Inclusión de Código

Introducción

PHP es fácil de integrar en el HTML. Se puede utilizar cualquier método de producción de HTML ya sólo se tiene que añadir las secciones de PHP más tarde. La inclusión de código de PHP puede ir desde simplemente un solo dígito entero a escribir trozos de código.

Características

Significa incluir un archivo dentro de otro.

Por ejemplo, armar un archivo con el encabezado y otro con el pie de página y después incluir estos dos archivos en cada una de las páginas.

Otra función útil es la de almacenar en un mismo archivo todas las funciones comunes y después incluirlas en lugares donde se necesiten.

Las sentencias para incluir código son: Include y Require. Ambas funcionan igual excepto en el modo de actuar ante un error.

Incluir y Requerir

Require

Incluye y evalúa un archivo. Ante un error produce un error fatal.

```
<?php

    require('archivo_que_no_existe.php');

    echo '<p>Esta linea nunca se imprime</p>';

?>
```

Include

Incluye y evalúa un archivo. Ante un error produce una advertencia (warning).

```
<?php

    include('archivo_que_no_existe.php');

    echo '<p>Esta linea se imprime</p>';

?>
```

require_once e include_once

Tanto require como include, disponen de una variante require_once e include_once. La diferencia con el ejemplo anterior es que PHP sólo incluirá el recurso si no fue solicitado con anterioridad. Esto es muy útil para evitar sobre-escritura de variables o funciones (en el último caso si una función es declarada dos veces PHP arroja un error fatal). También para asegurarnos que ciertos fragmentos sólo se incluyan una vez (por ejemplo, el encabezado de un HTML, ¡no debería existir más de una vez un <head>!)

El siguiente ejemplo incluye por unica vez el THEAD pero incluye tantas filas (TR) como personas haya en el listado.

```
<!-- archivo encabezado_tabla.php -->

<thead>

    <th>Nombre</th>

</thead>
```

```
<!-- archivo fila.php -->

<tr>

    <td><?php echo $nombre; ?></td>

</tr>
```

```
<?php

    //archivo 3

    $personas = array('Roberto', 'Guillermo', 'Dora');


    echo '<table border="1">';
```

```
    foreach($personas as $nombre){
        require_once('encabezado_tabla.php');
        require('fila.php');
    }
    echo '</table>';
?>
```

incluir archivos de forma dinámica

Como se puede notar en los ejemplos, la función include o require toman un String como parámetro. Esto quiere decir que podemos construir en tiempo de ejecución el nombre de un archivo para incluirlo en base a una operación…

```
<?php
    $dia = 'lunes'; //esto puede venir de un formulario o un calculo
    require_once('ofertas_'.$dia.'.html'); // incluye el archivo 'ofertas_lunes.html'
?>
```

buenas prácticas

Una recomendación es utilizar **require_once** a menos que tengamos una buena excusa para no hacerlo, contestándonos la preguntas, ¿por qué me interesa que la programación continúe en caso de no encontrar el recurso? ¿por qué me interesa que este archivo pueda ser incluído más de una vez?.

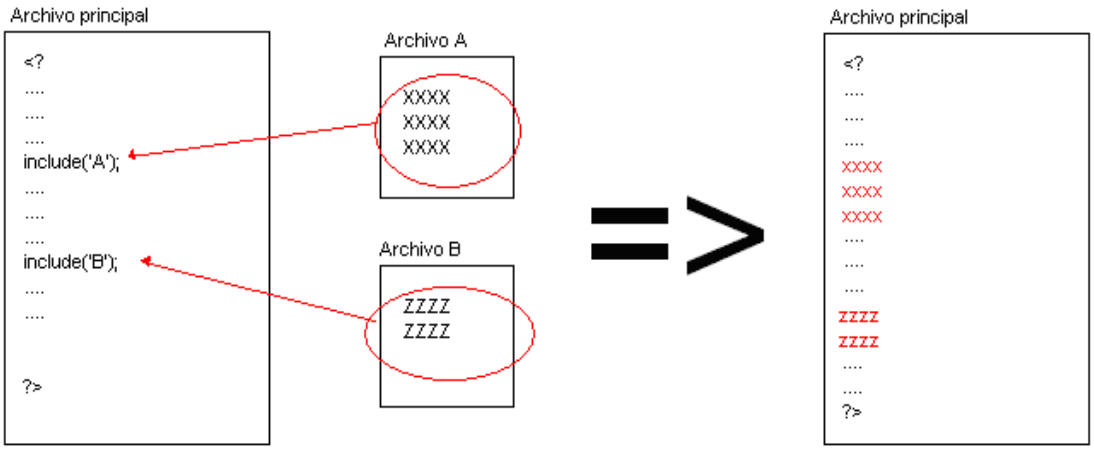
Si estamos incluyendo archivos de forma dinámica, puede ser útil conocer la función **file_exists** para evaluar primero si el recurso existe.

