte todos os caracteres con entidade HTML (ex.: &, <, >, " , ' , á , ñ , € , etc.)
Velocidade	Máis rápida (converte menos caracteres)	Máis lenta (converte máis caracteres)
Uso recomendado	Evitar inxeccións de código sen modificar o contido textual máis amplo	Cando precisas converter moitos caracteres especiais, non só os perigosos para HTML
Caract. que converte	& → & ; , < → < ; , > → > ; , " → " ; , ' → ' ; (con ENT_QUOTES)	Todos os anteriores más calquera carácter con entidade HTML, ex.: á → á , ñ → ñ , € → €
Caract. que non converte	Letras acentuadas (á , é , etc.), ñ , símbolos como € , © , §	Non aplica, xa que <code>htmlentities()</code> os converte todos



```
$texto = 'Ola & benvido á <a href="test">Mansión Spencer</a>';

echo htmlspecialchars($texto, ENT_QUOTES);

// Ola & benvido á <a href="test">Mansión Spencer</a>;
```



3.10 Validaciones



Formulario de exemplo con
autocompletado e erros
personalizados

```
<html>
<style>.erro { color: red; } </style>
<body>
<?php
    header('Content-Type: text/html; charset=utf-8');
    // Validação previa: Guarda os valores se o formulário
    $nome = isset($_POST['nome']) ? trim($_POST['nome']) : null;
    $email = isset($_POST['email']) ? trim($_POST['email']) : null;
?>
<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    <label> Nome (*): </label>
    <input type="text" name="nome" id="nome" value=<?php isset($nome) ? print $nome : ""; ?>">
    <?php // Se o campo está vazio amosa o erro
        if(isset($_POST["enviar"]) && empty($_POST["nome"])){
            echo '<span class="erro">Nome obrigatório</span>';
        }
    ?>
    <br><br><label> Email (*): </label>
    <input type="email" name="email" id="email" value=<?php isset($email) ? print $email : ""; ?>">
    <?php // Se o campo está vazio amosa o erro
        if(isset($_POST["enviar"]) && empty($_POST["email"])){
            echo '<span class="erro">Email obrigatório</span>';
        }
    ?>
    <br><br><input type="submit" value="Enviar" name="enviar" id="enviar">
</form>
</body>
</html>
```



3.10 Validacóns



Formulario de exemplo con lista de errores

```
<html>
<style>.erro { color: red; } </style>
<body>
<?php
    // Validación previa se o formulario foi enviado
    if(isset($_POST["enviar"])){
        if(empty($_POST["nome"])){
            $erros[] = "O nome é obrigatorio";
        }
        // O email é obligatorio e debe ter un formato adecuado
        if(!filter_var($_POST["email"], FILTER_VALIDATE_EMAIL) || empty($_POST["email"])){
            $erros[] = "Non se indicou o email ou non ten un formato correcto";
        }
        // Se o array $erros está baleiro, acéptase a validación e corrixense os filtros
        if(empty($erros)){
            $nome = $_POST["nome"];
            $email = filter_var($_POST["email"], FILTER_SANITIZE_EMAIL);
            echo "<p>TODO BEN.</p>";
        }
    }
?>
<ul>
<?php // Amosa a listxe de errores atopados
if(isset($erros)){
    foreach ($erros as $erro){
        echo "<li class='erro'>" . $erro . "</li>";
    }
}
?>
</ul>
<form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    <label> Nome*: </label>
    <input type="text" name="nome" id="nome" value=<?php isset($nome) ? print $nome : "" ; ?>>
    <br><br><label> Email*: </label>
    <input type="email" name="email" id="email" value=<?php isset($email) ? print $email : "" ; ?>>
    <br><br><input type="submit" value="Enviar" name="enviar" id="enviar">
</form>
</body>
</html>
```



3.10 Validacóns

```
<!DOCTYPE html>
<html>
<head>
<title>Iniciar Sesión</title>
</head>
<body>
    <h1>Iniciar Sesión</h1>

    <?php
    // Amosamos a mensaxe de erro se existe
    if (!empty($mensaxe_erro)) {
        echo "<p>$mensaxe_erro</p>";
    }
    ?>
    <form method="post" action="">
        <label for="usuario">Nome de Usuario:</label>
        <input type="text" name="usuario" required
               value="<?php echo htmlspecialchars($usuario); ?>">
        <br>
        <label for="contrasinal">Contrasinal:</label>
        <input type="password" name="contrasinal" required>
        <br>
        <input type="submit" value="Iniciar Sesión">
    </form>
</body>
</html>
```

