



UT3-4 Ejercicios Varios

1. Una de las características de PHP es su capacidad de gestionar **contenidos dinámicos**. Esto se puede aplicar a la creación de elementos y controles HTML como las listas, tablas, radio buttons, etc. Por ejemplo, una **lista** dinámica:

```
<?php

/* Función para listar una serie de datos que vienen como parámetro.
 * Dado que HTML es sólo texto y que el servidor web genera también
 * sólo texto, se creará una variable de texto y en ella se irá
 * almacenando la salida deseada. Esto será lo que al final retornará
 * la función.
 */
function listar($datos) {
    $salida = "<ul>";    // Inicializar

    foreach($datos as $v) {
        $salida .= "<li>" . $v . "</li>"; // Acumular (concatenar)
    }
    $salida .= "</ul>";

    return $salida;
}

// Datos de la lista
$países = array("España", "Francia", "Portugal", "Italia");

echo "<h3>Ejemplo sencillo de generación dinámica</h3>";
echo "<h4>Listado de países</h4>";

// Lo que devuelve la función es un texto. Por ello se utiliza echo
echo listar($países);

// Tb. valdría $lis = listar($países); echo $lis;

?>
```

Cuya ejecución dará como resultado lo siguiente:

Ejemplo sencillo de generación dinámica

Listado de países

- España
- Francia
- Portugal
- Italia

Si quisiéramos incluir el título del listado en la función, el cambio sería mínimo. Tan sólo sería necesario añadir dicho título como parámetro en la función y adaptar convenientemente la salida.

```
function listar($datos, $titulo) {
    $salida = "<h3>" . $titulo . "</h3>";
    $salida .= "<ul>";

    ...

}
```

Ahora, por ejemplo, se invocaría de la siguiente manera:

```
echo listar($países, 'Listado de países');
```

Y el resultado sería exactamente el mismo que antes.

2. **VALIDACIÓN DE DATOS EN EL SERVIDOR.** Cuando tenemos un formulario de datos, podemos hacer una validación de los mismos tanto en el cliente como en el servidor. Pero en cualquier caso, SIEMPRE en el servidor. Y eso a pesar de que la(s) tecnología(s) de programación clientes son más eficientes y “vistasas” en este aspecto.

El motivo no es otro que la existencia de mecanismos de invocación a la aplicación servidor distintos a la del programa cliente donde se realiza la validación de los datos.

Supongamos el siguiente cliente HTML (*Ejer-2.html*) que solicita el nombre de un directorio e invoca a un servidor PHP (*Ejer-2.php*) para que liste su contenido.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Listar contenido directorio</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  </head>
  <body>
    <h2>Listar directorio</h2>
    <form action="Ejer-2.php" method="get">
      Introduzca directorio:<input type="text" name="fich" required
/><br/>
      <input type="submit" name="enviar" value="Enviar"/>
    </form>
```

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

Archivo Ejer-2.html

Notar que en el input hemos puesto una validación en el campo de tipo **required**, lo que obliga a introducir el nombre del fichero.

```
<?php

function recorreDir($dir) {
    $archivos = array();
    if (is_dir($dir)) {
        if ($dh = opendir($dir)){
            while (($file = readdir($dh)) !== false){
                if ( $file != '.' && $file != '..' )
                    $archivos[] = $file;
            }
            closedir($dh);
        }
    }

    return $archivos;
}

?>
```

Archivo List_fich.php

```
<?php

include 'List_fich.php';

if (isset($_REQUEST['enviar'])) {

    $fich = $_REQUEST['fich'];
    $dir = "." . $fich;
    echo "<h4>Listado directorio " . $dir . "</h4>";
    $archivos = recorreDir($dir);

    foreach ($archivos as $v)
        echo $v . "<br/>";