Resumen

Por lo general, todos nuestros script tienen partes de código iguales. Las funciones include() y require() nos van ahorrar muchas de estas líneas de código, permitiéndonos incluir un fichero con código PHP de uso general en diferentes programas.

Es muy útil agrupar un conjunto de funciones generales en un archivo PHP e incluirlo en diferentes páginas y proyectos cuando sea necesario

No solamente se utiliza include() o require() para incluir código PHP, también se utiliza para incluir código HTML, Javascript y cualquier tipo de texto que consideremos necesario incluir dentro de una página.



Encabezados de una página Web

Protocolo HTTP

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un protocolo cliente-servidor que articula los intercambios de información entre los usuarios Web y los Servidores. Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML) es conocido por su URL.

HTTP Headers

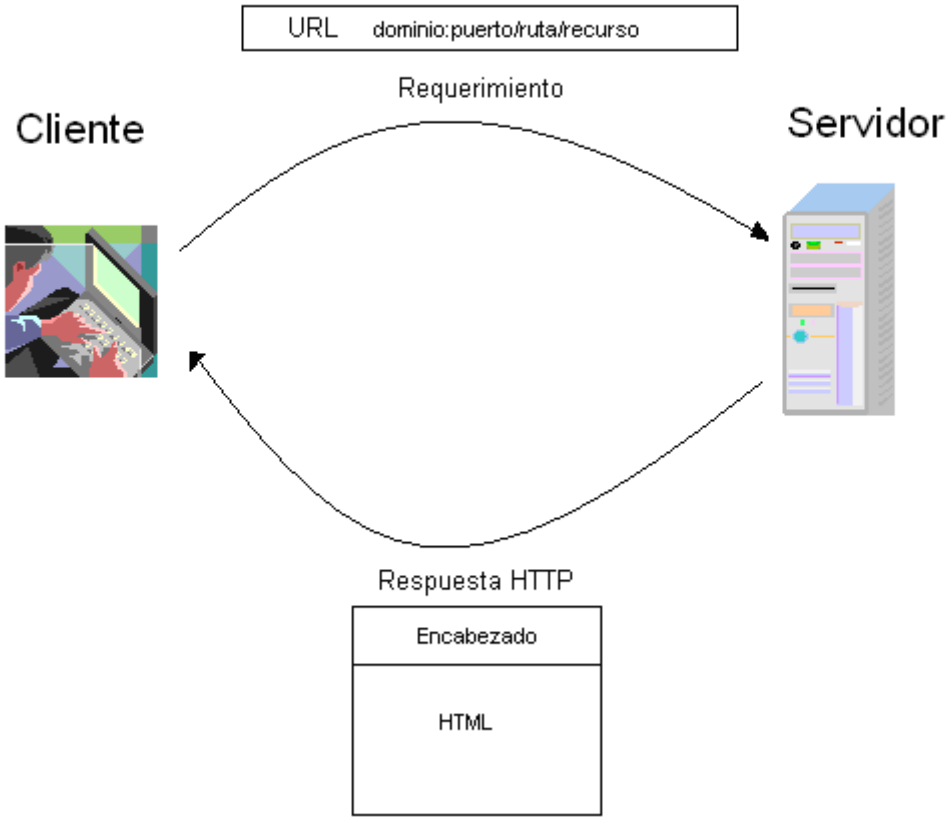
Anatomía de una Página Web

Mucha gente piensa que una página Web no es más que una colección de código HTML. Esto es correcto si se espera ser un Diseñador Web. Pero para un desarrollador PHP (o programador Web en general) los conocimientos deben ser más profundos si usted desea aprovechar mejor todo lo que la Web puede ofrecerle. Desde el punto de vista del Web Server, la generación de un documento, comienza con el requerimiento HTTP iniciado por el Cliente. Una vez que este requerimiento es recibido, el Servidor decodifica los datos que recibe a través del interpretador PHP (asumiendo que el requerimiento fue hecho sobre una página PHP). En otro caso, el Servidor decide manejarlo diferente; en el caso de una página estática (puro código HTML) o una imagen se enviaría directamente al Cliente sin ninguna interpretación previa.

