TEMA 7

Contenido

Aplicaciones web dinámicas. PHP y JavaScript

Caso práctico
Juan *y* ***Carlos*** *tienen definida la estructura de la aplicación web que deben desarrollar. Han llevado a cabo algunas pruebas de programación, y están contentos con los resultados obtenidos. Antes de seguir avanzando, deciden reunir al equipo de BK Programación para mostrarles los progresos realizados y el plan de desarrollo previsto.*
Durante la presentación de la aplicación web, todo el equipo se muestra entusiasmado con el aspecto que está tomando el proyecto. ***Ada****, la directora, les felicita por el trabajo que han llevado a cabo hasta el momento.* ***Ana****, que tiene experiencia como diseñadora gráfica, se ofrece a ayudarles con el aspecto visual del interfaz web. Pero de todas las opiniones, hay una que les llama la atención.* ***María****, que se encarga del mantenimiento de servidores y sitios web, les pregunta si han tenido en cuenta la posibilidad de* ***integrar en su aplicación algún tipo de código cliente.*** *Les comenta que muchas aplicaciones actuales lo utilizan dentro de su estructura para muy diversas funciones.*
Juan *y* ***Carlos*** *se miran, y saben que ambos están pensando lo mismo. Habrá que hacer un último esfuerzo e informarse sobre el tema. Si deciden que resulta interesante, aún están a tiempo de incorporarlo en su proyecto.*

1.- Programación del cliente web

Caso práctico
El primer paso que deciden dar ***Juan*** *y* ***Carlos****, siempre contando con el asesoramiento de* ***María****, es informarse sobre las tecnologías de programación web actuales. Conocen de oídas el lenguaje JavaScript, y saben cuáles son los fundamentos de la tecnología AJAX, pero no tienen claras las ventajas e inconvenientes que les puede suponer su incorporación al proyecto en el que están trabajando.*
Y lo más importante; si finalmente consideran que puede ser ventajoso introducir en su aplicación web algún tipo de programación que se ejecute en el navegador… ¿cómo se puede integrar ésta con la programación del servidor web? ¿Pueden coexistir, y aún más, integrarse ambos lenguajes de alguna forma?
Siguiendo su método de trabajo, deciden buscar información por separado y compartirla pasados unos días.

Cuando comenzaste con el presente módulo, uno de los primeros conceptos que aprendiste es a diferenciar entre la ejecución de código en el servidor web y la ejecución de código en el navegador o cliente web.

Todo lo que has aprendido hasta el momento se ha centrado en la ejecución de código en el servidor web utilizando el lenguaje PHP. La otra parte importante de una aplicación web, la programación de código que se ejecute en el navegador, no forma parte de los contenidos de este módulo. Tiene su propio módulo dedicado, **Desarrollo Web en Entorno Cliente.**

Muchas de las aplicaciones web que existen en la actualidad tienen esos dos componentes: una parte de la aplicación, generalmente la que contiene la lógica de negocio, se ejecuta en el servidor; y otra parte de la aplicación, de menor peso, se ejecuta en el cliente. Existen incluso cierto tipo de aplicaciones web, como Google Docs, en las que gran parte de las funcionalidades que ofrecen se implementan utilizando programación del cliente web.

En la presente unidad vas a aprender cómo integrar estos dos componentes de una misma aplicación web: el código PHP que se ejecutará en el servidor, con el código que se enviará al cliente para que éste lo ejecute.

La programación de guiones para su ejecución en un cliente web es similar a lo que ya conoces sobre PHP, salvo que en este caso el código completo del guión llega al navegador junto con las etiquetas HTML, y es éste el encargado de procesarlo.

Así como el código PHP se marcaba utilizando los delimitadores `<?PHP` y `?>`, en HTML existe una etiqueta que se utiliza para integrar el código ejecutable por el navegador junto al resto de etiquetas. Se trata de la etiqueta `<script>`, que puede indicar tanto la localización del código en un fichero externo, como simplemente delimitar unas líneas de código dentro del propio fichero HTML.

```
// Inclusión de código en el documento HTML
<script type="text/javascript">
// Código que ejecuta el navegador
</script>

// Inclusión de código en un fichero externo
<script type="text/javascript" src="codigo.js"></script>
```

Cuando el código se incluye en el propio documento HTML, se suele encerrar en una sección `CDATA` para no encontrar errores de validación en documentos XHTML. Estos errores aparecerían si dentro del código del guión hubiera algún carácter especial como `<` o `>`.

Sin embargo, al utilizar una sección `CDATA` (*forma de marcar una parte de un documento XML (XHTML en nuestro caso) para que no sea interpretada al procesar el documento. Las siglas se refieren a que esa parte del documento contiene sólo "Character Data"*), puede suceder que cuando se procese la página como documento HTML, algún navegador no reconozca éstas secciones. Es por este motivo que el texto `CDATA` que identifica estas secciones suele también ponerse dentro de un comentario (por ejemplo, utilizando `//` o `/* */`).

```
<script type="text/javascript">
   /* <![CDATA[ */
   // Código que ejecuta el navegador
   /* ]]> */
</script>
```

¿Qué etiqueta HTML se usa para marcar el código que ejecutará el navegador?

- ○ CDATA.
- ◉ **script.**

La etiqueta script puede incluir por si misma el código que ejecutará el navegador, o indicar el fichero en que se encuentra.

1.1.- Página web dinámicas.

La inclusión de código en páginas web para su ejecución por parte del navegador tiene ciertas limitaciones:

- ✓ Cuando ejecutas código PHP en un servidor, es normalmente el programador el que tiene el control sobre el entorno de ejecución. Al cliente llegan únicamente etiquetas en lenguaje HTML o XHTML. Sin embargo, cuando programas código para que se ejecute en un cliente web, no tienes siquiera la certeza de que el navegador del usuario soporte la ejecución del código que recibe. Existen ciertos sistemas, como dispositivos móviles o navegadores integrados en hardware específico, que no permiten la ejecución de código de cliente.
- ✓ El código que se ejecuta en el navegador está normalmente limitado a ser ejecutado en un entorno controlado, que no permite, por ejemplo, la lectura o escritura de ficheros en el ordenador del usuario. De esta forma se restringen los efectos negativos que pueda causar un guión y se favorece la confianza del usuario en este tipo de código.

Pese a estas limitaciones, la ejecución de código en el navegador encaja perfectamente con cierto tipo de tareas como:

- ✓ Comprobar y/o procesar los datos que introduce el usuario en los formularios, como paso previo a su envío al servidor web.

- ✓ Gestionar diferentes marcos y/o ventanas del navegador.
- ✓ Modificar de forma dinámica los elementos que componen la página web, ajustando sus propiedades o estilos en respuesta a la interacción del usuario.

El código JavaScript de una página se puede ejecutar en respuesta a eventos generados por el navegador. Por ejemplo, utilizando el evento `onsubmit` podemos llamar a una función `validar_email` para validar una dirección de correo introducida por el usuario cuando se intenta enviar el formulario:

```
<form action="usuario.php" method="get" name="datos_usuario"
onsubmit="return validar_email()">
<input type="text" id="email" />
</form>
```

Para la función que realiza la validación básica de una dirección de email puedes utilizar, por ejemplo, el siguiente código:


```
function validar_email() {
    valor = document.getElementById("email").value;
    pos_arroba = valor.indexOf("@");
    pos_punto = valor.lastIndexOf(".");
    if (pos_arroba < 1 || pos_punto < pos_arroba+2
|| pos_punto+2>=valor.length) {
        alert('Dirección de correo no válida.');
        return false;
    }
  return true;
}
```

Las páginas web que se aprovechan de las capacidades de ejecución de código en el cliente para cambiar su apariencia, o su funcionamiento, se conocen como páginas web dinámicas. Se llama **HTML dinámico** (DHTML) al conjunto de técnicas que emplean HTML, el modelo de objetos del documento web modelo de objetos del documento web (*DOM: referencia a un conjunto de objetos para representar un documentoHTML y XHTML, que pueden ser usados para acceder mediante programación a las distintas partes del mismo*), hojas de estilo CSS y lenguaje ejecutado en el navegador para crear sitios webs dinámicos.

1.2.- El lenguaje JavaScript.