}

?>
```

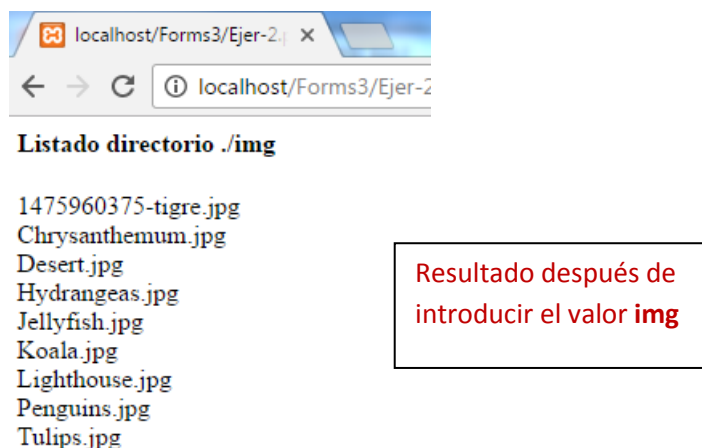
Archivo Ejer-2.php

Sin embargo, no hemos puesto ninguna validación para ese campo ('fich') en el servidor.

Una ejecución sin introducir el valor del campo, efectivamente impide la invocación al servidor.



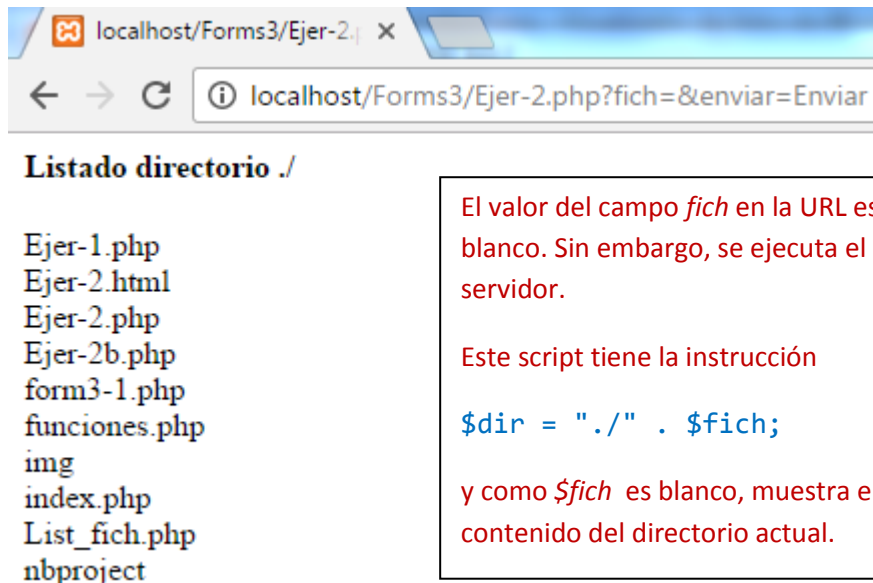
Y si introducimos un valor, sí que se invoca al script servidor. Hasta aquí todo funciona perfectamente y como se espera.



Sin embargo, es posible invocar al servidor sin pasar por el cliente HTML. Por ejemplo, si en el navegador web se introduce, por ejemplo, la URL

<http://localhost/Forms3/Ejer-2.php?fich=&enviar=Enviar>

el script servidor será ejecutado, aunque el valor de *fich* sea blanco.



Así pues, hemos **punteado** al cliente y sin introducir el campo del directorio en el formulario.

Modifiquemos el script servidor para que **valide** si se ha introducido el valor del campo.

```
<?php

include 'List_fich.php';

if (isset($_REQUEST['enviar'])) {

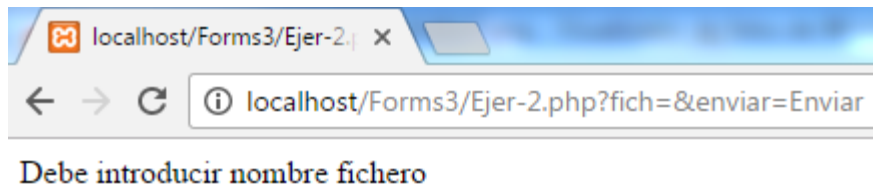
    $fich = $_REQUEST['fich'];
    if (empty($fich))
        echo "Debe introducir nombre fichero";
    else
    {
        $dir = "./" . $fich;
        echo "<h4>Listado directorio " . $dir . "</h4>";
        $archivos = recorrerDir($dir);