Formulario de login de exemplo con errores personalizados

```
<?php
// Definimos as credenciais válidas
$usuario_valido = "admin";
$contrasinal_valido = "admin";

// Inicializamos variables
$mensaxe_erro = "";
$usuario = "";
$contrasinal = "";

// Verificamos se o formulario foi enviado
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtemos os datos do formulario e aplicamos trim para quitar espazos en branco
    $usuario = trim($_POST["usuario"]);
    $contrasinal = trim($_POST["contrasinal"]);

    // Validamos que o usuario e o contrasinal sexan obligatorios
    if (empty($usuario) || empty($contrasinal)) {
        $mensaxe_erro = "Ambos os campos son obligatorios.";
    } else {
        // Validamos o nome de usuario (só letras, números, - e _)
        if (!preg_match("/^([a-zA-Z0-9_-])+$/", $usuario)) {
            $mensaxe_erro = "Nome de usuario non válido.";
        } elseif (strlen($contrasinal) < 8) {
            // Validamos que o contrasinal teña polo menos 8 caracteres
            $mensaxe_erro = "O contrasinal debe ter polo menos 8 caracteres.";
        } else {
            // Validamos o nome de usuario e o contrasinal
            if ($usuario === $usuario_valido && $contrasinal === $contrasinal_valido) {
                // As credenciais son válidas, rediriximos o usuario á páxina de inicio
                header("Location: inicio.php");
                exit();
            } else {
                // As credenciais son inválidas, mostramos unha mensaxe de erro
                $mensaxe_erro = "Nome de usuario ou contrasinal incorrectos.";
            }
        }
    }
}
?>
```



3.10 Validacións



Función filter_var() validar / corrixir campos (I)

- Para facilitar a validación de formularios PHP dispón dunha librería que permite validar os campos tendo en conta as comprobacións más habituais. A función **filter_var()** vai permitir, que indicándolle unha variable e un **filtro** aplique ese filtro sobre a variable. No caso de que a variable non cumpla as condicións do ciclo, devolverá **FALSE**.
- Por exemplo, o seguinte código valida se a variable **\$email** cumple co filtro de correo electrónico válido (**FILTER_VALIDATE_EMAIL**).

```
<?php

$email = "jon( . )snow@gardanoite///.com";

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "$email é un correo válido ";
} else {
    echo "$email non é un correo válido ";
}

?>
```

Saída: Indicará que non é válido xa que inclúe caracteres non permitidos nun email.



3.10 Validacións



Función filter_var() validar / corrixir campos (II)

- Os **filtros de validación** más habituais son:
 - **FILTER_VALIDATE_BOOLEAN** → Valida a variable coma un booleano.
 - **FILTER_VALIDATE_EMAIL** → Valida que a variable teña o formato dun correo electrónico válido a@a.com.
 - **FILTER_VALIDATE_FLOAT** → Valida que a variable sexa de tipo float.
 - **FILTER_VALIDATE_INT** → Valida a variable sexa un número enteiro.
 - **FILTER_VALIDATE_IP** → Valida a variable teña o formato dun enderezo IP: 0.0.0.0.
 - **FILTER_VALIDATE_REGEXP** → Valida a variable contra unha expresión regular enviada na variable de opcións. Funciona de maneira análoga a [preg_match\(\)](#).
 - **FILTER_VALIDATE_URL** → Valida que teña o formato dunha URL: <http://www.dominio.com>



3.10 Validacións



Función filter_var() validar / corrixir campos (III)

- Os **filtros de corrección** modifican o valor do campo para eliminar malformacións que puidera haber. Se o conseguén dise que sanean a variable e senón devolven erro. Por exemplo:

```
● ● ●  
<?php  
$email = "jon( . )snow@gardanoite///.com";  
  
if (filter_var($email, FILTER_SANITIZE_EMAIL)) {  
    echo("$email é un correo válido <br>");  
    // E ademais actualiza a variable para que quede saneada  
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);  
} else {  
    echo("$email non é un correo válido <br>");  
}  
?>
```

Saída: \$email quedaría coa cadea “jon.snow@gardanoite.com”



Precaución: Pode ser que a corrección non sexa tampouco correcta.



3.10 Validacións



Función filter_var() validar / corrixir campos (IV)

- Os **filtros de corrección** máis habituais son:
 - **FILTER_SANITIZE_EMAIL** → Elimina todos os caracteres pertenzan ao formato dun email.
 - **FILTER_SANITIZE_ENCODED** → Codifica a cadea coma unha URL válida.
 - **FILTER_SANITIZE_MAGIC_QUOTES** → Aplica a a función `addslashes()` (pon barras \ nas comillas).
 - **FILTER_SANITIZE_NUMBER_FLOAT** → Elimina todos os caracteres excepto números, e os símbolos + e -.
 - **FILTER_SANITIZE_NUMBER_INT** → Elimina tódolos caracteres excepto números e os signos + -.
 - **FILTER_SANITIZE_SPECIAL_CHARS** → Escapa caracteres HTML e caracteres con ASCII menor a 32 (caracteres HTML á ...).
 - **FILTER_SANITIZE_URL** → Elimina todos os caracteres excepto números, letras e \$-_._+!*'(),{}|\\"^~-[]`>#%";/?:@&=
 - **FILTER_SANITIZE_STRING** → Elimina etiquetas, opcionalmente elimina ou codifica caracteres especiais. Pode levar máis parámetros para definir áinda máis o formato do string.