Pero antes de enviar la respuesta, el Servidor escribe un encabezado -response header- que contiene información útil para el Cliente. Tal como: tipo de contenido que se está retornando, o si el mismo está codificado o no, como así también lo datos solicitados por el cliente.

Introducción a los Encabezados

Un cliente (Browser) comienza el circuito de navegación realizando un requerimiento al Servidor Web a través de una dirección URL. El Web Server responde al requerimiento HTTP por el cual su respuesta está compuesta por un conjunto de datos divididos en un encabezado y el contenido de la página. El encabezado contiene información acerca de los datos que se enviaran en el contenido.



No es obligatorio para un programador Web especificar el encabezado. En la mayoría de los casos el encabezado es generado de manera automática por el Web Server haciendo todo esto transparente para el programador. Pero hay momentos en que se puede desear cambiar el encabezado Standard o simplemente agregarle al mismo algunas características suyas. Este es un proceso extremadamente fácil. Todo lo que se tiene que hacer es llamar a la función header() y proveerle el formato apropiado del encabezado.

El único punto critico, es que debe llamar a la función header() antes que cualquier impresión en pantalla, incluso

cualquier carácter de un espacio en blanco que esté fuera de las marcas de PHP (o sea todo tipo de caracter o código perteneciente a HTML). Si se falla en respetar esta regla, dos cosas sucederán: su encabezado no tendrá efecto y PHP devolverá un error en pantalla.

Estructura del Encabezado

El encabezado es una simple cadena de caracteres separada por \n y cada dato que se envía en el mismo esta compuesto por el siguiente formato:

Key: value \n

Key: value \n

Key: value \n

Redireccionamiento

El uso más común del encabezado es el redireccionamiento a otra página.

Header(“Location: http://www.educacionit.com”);

Nótese que la redirección detiene la carga de la página actual y va hacia la nueva página especificada en la función header(). Es por eso que todo lo que se imprima en pantalla antes de redireccionar no tendrá sentido ya que la pagina actual jamás se mostrará.

Cambiar el formato de una pagina

Como se mencionó al inicio de este capítulo, el Web Server genera en forma automática un encabezado de la página perfectamente válido. Esto significa que si un usuario inicia una transacción Web realizando un requerimiento de una página con extensión .HTML, el Web Server responderá a dicho recurso con un encabezado específico para archivos HTML.

En el caso de una página PHP por defecto el encabezado sería también el de un documento HTML.

En el siguiente ejemplo se cambia el encabezado para devolver un documento XML:

```
<?php

header("Content-type: XML");

echo ‘<?xml version=”1.0” ¿>’;

echo ‘<datos>’;

echo ‘<nombre>Juab</nombre>’;
```

```
echo ‘<apellido>Perez</apellido>’;

echo ‘</datos>’;

?>
```

Resumen

Si bien muchas veces el armado del encabezado es un proceso automático, como desarrolladores podemos escribir información para establecer qué tipo de documento se está entregando (por defecto text/html), redireccionar la respuesta, incluir información adicional (como la codificación de caracteres utilizada).

Formularios

Introducción

Un formulario es una sección de un documento que comprende contenido normal, código, controles (checkboxes, cajas de texto, radio buttons, etc.).

Estos formularios son codificados con código HTML y no PHP.

En su uso normal, los usuarios cargan datos en un formulario. Modificando sus controles (introduciendo texto, seleccionando objetos de un menú, etc.).

Luego de completar los datos que requiera el formulario, el mismo es enviado a un servidor Web, o un servidor de correo, etc.

Los controles son los que permiten al usuario interactuar con el formulario.

Cada control tiene tanto un valor inicial como un valor actual.

La etiqueta FORM

Introducción

Un formulario HTML es una sección de un documento que contiene contenido normal, código, elementos especiales llamados controles (casillas de verificación (checkboxes), radiobotones (radio buttons), menúes, etc.), y rótulos (labels) en esos controles.

Los controles son diferentes objetos de captura de datos que les permiten a los usuarios completar determinada información que la página solicite.