        foreach ($archivos as $v)
            echo $v . "<br/>";
    }
}

?>
```

Archivo Ejer-2.php

Volvamos a intentar puentear al cliente HTML introduciendo directamente la URL en el navegador. Ahora el servidor **impide** que se muestre cualquier relación de archivos.



Evidentemente, todo esto ha sido posible porque el servidor PHP está admitiendo parámetros vía GET (los recoge mediante \$_REQUEST). *Si solamente los recogiera vía POST, su invocación directa por URL sería más complicada pero no imposible.* Como ejemplo, el siguiente cliente que incluye código JavaScript para invocar al formulario.

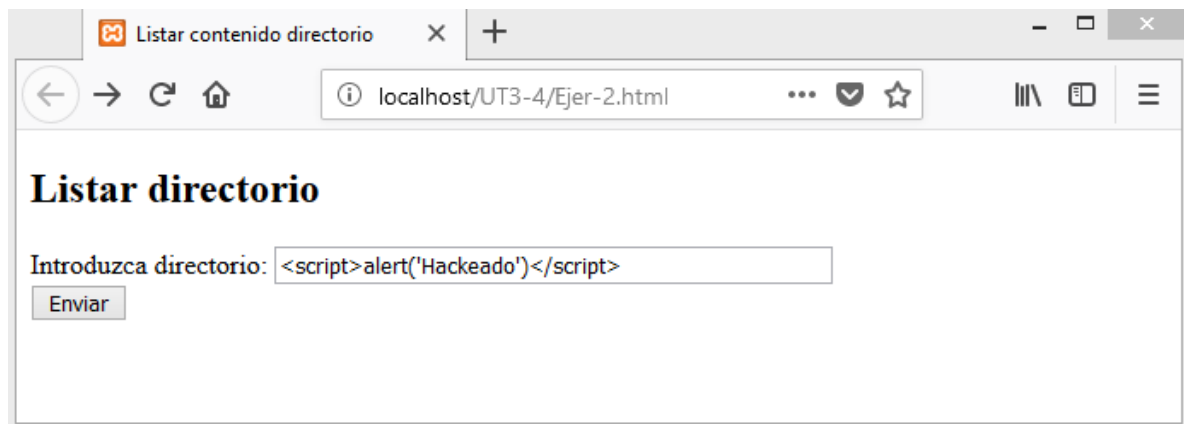
```
<html>
<head>
  <meta content='text/html; charset=utf-8' http-equiv='Content-Type' />
  <title>Invocando vía post mediante JavaScript</title>
  <script type='text/javascript'>
    function enviarForm(){
      document.F.submit();
    }
  </script>
</head>
<body onLoad='javascript:enviarForm();'>
  <form name='F' action='Ejer-2.php' method='post'>
    <input type="hidden" name="fich" value=""/><br/>
    <input type="hidden" name="enviar" value="Enviar"/>
  </form>
</body>
</html>
```

Archivo Ejer-2-JavaScript.html

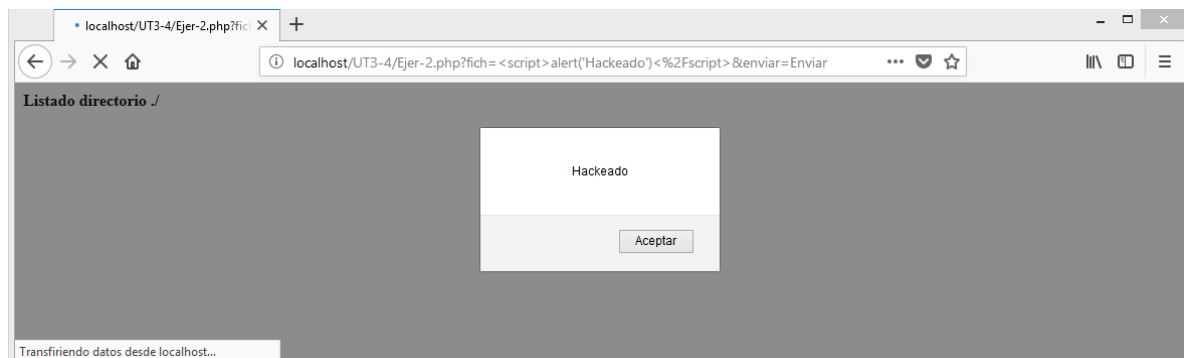
Al ejecutar este cliente, se accede directamente al script servidor vía POST **y sin introducir los datos del formulario**. El truco para este caso está en poner los campos *input* del formulario a **hidden**, campo botón incluido.

Otro problema con el que nos podemos encontrar es el hecho de no validar cierto tipo de caracteres en la entrada de los campos de texto de un formulario. No hay que suponer que el usuario deba introducir el nombre de un directorio. Un usuario malintencionado podría introducir cualquier otra cosa como, por ejemplo, lo siguiente:

<script>alert('hackeado');</script>

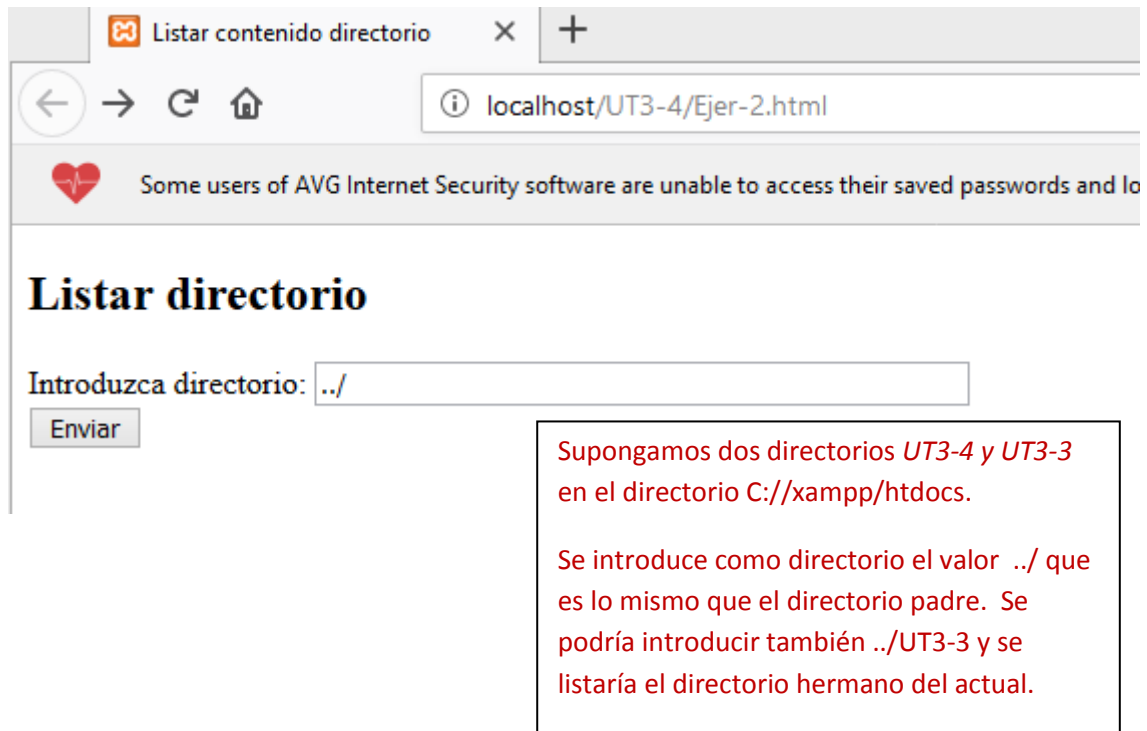


Cuyo resultado en algunos navegadores web como *Firefox* sería:



A este tipo de ataque web se le denomina **XSS** (*Cross-Site Scripting*). En este caso el resultado “no causa estragos” pero se pueden codificar scripts que sí los produzcan.

Otro efecto no deseado es lo que se conoce como **trayectoria transversal** (*traversal path*). Igualmente un usuario malintencionado haciendo uso del carácter “\” (o “/” en el caso de Linux) unido a “..”, (directorio padre) puede conseguir navegar por el árbol de directorios del servidor web. Por ejemplo:



Listar contenido directorio

localhost/UT3-4/Ejer-2.html

Some users of AVG Internet Security software are unable to access their saved passwords and logins.

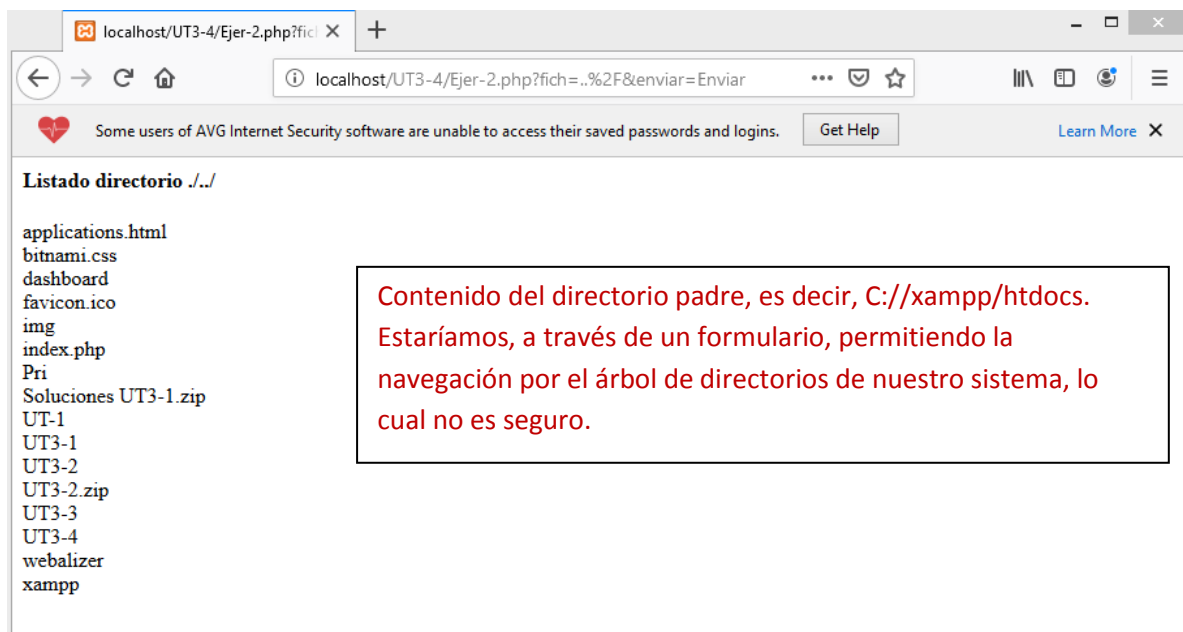
Listar directorio

Introduzca directorio:

Enviar

Supongamos dos directorios *UT3-4* y *UT3-3* en el directorio *C://xampp/htdocs*.

Se introduce como directorio el valor *../* que es lo mismo que el directorio padre. Se podría introducir también *../UT3-3* y se listaría el directorio hermano del actual.



localhost/UT3-4/Ejer-2.php?fich=..%2F&enviar=Enviar

Some users of AVG Internet Security software are unable to access their saved passwords and logins. Get Help Learn More

Listado directorio ../

- applications.html
- bitnami.css
- dashboard
- favicon.ico
- img
- index.php
- Pri
- Soluciones UT3-1.zip
- UT-1
- UT3-1
- UT3-2
- UT3-2.zip
- UT3-3
- UT3-4
- webalizer
- xampp

Contenido del directorio padre, es decir, *C://xampp/htdocs*. Estaríamos, a través de un formulario, permitiendo la navegación por el árbol de directorios de nuestro sistema, lo cual no es seguro.



Listar contenido directorio

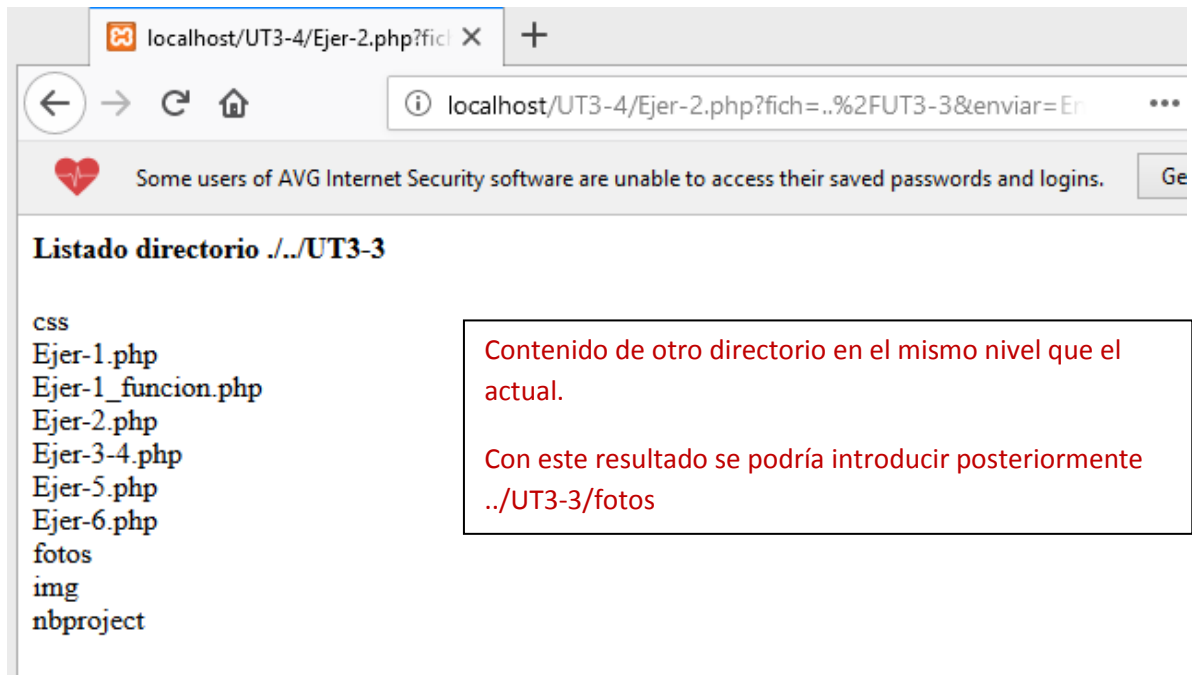
localhost/UT3-4/Ejer-2.html

Some users of AVG Internet Security software are unable to access their saved pas

Listar directorio

Introduzca directorio:

Enviar



Prevención: escapar caracteres HTML (a esto se le llama **sanear** la entrada).

Si sustituimos en el script servidor la línea

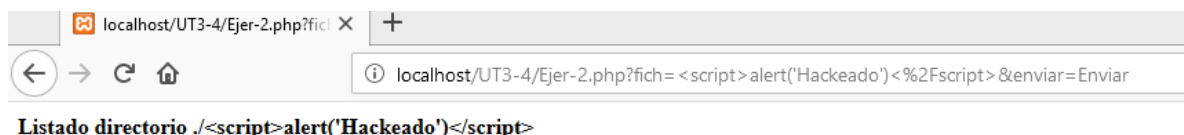
```
$fich = $_REQUEST['fich'];
```

por esta otra