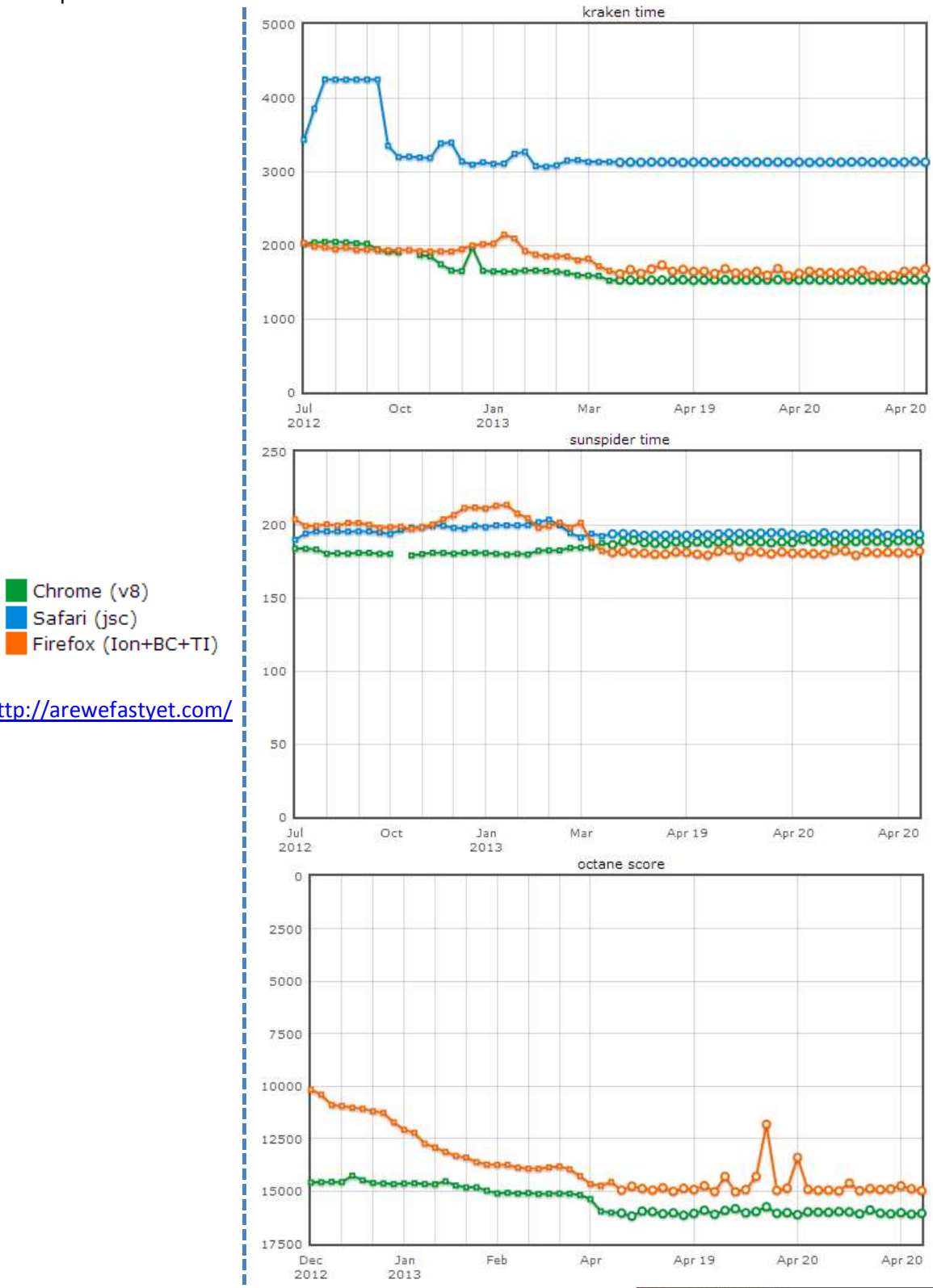
El lenguaje de guiones que se utiliza mayoritariamente hoy en día para la programación de clientes web es JavaScript. Su sintaxis está basada en la del lenguaje C, parecida a la que conocemos del lenguaje PHP. Aunque su utilización principal es incorporarlo a páginas web, también puedes encontrar **JavaScript** en otros lugares como en documentos PDF, o para definir la funcionalidad de extensiones de escritorio o de algunas aplicaciones (widgets *(aplicación, generalmente pequeña, que se ejecuta sobre otra aplicación denominada motor de widgets, y con frecuencia orientada a incorporar alguna funcionalidad adicional o a mejorar el aspecto visual de otra aplicación o del escritorio del sistema operativo del usuario)*).

Si bien, la gran mayoría de navegadores web soportan código en lenguaje JavaScript, debes tener en cuenta que:

- ✓ La ejecución de JavaScript en el navegador puede haber sido deshabilitada por el usuario.
- ✓ La implementación de JavaScript puede variar de un navegador a otro. Lo mismo sucede con el interface de programación que usa JavaScript para acceder a la estructura de las páginas web: el DOM. Por este motivo, es conveniente que verifiques la funcionalidad del código en diversos navegadores antes de publicarlo como parte de tu sitio web.

Se conoce como motor JavaScript a la parte del navegador encargada de interpretar y ejecutar el código JavaScript que forma parte de las páginas web. Los motores JavaScript que se incluyen en los navegadores han experimentado una importante mejora de rendimiento en los últimos tiempos.

Existen pruebas específicas destinadas a medir la velocidad de ejecución de distintos motores JavaScript:



http://arewefastyet.com/

Aunque no vamos a aprender en esta unidad a programar en JavaScript, deberías saber cómo depurar el código que vamos a utilizar. Es conveniente que manejes un depurador para cada navegador que utilices, pero para esta unidad llegará con la **extensión** Firebug para el navegador Firefox.
https://addons.mozilla.org/es-es/firefox/addon/firebug/

¿Cuál es una de las principales desventajas de la programación del cliente web?

- (•) **Que no es posible asegurar que el navegador vaya a ejecutar el código.**
- ( ) Que el navegador puede no ser capaz de mostrar correctamente la página al confundir el código con las etiquetas HTML / XHTML.

La ejecución de código JavaScript puede estar deshabilitada por el usuario, o el navegador puede no soportar la ejecución de código.

1.3.- Comunicación asíncrona con el servidor web: AJAX.

Una de las principales causas de la evolución de JavaScript en los últimos tiempos es, sin duda, la tecnología **AJAX**. Como ya vimos en la primera unidad, el término AJAX hace referencia a la posibilidad de una página web de establecer una comunicación con un servidor web y recibir una respuesta sin necesidad de que el navegador recargue la página.

AJAX utiliza el objeto `XMLHttpRequest`, creado originariamente por Microsoft como parte de su librería MSXML, y que hoy en día se ha incorporado de forma nativa a todos los navegadores actuales.

Pese al nombre del objeto, y a que la letra X de las siglas AJAX hace referencia a XML, la información que se transmite de forma asíncrona entre el navegador y el servidor web no es necesario que se encuentre en formato XML.

Entre las tareas que puedes llevar a cabo gracias a AJAX están:

- ✓ Actualizar el contenido de una página web sin necesidad de recargarla.
- ✓ Pedir y recibir información desde un servidor web manteniendo la página cargada en el navegador.
- ✓ Enviar información de la página a un servidor web en segundo plano.

Dependiendo del navegador del usuario, y de si utiliza una versión antigua o moderna, tendrás que usar un método u otro para crear el objeto `XMLHttpRequest`:

```
// Distintas formas para crear el objeto
// XMLHttpRequest según el navegador
xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
xmlhttp = new XMLHttpRequest();
```

Afortunadamente, muchas de las librerías JavaScript de las que hablábamos antes soportan también AJAX, y utilizan un código adaptado según el navegador del usuario. Si utilizas una de estas librerías podrás ahorrarte muchos quebraderos de cabeza al programar. Por ejemplo, si utilizas jQuery podrías utilizar AJAX para enviar en segundo plano el email validado en el ejemplo anterior. Simplemente tendrías que incluir en el HTML la librería jQuery.

```
// Utilizamos la versión de jQuery disponible en las CDN de Google
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
```

Y tras el código de validación, debes ejecutar la función **ajax** incluida en la librería jQuery:

```
function validar_email() {
    valor = document.getElementById("email").value;

    // Aquí iría el código de validación

    $.ajax({
      type: "POST", url: "email.php", data: "email=" + valor,
      statusCode: {
        404: function() { alert('Página no encontrada'); }
      },
      success: function(result) { alert( "Resultado: " + result ); }
    });
    return false;
```

```
}
```

La función anterior envía la dirección de email introducida mediante POST a la página `email.php`, y muestra un mensaje con el resultado obtenido.

2.- PHP y JavaScript.

Caso práctico
*Pasados unos días, **Juan** y **Carlos** se vuelven a reunir y deciden que en el diseño de su aplicación existen muchos lugares en los que podrían aprovechar las capacidades que ofrece la programación del navegador web, concretamente el lenguaje JavaScript.*
*Sin embargo, ambos siguen sin tener claro cómo pueden integrar el código del cliente web en una aplicación programada en lenguaje PHP. Animados por **María**, que les orienta sobre el rumbo que deben tomar, preparan una serie de pruebas de programación que les puedan orientar sobre el tema.*

Si sabes programar aplicaciones que se ejecuten en un servidor web (con un lenguaje como PHP) y en el navegador del usuario (con JavaScript/jQuery), tienes en tu mano las herramientas necesarias para construir aplicaciones web completas. Sin embargo, es necesario que antes de comenzar tengas claras las funcionalidades que soporta cada una de estas tecnologías de desarrollo, y cómo puedes hacer para utilizar ambas a la vez.

Llevándolo a los extremos, podrías hacer aplicaciones en PHP que utilicen programación del cliente simplemente para tareas sencillas como verificar campos en los formularios antes de enviarlos. O, por el contrario, sería también posible programar aplicaciones completas que se ejecuten en el navegador del usuario, dejando el lenguaje del servidor para proporcionar ciertas funcionalidades como almacenamiento en bases de datos.

Una alternativa no es necesariamente mejor que la otra. Es necesario analizar de forma independiente la lógica de cada aplicación, de manera que no se malgasten los recursos del servidor realizando tareas que podrían trasladarse al cliente web. En ocasiones también es necesario comprobar los tiempos de carga de las páginas y el tamaño de las mismas. Puede ser preferible utilizar AJAX para, por ejemplo, enviar nuevos registros al servidor, si el esfuerzo que invertimos redunda en un interfaz de usuario más ágil y usable. En cualquier caso, la consistencia y robustez de la aplicación no debe verse afectada.

