Tema 3: Generación dinámica de páginas web

¿Qué aprenderás?

- El patrón de diseño MVC utilizado para el desarrollo de aplicaciones web.
- Cómo incluir código de otros archivos y ubicaciones en mi aplicación.
- Cómo invocar una nueva página desde otra con la función header.

¿Sabías que…?

- La arquitectura de una aplicación es la manera en la que sus recursos y componentes se organizan y comunican entre sí.
- En la construcción de aplicaciones web uno de los mecanismos más usados es la arquitectura de capas, que dividen el código en varios niveles.

1. Arquitectura MVC

1.1. Introducción

La arquitectura MVC, siglas de Modelo Vista Controlador, es un patrón de desarrollo de aplicaciones muy utilizado. Se utiliza para cualquier tipo de aplicación, no sólo en entorno web.

Usando este tipo de arquitectura para desarrollar nuestras aplicaciones conseguimos separar la lógica de negocio de la interfaz de usuario. De esta forma favorecemos la funcionalidad, mantenibilidad y escalabilidad del sistema, y además mantenemos "separados" los diversos lenguajes de programación que usamos en el desarrollo de aplicaciones web.

Con la arquitectura MVC dividimos el desarrollo de la aplicación en tres capas: modelo, vista y controlador.

1.2. Capa Modelo

La capa del Modelo representa la lógica del negocio. Es la encargada de acceder directamente a los datos de la base de datos.

1.3. Capa Vista

Esta capa es la que se encarga de mostrar los datos al usuario, es decir, de representar y crear la interfaz gráfica de la aplicación.

1.4. Capa Controlador

El Controlador es la capa intermedia entre el Modelo y la Vista, y se encarga de la comunicación entre ambas capas. Controla las interacciones del usuario solicitando los datos al Modelo y entregándolos a la Vista para que ésta los presente.

1.5. Funcionamiento del patrón MVC

Podemos ver cómo se comunican las tres capas del patrón MVC explicando su funcionamiento resumido en los siguientes puntos:

- El usuario realiza una petición a través de la interfaz gráfica de la aplicación, esto es, la interfaz de usuario.
- El Controlador captura el evento.

- El Controlador hace la llamada al Modelo solicitando unos datos.
- El Modelo se encarga de interactuar con la base de datos, y devuelve información al Controlador.
- El Controlador recibe la información y la envía a la Vista.
- La Vista proceso los datos recibidos y los muestra al usuario construyendo la interfaz gráfica de la aplicación.

2. Archivos externos. Sentencia include

2.1. Introducción

En PHP podemos poner instrucciones en un archivo externo, es decir, en un archivo separado del programa principal, e insertar el archivo donde queramos en el programa, usando la instrucción include.

Podemos tener un archivo que contenga código que se utiliza diversas veces en nuestro programa, e incluirlo donde se quiera usar este código, sin necesidad de tener que reescribirlo. Con la instrucción include conseguimos unos programas más cortos y fáciles de leer.

El formato de un enunciado include es:

```
include("nombre_archivo");
```

El archivo puede tener cualquier nombre. Las instrucciones que contiene el archivo se incluyen tal y como están en el punto donde se usa la instrucción include.

Si el archivo include está en el mismo directorio que el programa, basta con usar el nombre del archivo en la sentencia include. No obstante, si el archivo se localiza en otro directorio, será necesario usar el nombre completo de la ruta del archivo.

2.2. Uso de la sentencia include

La forma más común de utilizar la instrucción include es para los siguientes casos:

- Archivos separados que contienen el código HTML: es común poner los formularios HTML en archivos externos, separados del programa principal. Así pues, cuando queramos mostrar un formulario, usaremos la sentencia include.
- Información de acceso a la base de datos: los datos de acceso a la base de datos de nuestra aplicación es mejor, por razones de seguridad, almacenarlos en variables, y éstas ponerlas en un archivo externo. Un ejemplo de archivo con los datos de la conexión sería:

```
<?php
     $servidor = "localhost";
     $usuario = "root";
     $contraseña = "12345";
?>
```

Debemos poner las etiquetas php porque la sentencia include inserta el archivo como HTML.
El archivo anterior lo incluiremos al principio de cada programa que necesite conectarse a la base de datos. Si cualquier información (como la contraseña) cambia, simplemente tendremos que cambiar la contraseña en el archivo include.