```
$fich = htmlspecialchars($_REQUEST['fich']);
```

estaremos consiguiendo que caracteres especiales de HTML (como < ó >) no sean interpretados como apertura o cierre de etiqueta HTML, sino como eso, *simples caracteres*.

Así pues, si ahora ejecutamos de nuevo el cliente anterior se obtiene en el script servidor lo siguiente:



El directorio, evidentemente, no existe y no se muestra ningún archivo. Los caracteres especiales de apertura y cierre de HTML han sido interpretados como simplemente como "<" y ">", respectivamente.

Si se debe aprender alguna cosa de estos ejercicios, que sea ésta:

Nunca, bajo ninguna circunstancia, se fíe de los datos que llegan del navegador.

Nunca se sabe quién está al otro lado de la conexión HTTP. Puede ser alguno de sus usuarios legítimos, pero puede igualmente ser un «cracker» o un «script kiddie» buscando una vulnerabilidad que explotar.

Cualquier dato de cualquier naturaleza que provenga del navegador debe ser tratado con una *dosis saludable de paranoia*. Esto incluye a los datos que están "en línea" —es decir, enviados desde formularios web— y "fuera de línea" —es decir, las cabeceras HTTP, las cookies y otra información de la petición. Es trivial falsear los metadatos de la petición que suelen añadir automáticamente los navegadores.

Se aconseja echar un vistazo a la página

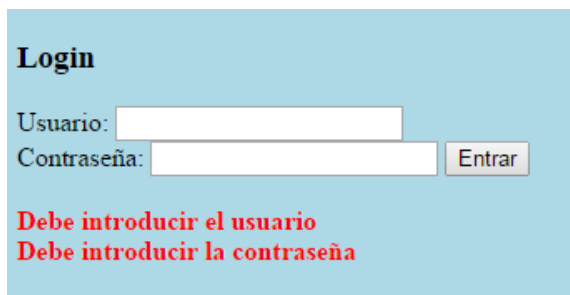
<http://www.wextensible.com/temas/php-formularios-seguros/>

3. Crear un script PHP de **login de acceso** a una aplicación. Aquí se comprobará que se introduzcan ambos datos: *Usuario* y *Contraseña*. Estos datos se validarán contra los existentes en el array asociativo de usuarios siguiente:

```
$credenciales = array(
    'ana' => 'a4a97ffc170ec7ab32b85b2129c69c50',
    'jose' => '10dea63031376352d413a8e530654b8b',
    'marga' => '35559e8b5732fbd5029bef54aeab7a21',
    'anton' => 'C707dce7b5a990e349c873268cf5a968'
);
```

Si falta por introducir alguno(s) de los datos, se mostrará un mensaje indicando cuál(es) falta(n) y no se dejará continuar.

Si el par usuario/contraseña no es válido, se mostrará un mensaje apropiado y no se dejará continuar.

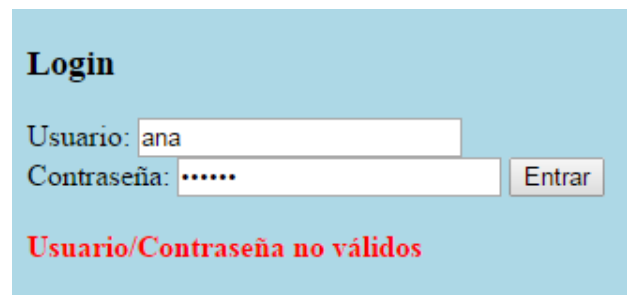


Login

Usuario:

Contraseña:

Debe introducir el usuario
Debe introducir la contraseña



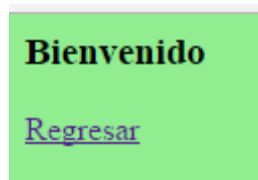
Login

Usuario:

Contraseña:

Usuario/Contraseña no válidos

Una vez validado correctamente el usuario se accederá directamente a una página en concreto. En ella se mostrará un mensaje de bienvenida.



Hasta ahora hemos creado un script PHP donde la parte PHP se encontraba al principio del archivo y la parte cliente al final. Esto era debido a que las variables PHP embebidas en el HTML tenían que estar inicializadas previamente y esto debía hacerse en la parte PHP.

Ahora vamos a hacer lo mismo pero poniendo el HTML al principio y el PHP al final del archivo. Para no tener “problemas” de variables no conocidas o no inicializadas, haremos uso de la función **isset**.

Así pues, creamos en el mismo archivo PHP (p.e. *Ejer-4.php*) el **código HTML** del formulario y seguidamente el **código del procesamiento del servidor** (Es todo el **if (isset(\$_POST['Enviar']))**) { // Se pulsó el botón enviar ==> Validación)

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8"/>
  <title>Ejer-4 control de accesos</title>
</head>
<body bgcolor="lightblue">

  <h3>Login</h3>

  <form action="" method="POST">

    <label>Usuario: </label><input type="text" name="usuario"
      value="<?php if (isset($_POST['usuario'])) echo $_POST['usuario'];
else echo ''; ?>"/><br/>

    <label>Contraseña: </label><input type="password" name="clave"
      value=""/>

    <input type="submit" name="Enviar" value="Entrar">

  </form>

  <br/>
</body>
</html>
```