Si decides unir en una aplicación programación del cliente web con programación del servidor web, habrá ocasiones en que necesites comunicar ambos lenguajes. En el ejemplo anterior ya has comprobado cómo puedes hacer para pasar un valor o una variable JavaScript desde el navegador web a un guión en PHP: enviándolo como parámetro `POST` o `GET` en una petición de nueva página, bien sea al cargarla en el navegador o utilizando AJAX en una comunicación asíncrona:

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Desarrollo Web</title>
   </head>
   <body>
      <script type="text/javascript">
         <?php
            // Creamos en la página un código JavaScript que
            //   utiliza la variable PHP "$email"
            $email = "alumno@educacion.es";
            print 'window.open("email.php?email='.urlencode($email).'");';
         ?>
      </script>
   </body>
</html>
```

En ambos casos deberás asegurarte de que las cadenas que insertes en las direcciones URL no incluyan caracteres inválidos. Para evitarlo, en PHP puedes usar la función `urlencode`, y en JavaScript `encodeURIComponent`.

¿Cómo es posible obtener en PHP el contenido de una variable JavaScript?

- (•) **En una petición de nueva página, como parámetro POST o GET.**
- ( ) Enviando desde el servidor web una solicitud al navegador web.

Además, ten en cuenta que cualquier contenido que forme parte de una URL debería procesarse previamente utilizando la función JavaScript encodeURIComponent.

2.1.- Aplicaciones web con PHP y JavaScript.

Con lo que has visto hasta el momento, seguramente, ya te has hecho una idea de qué tareas son las que con más frecuencia hacen uso de JavaScript. Y si hay una que destaca sobre el resto es, sin duda, la validación de formularios, que vamos a utilizar en los ejemplos que se desarrollan a lo largo de la presente unidad.

Como ya sabes, el mecanismo que utilizan las aplicaciones web para permitir al usuario la entrada de información es el formulario HTML. Los datos que se introducen en un formulario web se envían al servidor utilizando bien el método `POST`, bien el método `GET`. Muchos de los campos de los formularios tienen, normalmente, restricciones sobre el contenido que se debe rellenar. La validación de un formulario web consiste en comprobar que el contenido introducido en todos los componentes del mismo cumpla estas restricciones.

Si no utilizas código ejecutable en el navegador, la única forma de validar un formulario consiste en enviarlo al servidor web para comprobar si existen errores en los datos. En caso de que así sea, habrá que volver a enviar el formulario al navegador del usuario mostrando las advertencias oportunas.

Obviamente, si utilizas JavaScript en tus aplicaciones, la validación se puede realizar en el cliente web. De esta forma el proceso es mucho más rápido. No es necesario enviar la información al servidor hasta que se haya comprobado que no existen errores de validación.

Sin embargo, aunque parecen claras las ventajas de la validación de formularios en el cliente web, hay ocasiones en las que ésta no es posible. Ya vimos que no siempre podrás asegurar que el navegador que utiliza el usuario tiene capacidad para ejecutar código JavaScript, o incluso puede suceder que se haya deshabilitado por motivos de seguridad.

En los casos que no sea posible asegurar la capacidad de ejecución de código de los clientes, la solución óptima sería utilizar un escenario dual: diseñar las aplicaciones suponiendo que es posible ejecutar JavaScript en los clientes, y al mismo tiempo prever la posibilidad de que no sea así. Por ejemplo, en el caso de la validación del formulario, podrías crear el código JavaScript de validación y, si en algún cliente este código no se puede ejecutar, preparar un código PHP similar que realice la validación en el servidor.

Por ejemplo, supongamos que queremos validar los datos que se introducen en el siguiente formulario web:

Si realizas la validación en PHP, podrías generar junto con la página web el texto con las advertencias de validación. Y si el formulario lo quieres validar también utilizando JavaScript, tendrás que crear los mismos textos o similares. Una posibilidad para no repetir el código que introduce esos textos en el cliente y en el servidor, es introducir los textos de validación en las etiquetas HTML de la página web, y utilizar estilos para mostrarlos, o no, según sea oportuno.

Por ejemplo, para realizar la validación del formulario web anterior, puedes crear los siguientes textos en HTML (asociados al nombre, las contraseñas y la dirección de correo respectivamente):

```
<span class='error'>Debe tener más de 3 caracteres.</span>
<span class='error'>Debe ser mayor de 5 caracteres o no coinciden.</span>
<span class='error'>La dirección de email no es válida.</span>
```

El código PHP y JavaScript deberá ocultar cada uno de los textos cuando la validación de su elemento respectivo sea correcta. Por ejemplo, si el nombre tiene más de tres letras, validará correctamente y no se deberá mostrar el primer mensaje.

2.2.- Aplicaciones web con PHP y JavaScript (II).

Para realizar la validación del formulario en el servidor web utilizando PHP, necesitarás utilizar las siguientes funciones o similares:

```
<?php
   function validarNombre($nombre){
      if(strlen($nombre) < 4) return false;
      return true;
   }

   function validarEmail($email){
      return     preg match("/^[a-z0-9]+([ \\.-][a-z0-9]+)*@([a-z0-9]+([\.-][a-z0-9]+)*)+\\.[a-
z]{2,}$/i", $email);
   }

   function validarPasswords($pass1, $pass2) {
      return $pass1 == $pass2 && strlen($pass1) > 5;
   }

   function validar($nombre, $email, $pass1, $pass2) {
      return validarNombre($nombre) && validarEmail($email)
        && validarPasswords($pass1, $pass2);
   }
?>
```

Fíjate en el uso de la función **`preg_match`** y de expresiones regulares (*expresión regular o patrón es una cadena de texto compuesta por un conjunto de caracteres, algunos de los cuales tienen significado especial, que permite definir una serie de reglas con las que comprobar la validez, o no, de otras cadenas de texto*) para validar la dirección de correo.

Función preg_match **http://es2.php.net/manual/es/function.preg-match.php**

Ten en cuenta que las barras invertidas (`\`) tienen un significado especial dentro de una cadena de PHP (*sirven para escapar el siguiente carácter, por ejemplo por si queremos mostrar unas comillas*); por ese motivo, la doble barra `\\` se convierte en una barra simple `\` cuando se interpreta la expresión regular.

En ocasiones es importante saber construir expresiones regulares en PHP para realizar comparación de cadenas de texto con patrones. Tienes más información en el siguiente enlace.

Expresiones regulares en PHP
http://www.desarrolloweb.com/manuales/expresiones-regulares.html

Con las funciones anteriores, puedes crear código en PHP que oculte los mensajes de validación cuando no sean necesarios:

```
<span    id='errorNombre'    for='nombre'    class='<?php    if(!isset($_POST['enviar'])    ||
validarNombre($_POST['nombre']))   echo   "oculto   ";   ?>error'>Debe   tener   más   de   3
caracteres.</span>
<span   id='errorPassword'   for='password'   class='<?php   if(!isset($ POST['enviar'])   ||
validarPasswords($ POST['password1'], $ POST['password2'])) echo "oculto "; ?>error'>Debe ser
mayor de 5 caracteres o no coinciden.</span>
<span    id='errorEmail'    for='email'    class='<?php    if(!isset($_POST['enviar'])    ||
validarEmail($_POST['email']))   echo   "oculto   ";   ?>error'>La   dirección   de   email   no   es
válida.</span>
```

El código anterior aplica la clase `oculto` a los textos de validación que no sea necesario mostrar. En la hoja de estilos correspondientes, deberás definir para esa clase un estilo como `display:none` o `visibility: hidden`.

¿Qué significa la siguiente expresión regular: \.[a-z]{2,}?

- ○ Una barra seguida por un carácter cualquier, una letra entre la a y la z, y un número mayor o igual a 2.
- ◉ **Un punto seguido de 2 o más letras minúsculas.**

form.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario con PHP: form.php -->
<?php require_once("validar.php"); ?>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
</head>

<body>
    <div id='form'>
    <form id='datos' action='form.php' method='post'>
    <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
            <label for='nombre' >Nombre:</label>
            <input type='text' name='nombre' id='nombre' maxlength="50" value="<?php echo
$_POST['nombre'] ?>" /><br />
            <span id='errorNombre' for='nombre' class='<?php if(!isset($_POST['enviar']) ||
validarNombre($_POST['nombre'])) echo "oculto "; ?>error'>Debe tener más de 3
caracteres.</span>
        </div>
        <div class='campo'>
            <label for='password1' >Contraseña:</label>
            <input type='password' name='password1' id='password1' maxlength="50" value="<?php
echo $_POST['password1'] ?>" /><br />
            <span id='errorPassword' for='password' class='<?php if(!isset($_POST['enviar'])
|| validarPasswords($_POST['password1'], $_POST['password2'])) echo "oculto "; ?>error'>Debe
ser mayor de 5 caracteres o no coinciden.</span>
        </div>
        <div class='campo'>
            <label for='password2' >Repita la contraseña:</label>
            <input type='password' name='password2' id='password2' maxlength="50" value="<?php
echo $_POST['password1'] ?>" />
        </div>
        <div class='campo'>
            <label for='email' >Email:</label>
            <input type='text' name='email' id='email' maxlength="50" value="<?php echo
$_POST['email'] ?>" /><br />
            <span id='errorEmail' for='email' class='<?php if(!isset($_POST['enviar']) ||
validarEmail($_POST['email'])) echo "oculto "; ?>error'>La dirección de email no es
válida.</span>
        </div>

        <div class='campo'>
            <input type='submit' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
</body>
</html>
```

validar.php

```
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario: validar.php -->
<?php
```

```
function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return      preg_match("/^[a-z0-9]+([_\\.-][a-z0-9]+)*@([a-z0-9]+([\.-][a-z0-9]+)*)+\\.[a-
z]{2,}$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

?>
```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 340px;
  margin-left: -170px;
  height: 330px;
  margin-top: -150px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#form .campo {
  margin-top:8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
  border : 1px solid #999;
  height:18px;
  width:280px;
}

#form input[type="submit"] {
  width:100px;
  height:30px;
  padding-left:0px;
}

.error{
  font-size:10px;
  text-decoration: underline;
  background: #ffdddd;
  color: #ee2211;
}