3.10 Validacións



Función filter_var() validar / corrixir campos (V)

- Algúns exemplos de como aplicar os filtros de corrección serían por exemplo, eliminar etiquetas HTML (válidas ou non) dun String:

```
● ● ●  
  
<?php  
$cadea = "<p><xml>Han matado a Kenny... <br>";  
$cadea = filter_var($cadea, FILTER_SANITIZE_STRING);  
echo $cadea;  
?>
```

Eliminaría as etiquetas da cadea: `<p> <xml>` e `
`

- Para corrixir por exemplo un número con decimais:

```
● ● ●  
  
<?php  
$num = "3.14dieciseis";  
$num = filter_var($num, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRACTION);  
echo $num . "<br><br>";  
?>
```

Neste caso no string viría un número con decimais seguido de letras. O filtro `FILTER_SANITIZE_NUMBER_FLOAT` corrixe a cadea e deixaría 314 (só os números) pero se queremos que indique tamén os decimais, pódese engadir o Flag ou parámetro opcional `FILTER_FLAG_ALLOW_FRACTION` co que devolvería 3.14.



3.10 Validacións



Expresións regulares (I)

Unha **expresión regular ou regex** é un **patrón** que se compara cunha cadea ou String de esquerda a dereita, carácter a carácter. A biblioteca PCRE (Perl Compatible Regular Expressions) é unha extensión incorporada en PHP que permite utilizar expresións regulares en funcións para buscar, comparar e substituír cadeas de texto en PHP. As expresións regulares van encerradas en **delimitadores**, que son calquera par de caracteres non alfanuméricos (excepto a barra invertida \ e o espazo en branco). Os delimitadores más utilizados son as barras (/). Por exemplo:

```
/([A-Z])\w+/  
/abc/
```

A función **preg_match()** pode empregarse para **buscar por patróns**, devolverá 1 (true) se o atopa, e 0 (false) se non:

```
$abecedario = "abcdefghijklmnñopqrstuvwxyz";
// ¿Está a cadea "abc" dentro da cadea da variable $abecedario?
echo preg_match("/abc/", $abecedario); // Devuelve 1
```

Recomendación: Para este tipo de operacións sería máis recomendable utilizar strpos() ou strstr(), que son máis rápidas.



3.10 Validacións

Expresións regulares (II)



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 carácter anterior 0 ou máis veces.
	Separa dúas expresións alternativas.
[]	Abranguen un conxunto ou rango de caracteres.
{,}	Repetición do carácter ou rango de caracteres anterior un número mínimo e/ou máximo de veces.



3.10 Validaciones

[A-Z]* @ [A-Z]*

RegExp

Expresiones regulares (III)

Por exemplo, os corchetes representan un conxunto de caracteres:

```
$string = "Opa";
echo preg_match("/Op[aeiou]/", $string); // Devolve 1
// Tamén coincidiría con Opa, Ope, Opi, Opo y Opu
```

O metacarácter *, serán cero ou máis ocorrencias dese carácter:

```
$string1 = "pa";
$string2 = "ola";
$string3 = "oppppppa";
$string4 = "opa";
echo preg_match("/op*a/", $string1); // Devolve 0
echo preg_match("/op*a/", $string2); // Devolve 0
echo preg_match("/op*a/", $string3); // Devolve 1
echo preg_match("/op*a/", $string4); // Devolve 1
```



3.10 Validacións



Expresións regulares (IV)

¿Como validar? Aínda que definir estes patróns resulta algo complexo ou tedioso, temos que ter en conta que na maioría de casos precisaremos cadeas básicas xa definidas. A función `preg_match()` compara unha cadea cunha expresión regular, coma por exemplo:

```
<?php
$patron_texto = "/^[\a-zA-ZáéíóúÁÉÍÓÚäëöüÄÉÍÖÜàèòùÀÈÍÒÙ\s]+$/";
// Só letras da a á z, permitindo maiúsculas e acentos

if (!preg_match("/^[\a-zA-Z]+/", $usuario)) {
    $erro_usuario = "Só se permiten letras como nome de usuario <br>";
}
if (!preg_match('/^968]\d{8}$/', $telefono)) {
    $erro_telefono = "O teléfono debe ter 9 cifras e comenzar por 9, 6 ou 8";
}
if (!preg_match('/^[\A-ZÁÉÍÓÚ][\a-zA-Záéíóú]*$/i', $nombre)) {
    $erro_nombre = "O nome debería comenzar por mayúscula";
}
if (!preg_match('/[-0-9a-zA-Z.+_]+@[-0-9a-zA-Z.+_]+.[a-zA-Z]{2,4}/i', $correo)) {
    $erro_correo = "O formato do email é nome@dominio.extension";
}

?>
```

Nota: En webs como [Regex101](#), [Regexpr](#) pódense validar os patróns creados.