Los usuarios normalmente "completan" un formulario modificando sus controles (introduciendo texto, seleccionando objetos de un menú, etc.), antes de enviar el formulario a un agente para que lo procese (p.ej., a un servidor Web, a un servidor de correo, etc.)

Atributos del formulario

METHOD indica cómo se enviarán las respuestas "POST", que es el valor que envía los datos al agente de procesamiento almacenándolos en el cuerpo del formulario, en tanto que "GET" envía los datos agregándolos a la dirección URL y separándolos de la dirección con un signo de interrogación (para aprender más sobre los métodos POST y GET, consulte el artículo sobre protocolo HTTP)

ACTION indica la dirección a la que se enviará la información

ENCTYPE especifica cómo se codifican los datos del formulario. De cualquier forma, esto no necesita especificarse, ya que el valor predeterminado (application/x-www-form-urlencoded) es el único valor válido (este punto se verá en profundidad más adelante, durante el capítulo de UPLOAD).

ACCEPT se usa para establecer tipos MIME para los datos que el formulario puede enviar

onSubmit es un evento que se utiliza para ejecutar código JAVASCRIPT. El mismo se ejecuta cuando el formulario fue enviado.

onResert es un evento que se utiliza para ejecutar código JAVASCRIPT. El mismo se ejecuta cuando el formulario fue reinicializado.

NAME es nombre del formulario para los SCRIPTS. El mismo se utiliza con la ejecución de código JAVASCRIPT.

Controles del Formulario

Los usuarios interaccionan con los formularios a través de los llamados controles.

El "nombre de control" de un control viene dado por su atributo name.

Cada control tiene tanto un valor inicial como un valor actual, que son ambos cadenas de caracteres. Se debe consultar la definición de cada control para obtener información sobre los valores iniciales y las posibles restricciones que puede imponer cada control sobre sus valores. En general, el "valor inicial" de un control puede especificarse con el atributo value del elemento de control

Tipos de Controles:

- Cajas de texto
- Cajas de texto protegidas (estilo password)
- Controles ocultos (hidden controls)
- Casillas de verificación (checkboxes)
- Opciones (Radio buttons)
- Combo box (Select)
- Botones Submit
- Botones Reset
- Botones Genéricos

Gráfico de Controles

Etiqueta	Atributo	Valor	Resultado	Efecto visual
<FORM> ... </FORM>	METHOD	POST GET		
	ACTION		Envía a la dirección mostrada	
	ENCTYPE		Especifica el tipo de código	
<INPUT>	TYPE	submit	realiza la ACTION de la etiqueta <FORM>	Send
		text	línea simple de texto cuya longitud se especifica por el atributo SIZE	
		Reset	Elimina el contenido del formulario	Reset
		Radio	botón de radio	
		Checkbox	casilla de selección	
	NAME		Nombre	
	SIZE		Tamaño del texto	
<TEXTAREA> ... </TEXTAREA>	NAME		Casilla de texto	
	ROWS			
	COLS			
<SELECT> <OPTION> ... </OPTION> </SELECT>	NAME			Opción 1
	MULTIPLE		Múltiples selecciones posibles	Opción 1
<OPTION> ... </OPTION>	SELECTED	Elección predeterminada		Opción 1 Opción 2 Opción 3
	VALUE	Valor forzado		

Recepción de datos

Variables Externas

Cuando se envía un formulario al servidor Web, toda la información que ha sido cargada dentro del mismo por el usuario (en diferentes controles), pasa a almacenarse en variables disponibles para PHP.

Cada una de estas variables adopta como nombre el mismo con el que se nombró el control del formulario.

Dependiendo de la configuración del formulario, existen dos maneras de acceder a los datos que han sido cargados dentro del mismo: POST y GET.

Todos lo tipos de controles tienen una propiedad llamada “name”, que no es otro que el nombre de la variable con la cual recogeremos los datos en el script indicado por el modificador ACTION.

El modificador ACTION indica que archivo PHP, se ejecutará una vez que se han enviado los datos del formulario con la instrucción “Submit” (generalmente, pulsando el botón enviar datos).

Método GET

Este método para las variables (o sea los valores de los controles del formularios) a través de la URL.

Las variables son pasadas como parámetros de la URL. Existe un límite que depende de la combinación navegador / servidor, que podemos normalizar en 1024 bytes.