.oculto {
  display: none;
}
```

2.3.- Aplicaciones web con PHP y JavaScript (III).

Partiendo de la página PHP anterior, que ya incluye código para validar el formulario web en el servidor, vas a ver cómo puedes hacer para incorporarle código en JavaScript que realice la misma validación en el cliente. De esta forma, si el navegador del usuario soporta JavaScript, se reducirá el procesamiento del servidor y la transferencia de información entre éste y el cliente.

Crearás todo el código necesario en un archivo externo que llamaremos **validar.js**. Como vamos a utilizar la librería jQuery, tendrás que añadir las dos líneas siguientes a la página anterior:

```
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
<script type="text/javascript" src="validar.js"></script>
```

En el código JavaScript habrá que definir unas funciones de validación similares a las programadas anteriormente en PHP:

```
function validarNombre(){
    if(nombre.val().length < 4) {
        errorNombre.removeClass("oculto");
        return false;
    }
    errorNombre.addClass("oculto");
    return true;
}

function validarEmail(){
…
}

function validarPasswords(){
…
}

function validar(){
    return validarNombre() & validarEmail() & validarPasswords();
}
```

En JavaScript usamos la función `match` para validar las direcciones de email. Las expresiones regulares que admite esta función no son exactamente iguales a las que viste anteriormente para la función `preg_match` dePHP.

Las funciones usan los métodos de jQuery **addClass** y **removeClass** para incorporar, y quitar, respectivamente, los textos de validación a la clase `oculto`, lo mismo que hacía el código PHP anterior. Para seleccionar los elementos a ocultar, puedes utilizar también jQuery. Por ejemplo, para seleccionar el elemento de la página web con identificador **nombre**, se pone:

```
var nombre = $("#nombre");
```

Como no quieres que el formulario se envíe realmente, captura el evento **submit** del formulario forzando a que se valide con la función de JavaScript. Si la validación es correcta (**return true;**), el formulario finalmente se envía.

```
$("#datos").submit(function(){
    if(validar()) return true;
    else return false;
});
```

El código final del ejemplo realiza la validación tanto en el cliente (si éste soporta JavaScript) como en el servidor web (si no soporta JavaScript). Puedes probar a deshabilitar la ejecución de JavaScript en el navegador para comprobar que el formulario se sigue validando.

form.php

```
<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      HTML      4.01      Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
```

```
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario con PHP y JavaScript: form.php -->
<?php require_once("validar.php"); ?>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
</head>

<body>
    <div id='form'>
    <form id='datos' action='form.php' method='post'>
    <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
            <label for='nombre' >Nombre:</label>
            <input type='text' name='nombre' id='nombre' maxlength="50" value="<?php echo
$_POST['nombre'] ?>" /><br />
            <span id='errorNombre' for='nombre' class='<?php if(!isset($_POST['enviar']) ||
validarNombre($_POST['nombre'])) echo "oculto "; ?>error'>Debe tener más de 3
caracteres.</span>
        </div>
        <div class='campo'>
            <label for='password1' >Contraseña:</label>
            <input type='password' name='password1' id='password1' maxlength="50" value="<?php
echo $_POST['password1'] ?>" /><br />
            <span id='errorPassword' for='password' class='<?php if(!isset($_POST['enviar'])
|| validarPasswords($_POST['password1'], $_POST['password2'])) echo "oculto "; ?>error'>Debe
ser mayor de 5 caracteres o no coinciden.</span>
        </div>
        <div class='campo'>
            <label for='password2' >Repita la contraseña:</label>
            <input type='password' name='password2' id='password2' maxlength="50" value="<?php
echo $_POST['password1'] ?>" />
        </div>
        <div class='campo'>
            <label for='email' >Email:</label>
            <input type='text' name='email' id='email' maxlength="50" value="<?php echo
$_POST['email'] ?>" /><br />
            <span id='errorEmail' for='email' class='<?php if(!isset($_POST['enviar']) ||
validarEmail($_POST['email'])) echo "oculto "; ?>error'>La dirección de email no es
válida.</span>
        </div>

        <div class='campo'>
            <input type='submit' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
    <script                                                          type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
    <script type="text/javascript" src="validar.js"></script>
</body>
</html>
```

validar.php

```
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejemplo Validación formulario: validar.php -->
<?php
function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return      preg_match("/^[a-z0-9]+([_\\.-][a-z0-9]+)*@([a-z0-9]+([\.-][a-z0-9]+)*)+\\.[a-
z]{2,}$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}
```

```
?>
```

validar.js

```
$(document).ready(function(){
function validarNombre(){
    if(nombre.val().length < 4) {
        errorNombre.removeClass("oculto");
        return false;
    }
    errorNombre.addClass("oculto");
    return true;
}

function validarEmail(){
    if(!email.val().match("^[a-zA-Z0-9]+[a-zA-Z0-9_-]+@[a-zA-Z0-9]+[a-zA-Z0-9.-]+[a-zA-Z0-
9]+.[a-z]{2,4}$")) {
        errorEmail.removeClass("oculto");
        return false;
    }
    errorEmail.addClass("oculto");
    return true;
}

function validarPasswords(){
    if(password1.val().length < 6 || password1.val() != password2.val()) {
        errorPassword.removeClass("oculto");
        return false;
    }
    errorPassword.addClass("oculto");
    return true;
}

function validar(){
    return validarNombre() & validarEmail() & validarPasswords();
}

var nombre = $("#nombre");
var password1 = $("#password1");
var password2 = $("#password2");
var email = $("#email");
var errorNombre = $("#errorNombre");
var errorPassword = $("#errorPassword");
var errorEmail = $("#errorEmail");

$("#datos").submit(function(){
    if(validar()) return true;
    else return false;
});

});
```

3.- Utilización de AJAX con PHP.

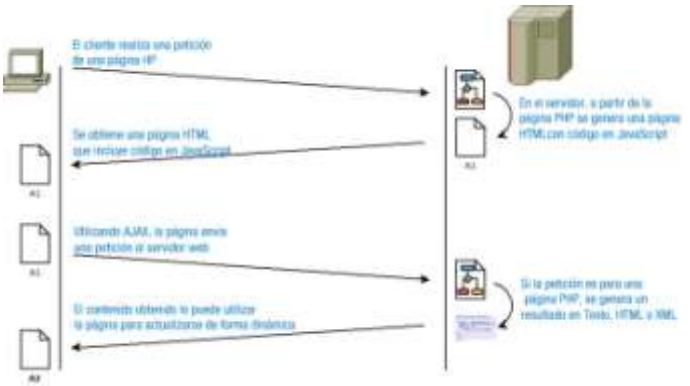
Caso práctico
*En los últimos días **Juan** y **Carlos** han logrado grandes avances. Tienen claras las capacidades que les ofrece la programación del cliente web. Han profundizado en la tecnología AJAX. Y saben, incluso, de qué manera pueden integrar en una misma aplicación ambos tipos de programación: la programación del servidor web en lenguaje PHP, y la programación del cliente web en lenguaje JavaScript.*
*Sólo les falta un detalle: las herramientas. Saben lo qué tienen que hacer, pero antes de ponerse manos a la obra, deben decidir de qué forma hacerlo. **María** les ha informado de que existen multitud de librerías y herramientas que se pueden utilizar para agilizar la programación de aplicaciones en lenguaje JavaScript. Su último paso será decidir qué librerías y herramientas utilizarán para facilitar la programación de la aplicación web, tomando como punto de partida que sería preferible utilizar un único lenguaje en todo el proyecto para evitar la fragmentación del código de la misma.*

Como ya sabes, la tecnología AJAX se utiliza desde el cliente web para permitir comunicaciones asíncronas con el servidor web, sin necesidad de recargar la página web que se muestra en el navegador. Se basa en la utilización de código en lenguaje JavaScript. Por tanto, te estarás preguntando, ¿qué tiene que ver la tecnología AJAX con el lenguaje PHP?

Acabamos de ver cómo se pueden crear aplicaciones web que utilicen de forma simultánea la programación del cliente web (con JavaScript) y la programación del servidor web (con PHP). En el procedimiento seguido en el ejemplo anterior, ambas tecnologías coexistían en una misma página, pero su programación era independiente. El código PHP se ejecutaba en el servidor, y en la misma página se incluían también guiones en lenguaje JavaScript que se ejecutan en el navegador. En nuestro ejemplo solamente existía una relación: si el navegador permitía la ejecución de código JavaScript, se deshabilitaba el envío del formulario y la validación se realizaba en el cliente. En este caso no se llegaba a enviar la página al servidor y no se ejecutaba por tanto el código PHP de validación.

Si vas a usar aplicaciones que utilicen ambos lenguajes, es preferible tener un mecanismo mejor para integrarlos. Afortunadamente existen librerías para PHP que te permiten aprovechar las capacidades de JavaScript utilizando casi exclusivamente código en lenguaje PHP. Estas librerías definen una serie de objetos que puedes utilizar en el código de servidor, y que generan de forma automática código JavaScript en las páginas web que se envían al cliente.

La mayoría de estas librerías añaden a las aplicaciones web funcionalidades de la tecnología AJAX. Esto es: permiten crear páginas PHP que, tras ejecutarse en el servidor, producen páginas web que incorporan código JavaScript con funcionalidades AJAX. El mecanismo de funcionamiento lo puedes observar en el siguiente diagrama.



Muchas de estas librerías suelen apoyarse en librerías JavaScript como jQuery para la ejecución de código en el cliente. En Internet puedes encontrar información sobre librerías PHP con soporte para AJAX.

Librerías PHP con soporte para AJAX **http://ajaxpatterns.org/PHP_Ajax_Frameworks**

A continuación vas a aprender a utilizar dos de estas librerías: `Xajax` y `jQuery4PHP`.

¿Qué es jQuery?

- ○ Una librería de programación para PHP.
- ◉ **Una librería de programación para JavaScript.**

La librería para PHP que nos permite utilizar la funcionalidad de jQuery se llama jQuery4PHP.

3.1.- Xajax.

Xajax es una librería PHP de código abierto que permite generar aplicaciones web con tecnología AJAX. Facilita la utilización desde el cliente de funciones existentes en el servidor. Al utilizar los objetos AJAX en el código PHP, las páginas HTML que se obtienen incorporan el código JavaScript necesario para realizar las llamadas al servidor mediante AJAX. En la página web del proyecto tienes información disponible sobre su utilización.

Página web del proyecto **http://www.xajax-project.org/en/home/**

Para poder utilizar AJAX, descárgate la última versión de la librería desde la sección de descargas de su página web. De las carpetas que contienen los ficheros comprimidos, necesitas el contenido de `xajax_core` y `xajax_js`. Cópialas a una ruta de tu servidor web en la que sean accesibles por tus aplicaciones web.

Sección de descargas **http://www.xajax-project.org/en/download/**

En las **páginas PHP en que quieras utilizar AJAX**, deberás incluir la librería escribiendo el siguiente código:

```
require_once("xajax_core/xajax.inc.php");
```

Asegúrate de que la ruta a la librería sea la correcta. Lo siguiente es crear un objeto de la clase `xajax`, indicando como parámetro el script PHP que contiene las funciones a las que se podrán realizar llamadas mediante AJAX. Puedes incluir estas funciones en una página aparte o en la misma página PHP, y en este caso no será necesario indicar ningún parámetro:

Clase xajax **http://www.xajax-project.org/en/docs-tutorials/api-docs/**

```
$xajax = new xajax();
```

AJAX necesita incluir en la página web que se envía al navegador su propio código JavaScript. Para ello, tienes que incluir en tu código PHP la siguiente llamada al método `printJavaScript` del objeto `$xajax`:

```
$xajax->printJavascript();
```

En caso necesario, también deberás configurar la ruta de acceso a la carpeta `xajax_js` que contiene el código JavaScript de la librería (usa tu propia ruta como segundo parámetro):

```
$xajax->configure('javascript URI','./libs/');;
```

Existen otras opciones de configuración de Xajax. Por ejemplo, cuando las cosas no funcionan como deberían, es muy interesante la opción que activa los mensajes de depuración:

```
$xajax->configure('debug',true);
```

Y por último, tienes que utilizar el método `register` para registrar cada una de las funciones PHP del servidor que estarán disponibles para ser ejecutadas de forma asíncrona desde el navegador:

```
$xajax->register(XAJAX_FUNCTION,"funcion1");
$xajax->register(XAJAX_FUNCTION,"funcion2");
…
```

En el **guión PHP en que definas las funciones** (si no es la misma página que la anterior), tienes que incluir la librería, crear el objeto y registrar las funciones de la misma forma que hiciste antes:

```
require_once("xajax_core/xajax.inc.php");
$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION,"funcion1");
$xajax->register(XAJAX_FUNCTION,"funcion2");
```

Al registrar una función web, crea automáticamente una función JavaScript en el documento HTML con su mismo nombre prefijado por "`xajax_`". En el caso anterior, se crearán las funciones `xajax_funcion1` y `xajax_funcion2`.

Además deberás utilizar el método `processRequest`, que es el encargado de procesar las llamadas que reciba la página.

```
$xajax->processRequest();
```

Es importante tener en cuenta que la llamada a `processRequest` debe realizarse antes de que el guión PHP genere ningún tipo de salida.

En cada una de las funciones que definas, podrás instanciar y utilizar un objeto de la clase `xajaxResponse` para devolver al navegador los comandos resultado del procesamiento:

Clase xajaxResponse
http://www.xajax-project.org/en/docs-tutorials/api-docs/xajax-core/xajaxresponse-inc-php/xajaxresponse/

```
function funcion1($a){
     $respuesta = new xajaxResponse();
     …
     return $respuesta;
}
```

3.2.- Xajax (II).

Vamos a ver un ejemplo de programación que utilice la librería xajax. Partiendo del formulario web que utilizaste anteriormente, veremos el código xajax necesario para utilizar AJAX en su validación. La idea es crear una función PHP, que llamaremos `validarFormulario`, que reciba los datos del formulario y realice su validación. La página web, al pulsar el botón de enviar del formulario, utilizará AJAX para llamar a esta función y mostrar los errores de validación que devuelva.

En este ejemplo, todo el código va a compartir la misma página, incluyendo las funciones que se ejecutarán mediante AJAX. El primer paso es incluir la librería xajax, y definir la función de validación, que aprovechará el código de validación PHP que creaste antes:

```
require_once("xajax_core/xajax.inc.php");

function validarFormulario($valores) {
    $respuesta = new xajaxResponse();
    $error = false;

    if (!validarNombre($valores['nombre'])) {
        $respuesta->assign("errorNombre", "innerHTML",
"El nombre debe tener más de 3 caracteres.");
        $error = true;
    }
    else $respuesta->clear("errorNombre", "innerHTML");

         // Validamos también las contraseñas y el email
         …
```

```
    if (!$error) $respuesta->alert("Todo correcto.");

    $respuesta->assign("enviar","value","Enviar");
    $respuesta->assign("enviar","disabled",false);
    return $respuesta;
}
```

Como ves, si el nombre no valida, se utiliza el método `assign` de `xajaxResponse` para asignar al elemento de `id=errorNombre` (un `span` que ya está creado) el mensaje de error correspondiente. Si el nombre es válido, se vacía el contenido de ese elemento utilizando `clear` (por si se estuviera mostrando algún mensaje de error anterior). Este procedimiento tendrás que realizarlo para los tres elementos que debes validar en el formulario.

Si no se produce ningún error de validación, se muestra un mensaje de información de que todo está correcto. Además, se utiliza `assign` para volver a habilitar el botón de envío del formulario, y restaurar su etiqueta a `Enviar`(ahora vemos en qué otro lugar se cambia).

Como ya viste anteriormente, antes de crear el contenido HTML de la página, deberás incluir las siguientes sentencias PHP:

```
$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION,"validarFormulario");
$xajax->processRequest();
```

¿En qué páginas debes crear un objeto de la clase xajax?

- En las páginas que vayan a utilizar AJAX para realizar peticiones a otras páginas PHP del servidor
- **En las páginas que vayan a utilizar AJAX para realizar peticiones a otras páginas PHP del servidor, y en las páginas del servidor que vayan a recibir dichas peticiones.**

Ambas páginas necesitan instanciar un objeto de la clase xajax.

3.3.- Xajax (III).

Para ejecutar el código de validación cuando se envíe el formulario deberás crear una función JavaScript y asignarla al evento `onsubmit` del formulario.

```
<form id='datos' action="javascript:void(null);" onsubmit="enviarFormulario();">
```

La función JavaScript `enviarFormulario` utilizará la función `xajax_validarFormulario` que ha sido creada automáticamente por Xajax al registrar la función correspondiente. Además se encargará de deshabilitar el botón de envío y de cambiar el texto que muestra:

```
xajax.$('enviar').disabled=true;
xajax.$('enviar').value="Un momento...";
xajax_validarFormulario (xajax.getFormValues("datos"));
```

Este método de llamar a una función registrada se conoce como asíncrono. Una vez realizada la llamada a la función, el procesamiento del formulario continúa sin esperar a recibir una respuesta. Es por este motivo que se deshabilita el botón de enviar el formulario. Cuando se recibe la respuesta de la función, se producen en el formulario los cambios que se indican en la misma.

Existe en Xajax otro método de realizar llamadas síncronas a una función registrada: el método JavaScript `xajax.request`:

```
respuesta = xajax.request({xjxfun:"validarFormulario"}, {mode:'synchronous', parameters:
["valor1", "valor2", …]})
```

La función PHP a la que se realiza la llamada, debe recibir tantos parámetros como se pasan. Para indicar el valor que se devuelve, puede usar el método `setReturnValue()` de la clase `xajaxResponse`.

```
$respuesta = new xajaxResponse();
…
$respuesta->setReturnValue("valorDevuelto");
return $respuesta;
```

Fíjate que para realizar estas acciones se utiliza el objeto JavaScript xajax. Su método `getFormValues` se encarga de obtener los datos de un formulario.

Clase JavaScript xajax
http://www.xajax-project.org/en/docs-tutorials/api-docs/xajax-js/xajax-core-uncompressed-js/xajax/

Para acceder a los elementos de una página HTML mediante JavaScript mediante su atributo id, se puede usar:

```
objUsuario = document.getElementById("usuario");
```

Y si quieres conocer su valor:

```
valorUsuario = objUsuario.value;
```

Si utilizas Firebug puedes comprobar que los parámetros se envían mediante POST.

Es importante que tengas clara la estructura en clases de la librería Xajax, y sepas diferenciar entre las clases PHP y las clases JavaScript de la misma. Las principales clases son las que has visto en el ejemplo anterior: `xajax` y `xajaxResponse` en cuanto a clases PHP, y `xajax` en cuanto a JavaScript.

3.4.- Xajax (IV).

Esta función, que se encargará de validar el formulario, puedes crearla en el mismo fichero PHP o en un fichero aparte. Deberás incluir también el código necesario para Xajax:

```
<head>
  …
  <?php $xajax->printJavascript();  ?>
  <script type="text/javascript" src="validar.js"></script>
</head>
```

Puedes comprobar el código completo de la página, incluyendo el código JavaScript y los estilos:

form.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con Xajax: form.php
 */

// Incluimos la lilbrería Xajax
require_once("xajax_core/xajax.inc.php");

// Creamos las funciones de validación, que van a ser llamadas
//  desde JavaScript

function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
```

```
    return        ereg('^[a-zA-Z0-9]+[a-zA-Z0-9_-]+@[a-zA-Z0-9]+[a-zA-Z0-9.-]+[a-zA-Z0-9]+.[a-
z]{2,4}$', $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

function validarFormulario($valores) {
    $respuesta = new xajaxResponse();
    $error = false;

    if (!validarNombre($valores['nombre'])) {
        $respuesta->assign("errorNombre",  "innerHTML",  "El  nombre  debe  tener  más  de  3
caracteres.");
        $error = true;
    }
    else $respuesta->clear("errorNombre", "innerHTML");

    if (!validarPasswords($valores['password1'], $valores['password2'])) {
        $respuesta->assign("errorPassword", "innerHTML", "La contraseña debe ser mayor de 5
caracteres o no coinciden.");
        $error = true;
    }
    else $respuesta->clear("errorPassword", "innerHTML");

    if (!validarEmail($valores['email'])) {
        $respuesta->assign("errorEmail", "innerHTML", "La dirección de email no es válida.");
        $error = true;
    }
    else $respuesta->clear("errorEmail", "innerHTML");

    if (!$error) $respuesta->alert("Todo correcto.");

    $respuesta->assign("enviar","value","Enviar");
    $respuesta->assign("enviar","disabled",false);

    return $respuesta;
}

// Creamos el objeto xajax
$xajax = new xajax();

// Registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"validarFormulario");
// Y configuramos la ruta en que se encuentra la carpeta xajax_js
$xajax->configure('javascript URI','./');

// El método processRequest procesa las peticiones que llegan a la página
// Debe ser llamado antes del código HTML
$xajax->processRequest();
?>
<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      HTML      4.01      Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario con Xajax</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
  <?php
  // Le indicamos a Xajax que incluya el código JavaScript necesario
  $xajax->printJavascript();
  ?>
  <script type="text/javascript" src="validar.js"></script>
</head>

<body>
    <div id='form'>
    <!-- Cuando se vaya a enviar el formulario ejecutamos
           una función en JavaScript, que realiza la llamada a PHP -->
    <form id='datos' action="javascript:void(null);" onsubmit="enviarFormulario();">
    <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
            <label for='nombre' >Nombre:</label><br />
            <input type='text' name='nombre' id='nombre' maxlength="50" /><br />
            <span id="errorNombre" class="error" for="nombre"></span>
        </div>
```

```
        <div class='campo'>
            <label for='password1' >Contraseña:</label><br />
            <input type='password' name='password1' id='password1' maxlength="50" />
            <span id="errorPassword" class="error" for="password"></span>
        </div>
        <div class='campo'>
            <label for='password2' >Repita la contraseña:</label><br />
            <input type='password' name='password2' id='password2' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='email' >Email:</label><br />
            <input type='text' name='email' id='email' maxlength="50" />
            <span id="errorEmail" class="error" for="email"></span>
        </div>

        <div class='campo'>
            <input type='submit' id='enviar' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
</body>
</html>
```

validar.js

```
function enviarFormulario() {
    // Se cambia el botón de Enviar y se deshabilita
    // hasta que llegue la respuesta
    xajax.$('enviar').disabled=true;
    xajax.$('enviar').value="Un momento...";

    // Aquí se hace la llamada a la función registrada de PHP
    xajax_validarFormulario (xajax.getFormValues("datos"));

    return false;
}
```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 310px;
  margin-left: -170px;
  height: 330px;
  margin-top: -150px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#form .campo {
  margin-top:8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
```

```
  border : 1px solid #999;
  height:20px;
  width:280px;
}

#form input[type="submit"] {
  width:100px;
  height:30px;
  padding-left:0px;
}

#form .error{
  font-size:10px;
  text-decoration: underline;
  background: #ffdddd;
  color: #ee2211;
}

.oculto {
  display: none;
}
```

Asegúrate de incluir la librería y ajustar las rutas en el código. Si tienes problemas, puedes utilizar Firebug para comprobar que los cambios que realiza Xajax en el HTML son los adecuados.

Utilizando Xajax, ¿cómo debes hacer para que la página web espere por la respuesta de una petición AJAX?

- ○ Llamando a la función JavaScript que crea Xajax cuando registras una función del servidor (su nombre comienza por xajax_).
- ◉ **Mediante el método request, indicando como parámetro mode:'synchronous'.**

La clase JavaScript xajax implementa este método que permite detener la ejecución de código en el navegador (modo síncrono) hasta que se resuelva la petición AJAX.

Partiendo de la aplicación de tienda online que programaste en unidades anteriores, utiliza Xajax para cambiar el mecanismo de login. Se trata de crear una función en PHP de nombre `validarLogin`, que reciba como parámetros un nombre y contraseña, y que compruebe estas credenciales con la base de datos y devuelva `false` si no son correctas o `true` sin son válidas. En este caso, deberá encargarse, también, de almacenar el nombre del usuario en una variable de sesión.

login.php

```
<?php
require_once('include/DB.php');

// Comprobamos si ya se ha enviado el formulario
if (isset($_POST['enviar'])) {

    if (empty($_POST['usuario']) || empty($_POST['password']))
        $error = "Debes introducir un nombre de usuario y una contraseña";
    else {
        // Comprobamos las credenciales con la base de datos
        if (DB::verificaCliente($_POST['usuario'], $_POST['password'])) {
            session_start();
            $_SESSION['usuario']=$_POST['usuario'];
```

```
            header("Location: productos.php");
        }
        else {
            // Si las credenciales no son válidas, se vuelven a pedir
            $error = "Usuario o contraseña no válidos!";
        }
    }
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejercicio: Formulario de Login con Xajax: login.php -->
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 5: Login Tienda Web</title>
  <link href="tienda.css" rel="stylesheet" type="text/css">
</head>

<body>
    <div id='login'>
    <form action='login.php' method='post'>
    <fieldset >
        <legend>Login</legend>
        <div><span class='error'><?php echo $error; ?></span></div>
        <div class='campo'>
            <label for='usuario' >Usuario:</label><br/>
            <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
        </div>
        <div class='campo'>
            <label for='password' >Contraseña:</label><br/>
            <input type='password' name='password' id='password' maxlength="50" /><br/>
        </div>

        <div class='campo'>
            <input type='submit' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
</body>
</html>
```

Modifica la página **login.php** para que, utilizando Xajax, haga una llamada a la función `validarLogin` cuando se pulsa el botón `Enviar` del formulario. Si la función devuelve `false`, mostrará un mensaje que indique que las credenciales son incorrectas. En caso contrario, cargará la página `productos.php`.
Deberás utilizar `xajax.request` para realizar una llamada síncrona a la función. Revisa el código que se propone como solución al ejercicio.

login.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con Xajax: form.php
 */

// Incluimos la lilbrería Xajax
require_once('include/xajax_core/xajax.inc.php');

// Creamos el objeto xajax
$xajax = new xajax('include/valida.php');

// Registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"validarLogin");

// Y configuramos la ruta en que se encuentra la carpeta xajax_js
$xajax->configure('javascript URI','./include/');
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 7 : Aplicaciones web dinámicas: PHP y Javascript -->
<!-- Ejercicio: Formulario de Login con Xajax: login.php -->
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Login Tienda Web utilizando Xajax</title>
  <link href="tienda.css" rel="stylesheet" type="text/css">
  <!-- Incluimos el código JavaScript necesario -->
  <?php $xajax->printJavascript(); ?>
  <script type="text/javascript" src="validar.js"></script>
</head>

<body>
    <div id='login'>
    <!-- Cuando se vaya a enviar el formulario ejecutamos
           una función en JavaScript, que realiza la llamada a PHP -->
    <form id='datos' action='productos.php' method='post' onsubmit='return
enviarFormulario();'>
    <fieldset >
        <legend>Login</legend>
        <div><span class='error'><?php echo $error; ?></span></div>
        <div class='campo'>
            <label for='usuario' >Usuario:</label><br/>
            <input type='text' name='usuario' id='usuario' maxlength="50" /><br/>
        </div>
        <div class='campo'>
            <label for='password' >Contraseña:</label><br/>
            <input type='password' name='password' id='password' maxlength="50" /><br/>
        </div>

        <div class='campo'>
            <input type='submit' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
</body>
</html>
```

valida.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejercicio: Formulario de Login con Xajax: verifica.php
 */

// Incluimos la lilbrería Xajax
require_once('xajax_core/xajax.inc.php');
require_once('DB.php');

// Creamos el objeto xajax
$xajax = new xajax();

// Registramos la función que vamos a llamar desde JavaScript
$xajax->register(XAJAX_FUNCTION,"validarLogin");

// El método processRequest procesa las peticiones que llegan a la página
// Debe ser llamado antes del código HTML
$xajax->processRequest();

// Validamos el nombre y contraseña enviados
function validarLogin($usuario, $password) {
    $respuesta = new xajaxResponse();

    if (empty($usuario) || empty($password))
        // $error = "Debes introducir un nombre de usuario y una contraseña";
        $respuesta->setReturnValue(false);
    else {
        // Comprobamos las credenciales con la base de datos
        if (DB::verificaCliente($usuario, $password)) {
            session_start();
            $_SESSION['usuario']=$usuario;
            $respuesta->setReturnValue(true);
```

```
        }
        else {
            // Si las credenciales no son válidas
            $respuesta->setReturnValue(false);
        }
    }

    return $respuesta;
}

?>
```

validar.js

```
function enviarFormulario() {
    var usuario = document.getElementById("usuario").value;
    var password = document.getElementById("password").value;