3.10 Validacións



Expresións regulares (V)

Cóbrete lembrar que estas expresións regulares as podemos incluir como **validacións dos campos HTML** de maneira concuxa ás do servidor. **SEMPRE hai que validar no servidor**, entón o feito de incluílas no HTML tan só é para **mellorar a Usabilidade**.

```
● ● ●

<form method="post" action="script.php">
    Nome:
    <input type="text" name="usuario" pattern="a.+" required>
    <br>
    <input type="submit" value="Enviar">
</form>
```

A regexp (expresión regular) `/a.+/` indica que debe escribirse un nome de usuario que comece por a e teña un (ou máis caracteres) a continuación. No seguinte exemplo defínese o tamaño 5-40 caracteres e que só pode levar letras e números:

```
● ● ●

<form method="post" action="script.php">
    Nome: <input type="text" name="username" placeholder="ManzDev" required pattern="[A-Za-z0-9]{5,40}">
    <br> <input type="submit" value="Enviar">
</form>
```



3.10 Validacións



Expresións regulares (VI)

O más habitual será filtrar o tamaño e tipo de caracteres en cada campo. Deste xeito teremos que definir patróns que identifiquen o comezo da cadea e o final e leven un conxunto de caracteres definido xunto co tamaño do campo. Por exemplo, se queremos definir un campo de texto ou validar o valor dunha variable que admitan todas as letras (maiúsculas e minúsculas), incluídas tildes, diéreses, ñ, ç, ou símbolos como ' ou \$, cun tamaño entre 5 e 30 caracteres, faríamos algo así:

```
/^[caracteres_permitidos]{tamaño_mínimo, tamaño_máximo}$/
```

```
/^[\a-zA-ZñÑç]{5,30}$/
```

```
<?php

$cadea = "aá'ab9ÁÑ.\$ç";

if (!preg_match('/^[\a-zA-ZñÑç]{5,30}$/, $cadea))
    echo "A cadea $cadea NON É VALIDA.<br>";
else echo "A cadea $cadea É VALIDA. <br>"
?>
```



3.10 Validaciones

[A-Z]* @ [A-Z]*

RegExp

Expresiones regulares (VII)



```
(?:[a-z0-9!#$%&'*+/=?^`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^`{|}~-]+)*|(?:\[x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]\|\|[x01-\x09\x0b\x0c\x0e-\x7f])*)(?:([a-z0-9]([a-z0-9-]*[a-z0-9])?\.+[a-z0-9]([a-z0-9-]*[a-z0-9])?|[\[(?:[25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?)]\.){3}([25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]):([x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]\|\|[x01-\x09\x0b\x0c\x0e-\x7f]))+)\]
```

HOW TO REGEX

STEP 1: OPEN YOUR FAVORITE EDITOR

@GARABATOKID

STEP 2: LET YOUR CAT PLAY ON YOUR KEYBOARD

DAY1 OF PROGRAMMING

Google



10 YEARS OF PROGRAMMING

Google





3.10 Validaciones

Volvendo a completar un formulario

Cando exista algún erro ao validar o formulario, habería que darlle ao usuario a opción de corrixilo sen perder a información enviada. Para iso temos que xerar o formulario en HTML de novo, pero modificando as súas propiedades. Por exemplo poñendo nos `value` o valor enviado, e engadindo `checked` ou `selected` nos `checkbox` e `select` respectivamente.

```
<?php
if (!empty($_POST['modulos']) && !empty($_POST['nombre']) && !empty($_POST['opcion'])) {
    // Aquí iría o código a executar cando os datos son correctos
} else {
    // Xeramos o formulario dinamicamente
    $nome = $_POST['nome'] ?? "";
    $modulos = $_POST['modulos'] ?? [];
    $opcion = $_POST['opcion'] ?? "";
    ?>
<form action=<?php echo $_SERVER['PHP_SELF'];?>" method="POST">
    <p><label for="nome">Nome do alumno:</label>
        <input type="text" name="nome" id="nome" value=<?php echo $nome ?>" />
    </p>
    <p>
        <label>Selecciona unha opción:</label>
        <select name="opcion">
            <option value="opcion1" <?php if ($opcion === "opcion1") echo 'selected'; ?>>Opción 1</option>
            <option value="opcion2" <?php if ($opcion === "opcion2") echo 'selected'; ?>>Opción 2</option>
        </select>
    </p>
    <p><input type="checkbox" name="modulos[]" id="modulosDWCS" value="DWCS"
        <?php if(in_array("DWCS",$modulos)) echo 'checked'; ?> />
        <label for="modulosDWCS">DWCS</label>
    </p>
    <p><input type="checkbox" name="modulos[]" id="modulosDWCC" value="DWCC"
        <?php if(in_array("DWCC",$modulos)) echo 'checked'; ?> />
        <label for="modulosDWCC">DWCC</label>
    </p>
    <input type="submit" value="Enviar" name="enviar"/>
</form>
<?php } ?>
```