Notar ahora que en el value no se pone <?php echo \$usuario; ?>, sino
=<?php if (isset(\$_POST['usuario'])) echo \$_POST['usuario'];
else echo ''; ?>

De esta manera se consigue al mismo tiempo comprobar la existencia de
una variable y mostrar un valor.

```
<?php

if ( isset($_POST['Enviar']) ) { // Se pulsó el botón enviar ==> Validación

  $usuario = isset($_POST['usuario']) ? $_POST['usuario'] : null;
  $clave = isset($_POST['clave']) ? $_POST['clave'] : null;

  $hay_datos = comprobar_datos($_POST);

  if ( $hay_datos == "" ) { // Se han introducido los datos

    $ok = comprobar_credenciales($usuario, $clave);

    if ( $ok == true ) { // Permitir ir a la aplicación
      header('Location: Ejer-4-bienvenida.php');
    }
    else {
      echo '<p style="color: red; font-weight:bold">Usuario/Contraseña no
válidos</p>';
    }
  }
  else
    echo '<p style="color: red; font-weight:bold">' . $hay_datos . '</p>';
}

?>
```

Archivo Ejer-3.php

Además, por comodidad, entre envío y envío, conservamos los datos introducidos. En el *value* de los input, en vez de dejar un valor por defecto, ponemos

```
value="<?php echo htmlspecialchars($usuario); ?>"/><br/>
```

y se mostrará en el formulario el valor que *\$usuario* tenga en ese momento, además de sanearlo. Opcionalmente, se puede hacer lo mismo para la contraseña, aunque no es lo habitual.

También incorporaremos al principio del archivo lo siguiente:

```
// Limpiar campos si no se ha pulsado el botón Enviar.

if ( isset($_POST['Enviar'])) {

    $usuario = isset($_POST['usuario']) ? $_POST['usuario'] : null;
    $clave = isset($_POST['clave']) ? $_POST['clave'] : null;
}
```

La conjunción de ambos grupos de instrucciones traerá como resultado que no aparezca nada en los campos del formulario antes de pulsar el botón de enviar. Y, *una vez pulsado dicho botón, el valor del campo quedará asignado con el valor establecido.*

Seguidamente, lo primero que habrá que hacer es **guardar** los campos recibidos del formulario en variables de PHP (se aconseja también realizar una saneado de dichos campos para evitar ataques XSS y similares).

Finalmente, realizar la **validación** de datos según la lógica de negocio correspondiente. En este caso, simplemente es que los datos hayan sido introducidos y que existan en el array de usuarios.

```
$hay_datos = comprobar_datos($_POST);
```