   // Aquí se hace la llamada a la función registrada de PHP
    var respuesta = xajax.request({xjxfun:"validarLogin"}, {mode:'synchronous', parameters:
[usuario, password]});
    if (respuesta==false) alert("Nombre de usuario y/o contraseña no válidos.");
    return respuesta;
}
```

3.5.- JQuery4PHP.

La otra librería con la que vas a trabajar se llama jQuery4PHP. Su objetivo es proporcionar un interface de programación en PHP que aproveche las capacidades de la librería de JavaScript jQuery. Es de código abierto, disponible bajo licencias MIT y GPLv2, y funciona únicamente con las últimas versiones de PHP (a partir de PHP5).

jQuery4PHP	**http://jquery4php.sourceforge.net/**

Al igual que con XAJAX, para poder utilizarla en tus páginas simplemente descárgate la última versión desde la página de descargas, y extrae del fichero comprimido la carpeta `YepSua`. Cópiala en un lugar accesible por el servidor web y tus aplicaciones.

Página de descargas	**http://sourceforge.net/projects/jquery4php/files/**

En las **páginas PHP en que quieras utilizar jQuery4PHP**, deberás incluir la librería escribiendo el siguiente código:

```
<?php
require_once("YepSua/Labs/RIA/jQuery4PHP/YsJQueryAutoloader.php");
YsJQueryAutoloader::register();
?>
```

Asegúrate de que la ruta a la librería sea la correcta.

El código anterior ejecuta un método estático de la clase `YsJQueryAutoloader` que se encarga de incluir todos los ficheros necesarios para la librería en el código de tu página.

Para acceder a las capacidades de jQuery4PHP, deberás utilizar en tu código PHP la clase `YsJQuery`.

Antes viste que con XAJAX necesitabas ejecutar en el servidor el método `printJavascript`, para que incluyese su propio código JavaScript en la página que se envía al cliente. La librería jQuery4PHP se apoya completamente en el código JavaScript de la librería jQuery. Por tanto, al escribir el código HTML de la página deberás asegurarte de que se incluye dicha librería, poniendo una línea como la siguiente:

```
<head>
// Utilizamos la versión de jQuery disponible en las CDN de Google
  <script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
  …
</head>
```

En la documentación de la librería, y en los ejemplos que vas a programar a continuación, se utiliza una sintaxis peculiar de programación propia de PHP. Por ejemplo, comprueba el siguiente código:

```
<?php
$jq = YsJQuery::newInstance();
$jq->onClick();
$jq->in('#enviar');
$jq->execute('alert("Has pulsado el botón.")');
$jq->write();
?>
```

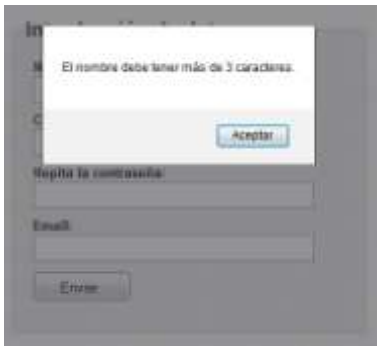
Como ves, se crea un nuevo objeto a partir del método estático **`newInstance`**, y a continuación se ejecutan ciertos métodos implementados en la clase **`YsJQuery`**. Estos métodos definen código JavaScript asociado al evento **`onClick`** del botón con id **`enviar`**. El código que se ejecutará cuando se pulse el botón se incluye dentro del método **`execute`**, y muestra un mensaje en pantalla. El último método (**`write`**) es el encargado de generar el código JavaScript en la página.

Ese mismo código se puede escribir de la forma siguiente, que es la que utilizaremos a continuación.

```
<?php
echo
YsJQuery::newInstance()
  ->onClick()
  ->in('#enviar')
  ->execute('alert("Has pulsado el botón.")')
?>
```

Fíjate que no se asigna nombre al objeto que se crea, pues no es necesario nombrarlo si los métodos se ejecutan justo a continuación de su instanciación. Además, se ha sustituido la llamada al método `write` por un comando `echo` al comienzo.

3.6.- JQuery4PHP (II).



Para comenzar a ver la librería, utilizaremos una versión simplificada del formulario de introducción de datos con el que has estado trabajando. El objetivo es el mismo, validar los datos introducidos por el usuario; pero en lugar de mostrar los errores de validación integrados en la página web, vas a emplear la función `alert` de JavaScript.

La página `form.php` es similar a la que has utilizado anteriormente, añadiéndole código para:

- ✓ Incluir y registrar la librería:

```
require_once("YepSua/Labs/RIA/jQuery4PHP/YsJQueryAutoloader.php");
YsJQueryAutoloader::register();
```

- ✓ Incluir también la librería de JavaScript jQuery:

```
<script                                                    type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
```

Además, hay que capturar el evento **`onClick`** del formulario, e indicarle que envíe los datos del mismo a una página PHP de validación.

```
<?php
   echo
```

```
    YsJQuery::newInstance()
       ->onClick()
       ->in("#enviar")
       ->execute(
             YsJQuery::getJSON(
                 "validar.php",
                 YsJQuery::toArray()->in('#datos input'),
                 new YsJsFunction('
                     if(msg.errorNombre) alert(msg.errorNombre);
                     if(msg.errorPassword) alert(msg.errorPassword);
                     if(msg.errorEmail) alert(msg.errorEmail);','msg'
                     )
             )
       );
?>
```

En el código anterior, primero se asocia al evento `onClick` del botón `enviar` al código que se encuentra en la llamada a `execute`. Este código llama al guión `validar.php`, enviando como parámetros los datos del formulario (los que se han introducido en etiquetas de tipo `input`) convertidos a array y, una vez recibida la respuesta, ejecuta una función JavaScript. En esta función se usa la función `alert` de JavaScript para mostrar los errores de validación obtenidos.

Fíjate que para comunicarse con el servidor, se utiliza el método `getJSON`. Este método utiliza notación JSON (*formato de intercambio de información más sencillo de procesar que XML (especialmente al utilizar el lenguaje JavaScript)*) para transmitir la información con el servidor.

¿Cuál es la función del método write de la clase YsJQuery?

- (●) **Generar el código JavaScript necesario para que la página lleve a cabo las funciones que se han definido.**
- ( ) Mostrar un texto en la página web generada, de forma similar al echo o print de PHP.

Recuerda que no es necesario llamar a este método si empleas una construcción comenzando por echo, como en los ejemplos anteriores.

3.7.- JQuery4PHP (III).

Al ejecutar la página PHP que has programado, puedes observar que se genera el siguiente código JavaScript en la misma cuando se envía al navegador:

```
<script type="text/javascript" language="javascript">
/* <![CDATA[ */
jQuery('#enviar').click(function(){
  jQuery.getJSON('validar.php',
                  jQuery('#datos input').toArray(),
                  function(msg){
                    if(msg.errorNombre) alert(msg.errorNombre);
                    if(msg.errorPassword) alert(msg.errorPassword);
                    if(msg.errorEmail) alert(msg.errorEmail);
                  }
                 )
})
/* ]]> */
</script>
```

Como ves, todo el código creado utilizando la librería jQuery4PHP se traduce en llamadas a la librería jQuery de JavaScript.

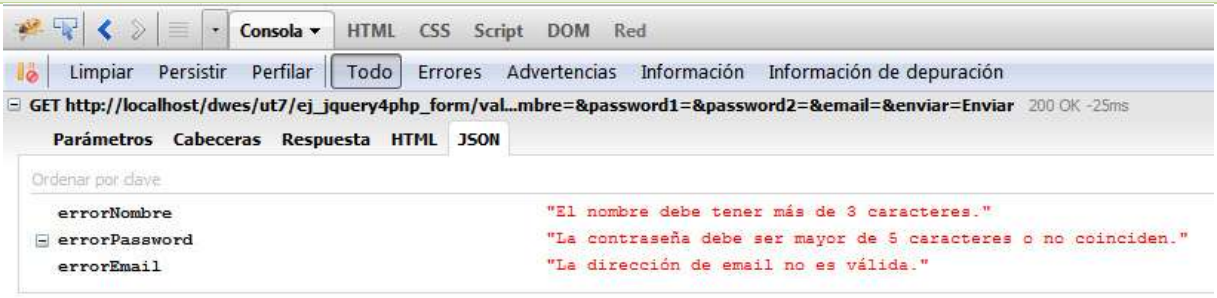
En la página PHP de validación, `validar.php`, puedes utilizar las mismas funciones de ejemplos anteriores y definir una nueva de nombre `validarFormulario`:

```
function validarFormulario($valores) {
    $respuesta = array();
    if (!validarNombre($valores['nombre']))
        $respuesta['errorNombre'] =
"El nombre debe tener más de 3 caracteres.";

    // Validamos también las contraseñas y el email
    …
```

```
    return $respuesta;
}

echo json_encode(validarFormulario($_REQUEST));
```



Utilizaremos la función `json_encode` de PHP para devolver los errores de validación con notación JSON. Revisa el código obtenido y comprueba su funcionamiento.

form.php

```
<?php
/**
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con jQuery4PHP: form.php
 */

// Incluimos la lilbrería jQuery4PHP
include_once('../lib/YepSua/Labs/RIA/jQuery4PHP/YsJQueryAutoloader.php');
YsJQueryAutoloader::register();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario con jQuery4PHP</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
  <!-- Incluímos la librería de JavaScript jQuery -->
  <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
</head>

<body>
    <div id='form'>
    <form id='datos' action="javascript:void(null);">
    <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
            <label for='nombre' >Nombre:</label>
            <input type='text' name='nombre' id='nombre' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='password1' >Contraseña:</label><br />
            <input type='password' name='password1' id='password1' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='password2' >Repita la contraseña:</label><br />
            <input type='password' name='password2' id='password2' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='email' >Email:</label><br />
            <input type='text' name='email' id='email' maxlength="50" />
        </div>

        <div class='campo'>
            <input type='submit' id='enviar' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
<?php
   echo
   YsJQuery::newInstance()
```