3.10 Validacións



```
<?php
function rexistro() {
    if (!empty($_POST)) {
        $mensaxe = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^([a-z\d]{2,64})$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['mensaxe'] = "Usuario rexistrado, pode loguearse.";
                                                header("Location: login.php");
                                                exit;
                                            } else $mensaxe = "Debe indicar un email válido";
                                        } else $mensaxe = "O email debe ter menos de 64 caracteres";
                                    } else $mensaxe = "O email non pode estar baleiro";
                                } else $mensaxe = "O nome de usuario xa existe";
                            } else $mensaxe = "No nome de usuario só se permiten caracteres: a-z, A-Z e 0-9";
                        } else $mensaxe = "O nome de usuario debe ter entre 2 e 64 caracteres";
                    } else $mensaxe = "O contrasinal debe ter coma mínimo 6 caracteres";
                } else $mensaxe = "Os contrasinais non coinciden";
            } else $mensaxe = "Contrasinal baleiro";
        } else $mensaxe = "Nome de usuario baleiro";
        $_SESSION['mensaxe'] = $mensaxe;
    }
    header("Location: rexistro.php");
    exit;
}
```

Formulario de rexistro de exemplo con validacións personalizadas

11

Envío de arquivos en formularios



3.11 Envío de archivos en formularios



- Para enviar archivos empregando un formulario cómpre facer unha serie de **configuracións previas no servidor**:
 - Revisar os límites de envío de archivos no [php.ini](#) para ver que se axusten ao tamaño co que imos traballar.
 - Comprobar que a **directiva file_uploads** estea establecida a [On](#).
 - Dar **permisos** de escritura ao Apache no cartafol onde imos subir os archivos. Se por exemplo imos subilos ao cartafol: [/var/www/html/arquivos](#), no contorno de desenvolvemento (local) podemos simplemente facer algo así:

```
$ sudo chown -R www-data:www-data /var/www/html/arquivos
```

```
$ sudo chmod 755 /var/www/html/arquivos
```

- Para permitir o envío de archivos cómpre indicar no código HTML:

- Na etiqueta `<form>`, o **atributo enctype="multipart/form-data"**.
- Dentro dese formulario engadir un campo `<input>` de tipo “file”.

← → ⌂ http://localhost/DWCS/exemploUpload.php

Subir Arquivos

Escolle un arquivo: Ninguno arquivo selec.



3.11 Envío de arquivos en formularios



```
<!DOCTYPE html>
<html>
<head>
    <title>Subir Arquivos</title>
</head>
<body>
    <h1>Subir Arquivos</h1>

    <form action="procesar_carga.php" method="post" enctype="multipart/form-data">

        <label for="arquivo">Escolle un arquivo:</label>
        <input type="file" name="arquivo" id="arquivo">
        <br>
        <input type="submit" value="Cargar Arquivo">

    </form>
</body>
</html>
```



3.11 Envío de arquivos en formularios



- A información sobre un arquivo subido array asociativo multidimensional **`$_FILES`**.
- Este array créase na primeira dimensión coa clave asociativa que se indica como `id/name` no input do formulario, no anterior exemplo sería “[arquivo](#)”. No caso de haber máis dun arquivo teríamos unha fila nova por cada un.
- Exemplo de información almacenada en **`$_FILES`**:
 - `$_FILES["arquivo"]["name"]`. Garda o nome de arquivo orixinal.
 - `$_FILES["arquivo"]["type"]`. Garda o MIME type do arquivo.
 - `$_FILES["arquivo"]["size"]`. Garda o tamaño do arquivo en bytes.
 - `$_FILES["arquivo"]["tmp_name"]`. Garda o nome do arquivo temporal.
 - `$_FILES["arquivo"]["error"]`. Garda calquera código de erro que poida acontecer.



3.11 Envío de archivos en formularios



- Unha das claves deste mecanismo é empregar a función `move_uploaded_file()` para mover o arquivo subido do directorio temporal ao directorio que se lle indique.

```
<?php
    // Comprobar se se cargou un arquivo
    if (isset($_FILES["arquivo"])) && $_FILES["arquivo"]["error"] == 0) {

        $nomeTemporal = $_FILES["arquivo"]["tmp_name"];
        $nomeArquivo = $_FILES["arquivo"]["name"];

        // Mover o arquivo temporal a unha ubicación concreta
        $destino = "arquivos/" . $nomeArquivo;

        // Paso IMPORTANTE
        if (move_uploaded_file($nomeTemporal, $destino)) {
            echo "O arquivo cargouse correctamente.";
        } else {
            echo "Houbo un erro ao cargar o arquivo.";
        }
    } else {
        echo "Por favor, escolla un arquivo válido para cargar.";
    }

?>
```