- Funciones: la declaración y el código de las funciones no necesitamos tenerlas en el programa, podemos tenerlas en archivos include. Podemos organizar estas funciones según lo que hagan en varios archivos diferentes (por ejemplo: funciones_basedatos, funciones_formularios). Pondremos el enunciado include al principio de los programas en los que queramos usar las funciones, y ya podremos llamarlas.
- Código común para todos los archivos del sitio web: todas las aplicaciones web tienen código común para todas sus páginas, como por ejemplo cabeceras o pies de página. Podemos poner el código referente a estas partes en archivos include, y llamarlos cuando sea necesario, ubicándolos en la zona deseada de nuestro programa. Por ejemplo, podríamos tener un archivo con el siguiente contenido:

```
<html>
<head><title><?php echo $title ?></title></head>
<body>
<p align="center"><img src="logo.png"/>
```

2.3. Sentencia include_once

La sentencia include_once actúa igual que la sentencia include, salvo que comprueba antes de incluir el código del archivo externo si éste ya se había incluido anteriormente. De esta forma evitamos que se redefinan funciones, se reasignen variables, o errores de este tipo.

El formato de la sentencia include_once es:

```
include_once("nombre_archivo");
```

2.4. Seguridad

Es una buena práctica almacenar los archivos include en un directorio fuera del espacio web, de modo que los visitantes al sitio web no puedan tener acceso a ellos.

Podemos configurar un directorio include donde PHP busque cualquier archivo especificado en el enunciado include. Se puede establecer el directorio include en el archivo php.ini. Para ello buscamos la configuración para include_path y la cambiamos escribiendo la ruta del directorio en que situaremos los archivos include. Un ejemplo de configuración de include_path sería:

```
include_path=".;\include";              #Windows
include_path=".:/usuario/local/include";    #Linux
```

En esta configuración hemos establecido dos directorios donde buscar los archivos include. El primer director es punto (.), es decir, el directorio actual, seguido de la ruta del segundo directorio. Las rutas de los directorios se separan con un punto y coma para Windows y dos puntos para Linux. Podemos especificar tantos directorios como queramos.

Si no tenemos acceso al archivo php.ini, podemos configurar la ruta en cada script individual usando el siguiente código:

```
ini_set("include_path", "d:\directorio");
```

Este enunciado establece la ruta (include_path) hacia el directorio especificado sólo mientras el programa está corriendo. No establece el directorio para todo el sitio web.

2.5. Sentencia require

El enunciado require, y su homólogo require_once, funcionan exactamente igual que include e include_once, salvo que los dos primeros generan un error fatal en caso de no encontrar el archivo especificado.

La página que contenga una sentencia require o require_once cuyo archivo no se encuentre, detendrán la ejecución del script, es decir, no se ejecutarán las siguientes instrucciones que pudiera tener la página. En cambio, con las sentencias include e include_once, se produciría una advertencia, pero el programa seguiría ejecutando las siguientes líneas de código.

La sintaxis de estas sentencias es la siguiente:

```
require("nombre_archivo");
require_once("nombre_archivo");
```

3. La función header

3.1. Introducción

La función header de PHP se utiliza para enviar al servidor web un mensaje diciéndole que envíe una nueva página. De esta forma podemos cargar en el explorador una página sin necesidad de que el usuario clique sobre un vínculo o un botón.