y

```
$ok = comprobar_credenciales($usuario, $clave);
```

```
<!DOCTYPE html>
<head>
    <meta charset="utf-8"/>
    <title>Ejer-3 login</title>
</head>
<body style="background-color: lightblue;">

    <h3>Login</h3>

    <form action="" method="POST">
        <label>Usuario: </label><input type="text" name="usuario"
```

```
        value="<?php      if      (isset($_POST['usuario']))      echo
$_POST['usuario']; else echo ' '; ?>"/><br/>

        <label>Contraseña: </label><input type="password" name="clave"
        value="<?php      if      (isset($_POST['clave']))      echo
$_POST['clave']; else echo ' '; ?>"/>

        <br/>
        <input type="submit" name="Enviar" value="Entrar">
    </form>

</body>
</html>

<?php

function comprobar_datos($datos) {

    $mensaje = "";    // Alberga mensajes de introducción de datos

    if ( $datos['usuario'] == "" )    // Tb. empty($datos['usuario'])

        $mensaje = "Debe introducir el usuario <br/>";

    if ( $datos['clave'] == "" )
        $mensaje .= "Debe introducir la contraseña <br/>";

    return $mensaje;
}

function comprobar_credenciales($usuario, $clave) {

    include_once 'config.inc.php';

    $enc = false;
    foreach ($credenciales as $usu => $vclave) {
        if ( $usuario == $usu  && md5($clave) == $vclave ) {
            $enc = true;
            break;
        }
    }

    return $enc;
}

if ( isset($_POST['Enviar']) ) { // Se pulsó el botón enviar ==> Validación

    $usuario = isset($_POST['usuario']) ? $_POST['usuario'] : null;
    $clave = isset($_POST['clave']) ? $_POST['clave'] : null;

    $hay_datos = comprobar_datos($_POST);
```

```
if ( $hay_datos == "" ) { // Se han introducido los datos

    $ok = comprobar_credenciales($usuario, $clave);

    if ( $ok == true ) { // Permitir ir a la aplicación
        header('Location: Ejer-4-bienvenida.php');
    }
    else {
        echo ' <p style="color: red; font-weight:bold">Usuario/Contraseña
no válidos</p>';
    }
}
else
    echo ' <p style="color: red; font-weight:bold">' . $hay_datos .
'</p>';
}

?>
```

Ejer-3.php

4. Nota: es aconsejable ver el ejemplo de validación existente en la URL

<https://diego.com.es/formularios-en-php>

5. Procesamiento de XML

En ocasiones es necesario leer y procesar información desde un archivo XML. Esto puede ser habitual cuando se transmite información a través de aplicaciones.

Supongamos el siguiente documento XML con datos de libros de una biblioteca.

```
<?xml version="1.0" ?>
<biblioteca>
  <tema id="informatica">
    <libro>
      <titulo>Manual Imprescindible de PHP 6</titulo>
      <autor>Luis Miguel Cabezas Granado</autor>
      <editorial>Anaya Multimedia</editorial>
      <imagen>http://ecx.images-amazon.com/images/I/41YZue-
bJ4L_AA160_.jpg</imagen>
    </libro>
    <libro>
      <titulo>Professional PHP 5</titulo>
      <autor>Lecky Thomson</autor>
      <editorial>Wrox</editorial>
```



```
<imagen>http://ecx.images-amazon.com/images/I/414-
yi8KZuL._AA160_.jpg</imagen>
</libro>
<libro>
  <titulo>Agile Web Development with Rails</titulo>
  <autor>Sam Ruby</autor>
  <editorial>Pragmatic Programmers</editorial>
  <imagen>http://ecx.images-
amazon.com/images/I/41nToGax%2BRL._AA160_.jpg</imagen>
</libro>
<libro>
  <titulo>Rails 4 in Action</titulo>
  <autor>Ryan Bigg</autor>
  <editorial>Manning</editorial>
  <imagen>http://ecx.images-
amazon.com/images/I/51tRMoQ%2BmvL._AA160_.jpg</imagen>
</libro>
</tema>
</biblioteca>
```

El siguiente script PHP es capaz de leer y procesar dicho archivo para generar una salida HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Lectura de XML con SimpleXML</title>
  </head>
  <body>
    <?php
      $biblioteca = simplexml_load_file("biblioteca.xml");
      foreach ($biblioteca->tema as $tema) {
        ?>
        <h2><?php echo $tema["id"]; ?></h2>
        <table class="table">
          <tr>
            <?php
              foreach ($tema->libro as $libro) {
                ?>
                <td>
                  
                  <b><?= $libro->titulo ?><br></b>
                  <?= $libro->autor ?><br>
                  <?= $libro->editorial ?><br>
                </td>
              <?php
            }
          }
        }
      }
    }
  }
}
```

```
?>
</tr>
</table>

</body>
</html>
```

leerXML.php

informatica



Manual
Imprescindible de
PHP 6
Luis Miguel
Cabezas Granado
Anaya Multimedia



Professional PHP 5
Lecky Thomson
Wrox



Agile Web
Development with
Rails
Sam Ruby
Pragmatic
Programmers



Action
Ryan Bigg
Manning

Rails
4 in

Salida de leerXML.php