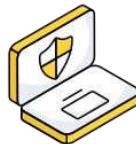


12

Seguridad en formularios

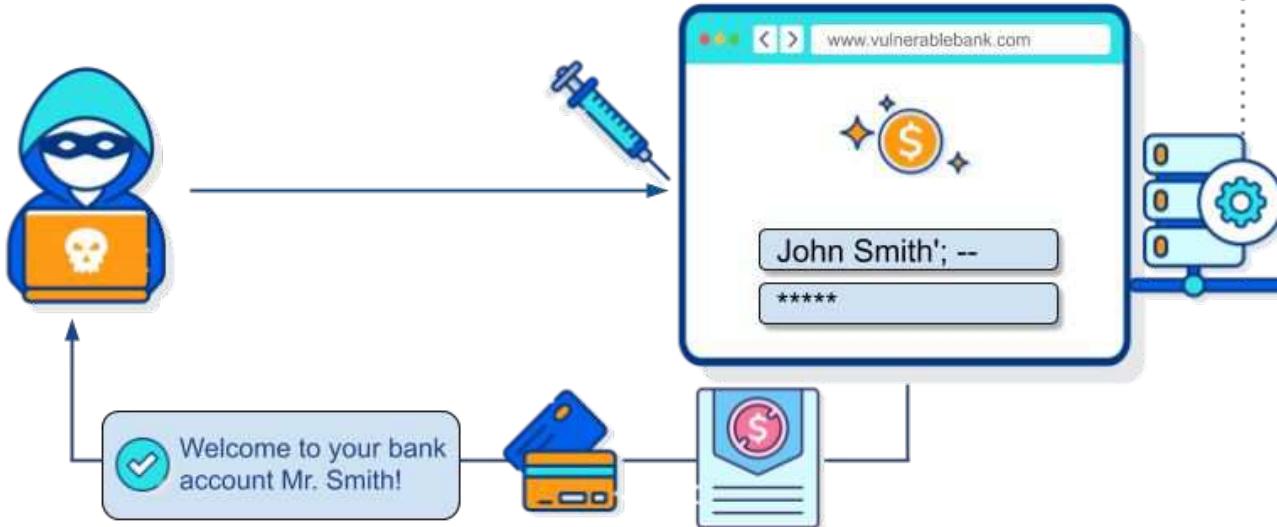


3.12 Seguridade en formularios



- Á hora de traballar con formularios estase a recibir directamente información dos navegadores dos clientes, información que se procesa no servidor e que por tanto, por erro ou de xeito malintencionado pode provocar erros ou riscos de seguridade no noso servidor web. Un exemplo clásico son as **inxeccións SQL**, pero existen máis amenazas como o **Cross-Site-Scripting (XSS)**.

```
SELECT * FROM users WHERE name='John Smith'; --' and password='wrong'
```





3.12 Seguridade en formularios



Código vulnerable a inxección SQL

```
<?php
include('db.php');
$usuario = $_POST['usuario'];
$contrasinal = $_POST['contrasinal'];
$consulta = "SELECT * FROM usuarios WHERE usuario '" . $usuario .
            "' AND contrasinal = '" . $contrasinal . "'";
$resultado = mysqli_query($bd, $consulta);
if (mysqli_num_rows($resultado) != 0) {
    echo "Acceso correcto. <br>";
}
else {
    echo "Acceso denegado. <br>";
}
?>
```



3.12 Seguridade en formularios



Un exemplo clásico é cando á hora inserir un usuario e contrasinal, no campo deste último se engade código SQL que permita que se valide sempre ese contrasinal. Orixinalmente recolleríase:

```
$usuario = $_POST['usuario']; // Por exemplo admin  
$contrasinal = $_POST['contrasinal']; // Por exemplo 1234
```

No `input` do contrasinal pódese inserir calquera cousa, por tanto tecleando unha entrada como:

- Un usuario calquera seguido de comentario SQL: “`admin' --`”
- Un contrasinal calquera seguido de código SQL: “`1234 OR '1'='1`”

Con códigos similares a validación do contrasinal **devolverá sempre true**, dando acceso ao usuario aínda que non coñeza o contrasinal. A consulta SQL resultante quedaría algo así:

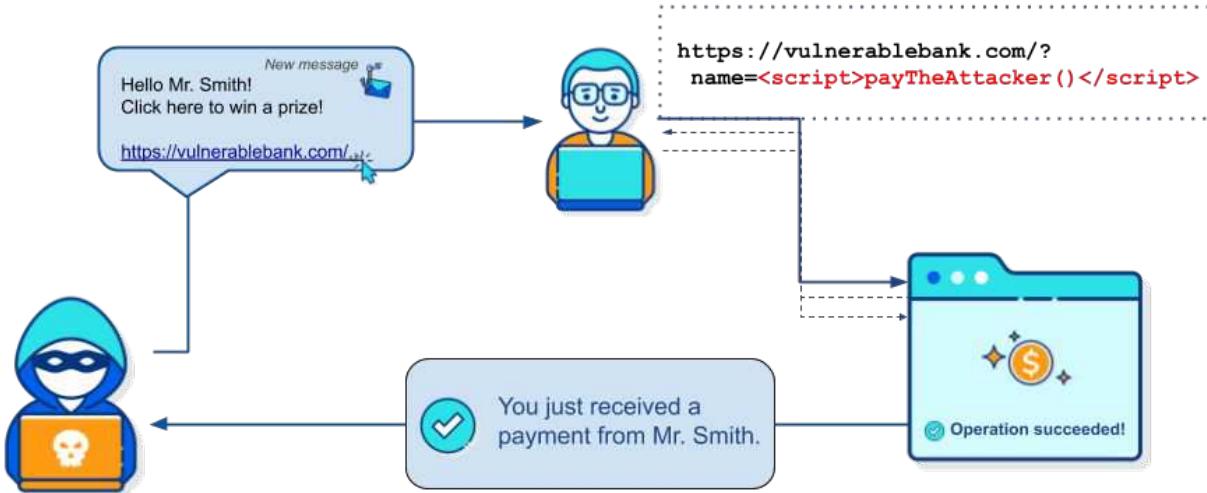
```
SELECT *  
  
FROM usuarios  
  
WHERE usuario='admin'  
  
AND contrasinal='1234' OR '1'='1'
```