```
      ->onClick()
      ->in("#enviar")
      ->execute(
            YsJQuery::getJSON(
                "validar.php",
                YsJQuery::toArray()->in('#datos input'),
                new YsJsFunction('
                    if(msg.errorNombre) alert(msg.errorNombre);
                    if(msg.errorPassword) alert(msg.errorPassword);
                    if(msg.errorEmail) alert(msg.errorEmail);','msg'
                    )
            )
      );
?>
</body>
</html>
```

validar.php

```
<?php
// Creamos las funciones de validación, que van a ser llamadas
//  desde JavaScript

function validarNombre($nombre){
    if(strlen($nombre) < 4) return false;
    return true;
}

function validarEmail($email){
    return      preg_match("/^[a-z0-9]+([_\\.-][a-z0-9]+)*@([a-z0-9]+([\.-][a-z0-9]+)*)+\\.[a-
z]{2,}$/i", $email);
}

function validarPasswords($pass1, $pass2) {
    return $pass1 == $pass2 && strlen($pass1) > 5;
}

function validarFormulario($valores) {
    $respuesta = array();
    if (!validarNombre($valores['nombre']))
        $respuesta['errorNombre'] = "El nombre debe tener más de 3 caracteres.";

    if (!validarPasswords($valores['password1'], $valores['password2']))
        $respuesta['errorPassword'] = "La contraseña debe ser mayor de 5 caracteres o no
coinciden.";

    if (!validarEmail($valores['email']))
        $respuesta['errorEmail'] = "La dirección de email no es válida.";

    return $respuesta;
}

echo json_encode(validarFormulario($_REQUEST));
?>
```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 340px;
  margin-left: -170px;
  height: 300px;
  margin-top: -150px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#form .campo {
```

```
  margin-top:8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
  border : 1px solid #999;
  height:20px;
  width:280px;
}

#form input[type="submit"] {
  width:100px;
  height:30px;
  padding-left:0px;
}

.oculto {
  display:none;
}
```

3.8.- JQuery4PHP (IV).

Una de las características que cabe destacar de jQuery4PHP es su extensibilidad. Existen varias extensiones que se integran con la librería y permiten realizar de forma sencilla tareas adicionales a las que soporta el núcleo de la misma.

Por ejemplo, si en la página anterior quisiéramos integrar en etiquetas los mensajes de validación, el código necesario sería más complejo. De hecho, ya has visto que para mostrar los mensajes de alerta has tenido que utilizar código JavaScript mezclado con el código PHP.

Vamos a ver cómo podemos utilizar la **extensión** `JqValidate` para realizar de forma mucho más sencilla y eficaz la validación del formulario anterior.

Extensión JqValidate

http://jquery4php.sourceforge.net/index.php?section=plugins&module=jqValidate&method=about

Para poder usar esta extensión en tus páginas has de:

- ✓ Indicar a la librería jQuery4PHP que vas a usar la extensión:

```
YsJQuery::usePlugin(YsJQueryConstant::PLUGIN_JQVALIDATE);
```

- ✓ Incluir el código JavaScript necesario por la extensión, que en este caso concreto se corresponde con la **extensión** `Validate` **de jQuery**:

```
<script type="text/javascript"
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.8.1/jquery.validate.min.js"></script>
```

- ✓ Cargar los mensajes de validación en idioma español (de no hacerlo, se mostrarán en inglés):

```
echo YsJQueryAssets::loadScripts('jq4php-showcase/showcase/jquery4php-
assets/js/plugins/bassistance/validate/localization/messages_es.js')->execute();
```

Como siempre, revisa las rutas del código anterior para ajustarlas a las de tu sistema.

3.9.- JQuery4PHP (V).

Al utilizar la extensión `JqValidate` para validar los datos introducidos en un formulario, el código de validación que se usará utiliza la extensión `jQuery.Validate`. No será necesario que programes los

algoritmos de validación, sino que indiques qué reglas se deberán aplicar a cada uno de los campos del formulario. Esto se hace utilizando el método `_rules`.
Para indicar las reglas, se pasa como parámetro un array, que contendrá tantos elementos como campos a validar. Para cada uno de estos campos se crea un nuevo array, que contendrá las reglas de dicho campo. Cada una de las reglas se compone en base a los distintos métodos de validación que incorpora la extensión de JavaScript `jQuery.Validate`.

Métodos de validación
http://docs.jquery.com/Plugins/Validation#List_of_built-in_Validation_methods

El código PHP necesario para validar nuestro formulario será:

```
<?php
   echo
   YsJQuery::newInstance()
      ->onClick()
      ->in("#enviar")
      ->execute(
         YsJQValidate::build()->in('#datos')
           -> rules(array(
'nombre' => array('required' => true, 'minlength' => 4),
'email' => array('required' => true, 'email' => true),
'password1' => array('required' => true,
'minlength' => 6, 'equalTo' => '#password2')
)
)
      );
?>
```

Fíjate que la asociación del código con el evento `onClick` del botón se sigue haciendo como antes.

Si revisas el código JavaScript que se genera en la página HTML, observarás lo siguiente:

```
<script type="text/javascript" language="javascript">
/* <![CDATA[ */
jQuery('#enviar').click(function(){
jQuery('#datos').validate({"rules": {
"nombre": {"required": true,"minlength": 4},
"email": {"required": true,"email": true},
"password1": {"required": true,"minlength": 6,
"equalTo": '#password2'}}})
})
/* ]]> */
</script>
```

Como ves, todo se sigue traduciendo en llamadas a la librería jQuery de JavaScript. Además, una de las grandes ventajas de este método es que una vez definidas las reglas de validación en PHP, todo el código que se ejecuta para verificarlas es JavaScript. Si el navegador soporta la ejecución de código en lenguaje JavaScript, no es necesario establecer ningún tipo de tráfico de validación con el servidor web.

Revisa el código obtenido y comprueba su funcionamiento.
Código obtenido (2.00 KB)

Al emplear la extensión JqValidate de jQuery4PHP, ¿qué código JavaScript deberás incluir en tus páginas?

- (•) **El correspondiente a la librería jQuery y a su extensión Validate.**
- ( ) Únicamente el correspondiente a la libreríajQuery

JqValidate utiliza en la parte cliente la extensión Validate de jQuery.

form.php

```
<?php
/**
```

```
 * Desarrollo Web en Entorno Servidor
 * Tema 7 : Aplicaciones web dinámicas: PHP y Javascript
 * Ejemplo Validación formulario con jQuery4PHP y JqValidate: form.php
 */

// Incluimos la lilbrería jQuery4PHP
include_once('../lib/YepSua/Labs/RIA/jQuery4PHP/YsJQueryAutoloader.php');
YsJQueryAutoloader::register();
// Y el plugin de validación
YsJQuery::usePlugin(YsJQueryConstant::PLUGIN_JQVALIDATE);
?>
<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      HTML      4.01      Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <title>Ejemplo Tema 7: Validación formulario con jQuery4PHP</title>
  <link rel="stylesheet" href="estilos.css" type="text/css" />
  <!-- Incluímos la librería de JavaScript jQuery -->
  <script                                                                type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
  <!-- Y también la de validación -->
  <script                                                                type="text/javascript"
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.8.1/jquery.validate.min.js"></script>
</head>

<body>
<?php
// Cargamos los mensajes de validación en castellano
echo                      YsJQueryAssets::loadScripts('../jq4php-showcase/showcase/jquery4php-
assets/js/plugins/bassistance/validate/localization/messages_es.js')->execute();
?>
    <div id='form'>
    <form id='datos' action="javascript:void(null);">
    <fieldset >
        <legend>Introducción de datos</legend>
        <div class='campo'>
            <label for='nombre' >Nombre:</label><br />
            <input type='text' name='nombre' id='nombre' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='password1' >Contraseña:</label><br />
            <input type='password' name='password1' id='password1' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='password2' >Repita la contraseña:</label><br />
            <input type='password' name='password2' id='password2' maxlength="50" />
        </div>
        <div class='campo'>
            <label for='email' >Email:</label><br />
            <input type='text' name='email' id='email' maxlength="50" />
        </div>

        <div class='campo'>
            <input type='submit' id='enviar' name='enviar' value='Enviar' />
        </div>
    </fieldset>
    </form>
    </div>
<?php
   echo
   YsJQuery::newInstance()
      ->onClick()
      ->in("#enviar")
      ->execute(
         YsJQValidate::build()->in('#datos')
           ->_rules(
            array('nombre' => array('required' => true, 'minlength' => 4),
                  'email' => array('required' => true, 'email' => true),
                  'password1'  =>  array('required'  =>  true,  'minlength'  =>  6,  'equalTo'  =>
'#password2')
               )
            )
      );
?>
</body>
</html>
```

estilos.css

```
#form fieldset {
  position: absolute;
  left: 50%;
  top: 50%;
  width: 310px;
  margin-left: -170px;
  height: 320px;
  margin-top: -150px;
  padding:10px;
  border:1px solid #ccc;
  background-color: #eee;
}

legend, h3 {
  font-family : Arial, sans-serif;
  font-size: 1.3em;
  font-weight:bold;
  color:#333;
}

#form .campo {
  margin-top:8px;
  margin-bottom: 10px;
  margin-left: 10px;
}

#form label {
  font-family : Arial, sans-serif;
  font-size:0.8em;
  font-weight: bold;
}

#form input[type="text"], #form input[type="password"] {
  font-family : Arial, Verdana, sans-serif;
  font-size: 0.8em;
  line-height:140%;
  color : #000;
  padding : 3px;
  border : 1px solid #999;
  height:20px;
  width:280px;
}

#form input[type="submit"] {
  width:100px;
  height:30px;
  padding-left:0px;
}

#form .error{
  list-style: square;
  font-size: 10px;
  color: #e46c6d;
}
```