3.2. Sintaxis de la función header

La sintaxis de la función header es la siguiente:

```
header("Location: URL");
```

También podemos añadir un "retraso" a la función header, es decir, que nos envíe a la página de destino después de pasados unos segundos. Esto lo haremos con la siguiente sintaxis:

```
header("refresh:N; url=URL");
```

Por ejemplo, queremos que la página actual nos redirija a la página login.php pasados 3 segundos:

```
header("refresh:3; url=login.php");
```

3.3. Dónde colocar la función header

La función header sólo puede usarse antes de enviar cualquier otro output. No se puede mandar un mensaje solicitando una página nueva en medio de un programa después de haber hecho eco a algún output de la página web. Si lo hacemos, obtendremos el siguiente mensaje de error:

```
Cannot add header information – headers already sent
```

El mensaje también proporcionará el nombre del archivo e indicará qué línea envió el output anterior.

El siguiente código fallará ya que el header no es el primer output, se envían tres líneas de código HTML antes:

```
<html>
<head><title>Página de prueba</title></head>
     <body>
     <?php
          header("Location:
          http://www.empresa.com/inicio.php");
     ?>
     </body>
</html>
```

El siguiente código también fallará:

```
<?php
     header("Location:
     http://www.empresa.com/inicio.php");
?>
<html>
<head><title>Página de prueba</title></head>
     <body>
</body>
</html>
```

Este caso es más complicado de ver. El error proviene del espacio en blanco que hay antes de la etiqueta PHP de apertura. Este espacio vacío es output para el explorador.

Nuestros programas pueden tener tantas instrucciones PHP como queramos antes de una función header, siempre y cuando no envíen outputs. Por ejemplo, podemos usar el siguiente programa:

```
if($edad_usuario<18){
     header("Location: menordeedad.php");
}
else{
     header("Location: catalogo.php");
}
```

Las funciones de session y el enunciado setcookie también deben ir antes de cualquier output. Estos enunciados se verán en el siguiente tema.

Test de autoevaluación

En el patrón MVC, ¿qué capa es la que ve el usuario?

a) Vista
b) Persistencia
c) Modelo
d) Controlador

¿Qué significan los siglas MVC?

a) Modelado-Visualización-Comprobación
b) Modelar-Verificar-Controlar
c) Modeling-Virtual-Component
d) Modelo-Vista-Controlador

¿Qué diferencia hay entre los enunciado include y require?

a) El primer enunciado puede añadir varias veces el código de un archivo externo y el segundo sólo una vez.
b) El primer enunciado se usa para añadir código PHP de un archivo externo y el segundo para cargar otra página.
c) Son dos enunciados que realizan la misma función, pero el segundo produce un error fatal en caso de fallo y el primero una advertencia.
d) Son dos enunciados idénticos, realizan la misma función.

Recursos y enlaces

- Patrón MVC
- Manual oficial de PHP: include y require
- Manual oficial de PHP: header

Conceptos clave

- **Patrón MVC**: siglas de Modelo Vista Controlador, es un patrón de desarrollo de aplicaciones muy utilizado en el desarrollo de aplicaciones que separa la lógica de negocio en tres capas: capa de Modelo, capa de Vista y capa de Controlador.
- **include**: sentencia de PHP que incluye el archivo especificado en aquel que contiene la sentencia.
- **require:** sentencia de PHP similar a include. Se diferencia del otro en que require genera un error fatal en caso de error y detiene la ejecución del programa, mientras que include genera un aviso y sigue con la ejecución.
- **header:** función de PHP que nos permite enviar encabezados sin formato al cliente (navegador).

Ponlo en práctica

Actividad 1

Realiza una aplicación web que pida el nombre y el apellido a un usuario mediante dos campos de un formulario, y muestre en otra página un saludo con los datos introducidos. Deberás crear una función que compruebe que los campos del formulario no están vacíos. Esta función debe encontrarse en un archivo a parte que deberá ser llamado cuando sea necesario.

SOLUCIÓN