3.12 Seguridade en formularios



Os **ataques XSS** xorden cando unha aplicación recibe datos nunha solicitude HTTP e inclúe esos datos na resposta inmediata dunha forma insegura.





3.12 Seguridade en formularios



Código vulnerable a XSS

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo Vulnerable a XSS</title>
</head>
<body>
    <?php
        // Supoñamos que recibidos un variable get na URL
        $nomeUsuario = $_GET['nome'];

        // O código amosa sen filtrar o parámetro:
        echo "<div>Benvid@, $nomeUSuario! </div>";
    ?>
</body>
</html>
```

Unha ligazón desta forma: [http://localhost/exemplo.php?nome=<script>alert\('¡Ataque XSS!'\);</script>](http://localhost/exemplo.php?nome=<script>alert('¡Ataque XSS!');</script>)

Provocaría que ese código Javascript se execute na páxina.



3.12 Seguridade en formularios



- Para **evitar este tipo de ataques**, o primeiro é **validar sempre TODAS as entradas do usuario**.
- PHP proporciona mecanismos específicos para validación como:
 - **Validar e corrixir os campos** cos filtros de corrección de strings que eliminan caracteres especiais a través de `htmlspecialchars()` ou `filter_var()` coa opción `FILTER_SANITIZE_MAGIC_QUOTES`. Evitar “ ‘ ; - \ etc. → PARCIAL
 - **Limitar sempre o tamaño** dos campos recibidos, pensando tamén na base de datos. Por exemplo 20 caracteres para o usuario e 10-12 para o contrasinal. → PARCIAL
 - **mysqli_real_escape_string()**. Elimina caracteres especiais dunha cadea para deixala “limpa” e usala en consultas SQL. → PARCIAL
 - **Consultas parametrizadas**. Doadas de emplegar coa **clase PDO**. → TOTAL

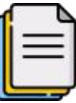


13

Ficheiros

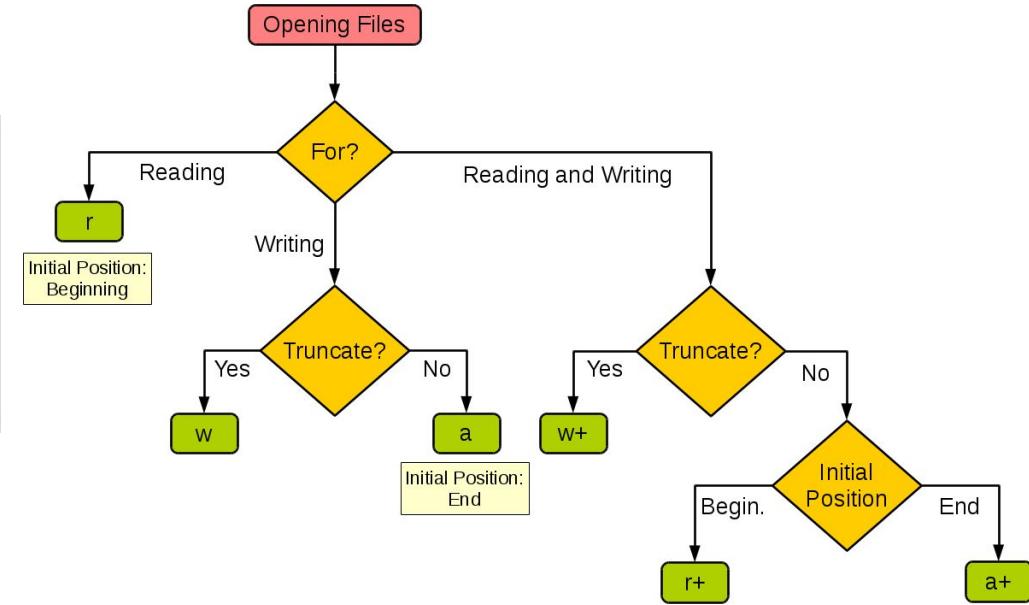


3.13 Ficheiros



- PHP permite xestionar arquivos ou ficheiros para crealos, escribir neles, lelos ou borralos.
- Empregando as funcións disponíveis en PHP pódese definir un punteiro ao arquivo que nos permita operar con el. Para **abrir un arquivo** con PHP empregarse a función **fopen()**, indicando o modo lectura (**r** para lectura, **x** para escritura, seguido de **+** para facer ambas operacións, áñada que admite máis modos):

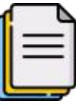
```
● ● ●  
<?php  
  
if (!$/fp = fopen("arquivo.csv", "r")) {  
    echo "Non se pudo abrir o arquivo. <br>";  
}  
  
?>
```



- Podemos comprobar se o ficheiro existe coa función **file_exists()**.



3.13 Ficheiros



- Unha vez aberto pódese **ler** o contido segundo a función **fread()** que collendo o punteiro ao arquivo e unha lonxitude, no exemplo o tamaño do arquivo, obtén o seu contido:

```
<?php  
  
if (!$fp = fopen("arquivo.csv", "r")) {  
    echo "Non se pudo abrir o arquivo. <br>";  
} else {  
    $contido = fread($fp, filesize($arquivo));  
    echo $contido;  
}  
  
?>
```

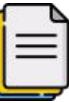
- Igual que ler, podemos **escribir** no arquivo, coa función **fwrite()**, de maneira similar. Tamén convén pechar o arquivo unha vez que rematamos de traballar con el (aínda que non é obligatorio, é unha boa práctica):

```
<?php  
  
if (!$fp = fopen("arquivo.csv", "w")) {  
    // Cambiamos o parámetro a escritura: "w"  
    echo "Non se pudo abrir o arquivo. <br>";  
} else {  
    $texto = "Escrebe isto";  
    $contido = fwrite($fp, $texto);  
    fclose($fp);  
}  
  
?>
```

- Variantes → A función **file()** le o contido de todo o ficheiro e o garda nun array, mentres que **file_get_contents()** o garda nun único string.



3.13 Ficheiros



- Á hora de traballar cos ficheiros pode ser interesante establecer un tamaño de rexistro, ou ben empregar o fin de liña para estruturar a información que almacenamos nel. Se por exemplo quixeramos ir lendo o arquivo **líña a líña**, dispoñemos da función **fgets()** que nos permitirá ir lendo as liñas do arquivo ata chegar ao final do mesmo que podemos controlar coa función **feof()**.



```
<?php  
  
if (!$fp = fopen("arquivo.csv", "r")) {  
    echo "Non se pudo abrir o arquivo. <br>";  
} else {  
    while (!feof($fp)) {  
        $linea = fgets($fp);  
        echo $linea;  
    }  
    fclose($fp);  
}  
?>
```

- Para **escribir** no ficheiro no mesmo formato (**líña a líña**) empregarase a función **fwrite()**, rematando o string a inserir cun salto de liña (**\n**):



```
<?php
```

```
if (!$fp = fopen("arquivo.csv", "a")) {  
    echo "Non se pudo abrir o arquivo. <br>";  
} else {  
    fwrite($fp, "Nova linea... \n");  
    fclose($fp);  
}  
?>
```



3.13 Ficheiros



- Outra función útil, se estamos a traballar con **archivos CSV** é **fputcsv()** que converte directamente un array a formato CSV no ficheiro. Logo pódese traballar coa función **fgetcsv()** para recuperar a información.

```
● ● ●  
=?php  
  
$animais = array (  
    array( 'Pedro', 'Poni', '5' ),  
    array( 'Susi', 'Sheep', '10' ),  
    array( 'Zoe', 'Zebra', '8' ),  
);  
  
$fp = fopen( 'arquivo.csv' , 'w+' );  
  
// Escribir  
foreach ($animais as $fila) {  
    fputcsv($fp, $fila);  
}  
  
// Ler  
while (( $fila = fgetcsv($fp) ) !== false) {  
    // Imprimir cada fila do CSV  
    echo implode( ' ' , $fila ) . "<br>";  
}  
fclose($fp);  
?>
```

- Para **eliminar** un ficheiro podemos empregar a función **unlink()**. Así mesmo dispoñemos de **un bo feixe de operación** más.

```
● ● ●  
  
// Borrar o arquivo  
if (unlink('arquivo.csv')) {  
    echo "O arquivo borrouse correctamente.";  
} else {  
    echo "Houbo un erro ao tentar borrar o arquivo.";  
}
```



Documentación complementaria

- [Manual oficial de PHP.](#)
- [Wiki CIFP Rodolfo Ucha: Sintaxe básica de PHP.](#)
- [Wiki San Clemente: PHP.](#)
- [Patróns PCRE.](#)
- [Funcións PHP para uso de ficheiros.](#)
- [Guía de estilo PHP.](#)





Fin!

¿Algunha pregunta?

Lembra que tamén podes preguntar:

- ◉ Nos foros da aula virtual.
- ◉ Pola mensaxería da aula virtual.