Método POST

Este método para las variables (o sea los valores de los controles del formularios) de manera invisible, ya que se incluyen en el encabezado del pedido al servidor.

Permite enviar variables con mucha más información, ya que no dispone del limite de GET de ser una gran cadena de texto.

Permite adjuntar archivos en el pedido.

Mitos y leyendas: POST es más seguro que GET

Desde el punto de vista de la seguridad informática no hay ninguna diferencia en el envío de los datos, cualquier hacker que puede interceptar el pedido puede “ver” las variables enviadas. Si nos interesa proteger la comunicación cliente/servidor, deberíamos utilizar un protocolo adecuado, como HTTPS, que implica certificados de seguridad y encriptación de datos.

Desde el punto de vista del usuario, se dice que es más seguro POST ya que las variables “no se ven” en la barra de direcciones. También evita que al refrescar la página se vuelvan a mandar los datos.

Recomendaciones

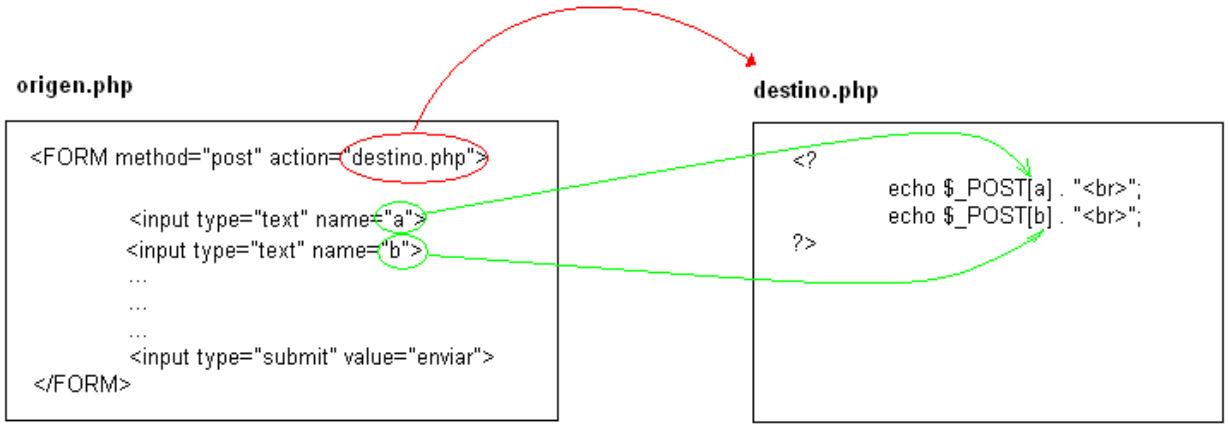
GET y POST son verbos del protocolo HTTP, lo ideal sería utilizar GET cuando el pedido se focaliza en obtener información y POST cuando implica una acción en el servidor (login, edición, inserción, eliminación).
La particularidad de GET de exponer las variables en la URL lo hacen ideal para poder compartir un link que

incluya variables, o guardar en los favoritos, cosa imposible de realizar con POST.
Debemos evitar las fórmulas del tipo “entonces me conviene utilizar siempre…”, ya que como vemos, es una cuestión de contexto.

Captura de Datos de un Formulario

Los datos enviados por un formulario son recibidos en la página destino, especificada en el atributo action de la etiqueta FORM desde la página origen. Estos datos pueden capturarse de diferente manera según el método utilizado en el formulario (GET o POST).

Si no se especifica el atributo action, entonces la página destino será la misma página origen donde se encuentra el formulario.



$_GET Variables proporcionadas al script por medio de HTTP GET.
$_POST Variables proporcionadas al script por medio de HTTP POST.
$_REQUEST Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario (GET o POST)
Esta variable también lee información sobre variables de sesión y cookies (temas que se verán mas adelante). Es insegura y no recomendable, por lo tanto no se puede confiar en ellas.

Ejemplo completo del uso de Formularios

```
<?php

//el archivo debe llamarse Eje01.php

$titulo = 'Recepción de datos a través de Formularios';
```

```
$encabezado = 'Ejemplo de recepción de datos externos con Formularios en PHP';

?>


<html>
<head>
<meta charset="utf-8">
<title><?php echo $titulo; ?></title>
<style>
    h1 {
        text-align: center;
    }
</style>
</head>

<body>

<h1><?php echo $encabezado; ?></h1>

<form name="personas" method="post" action="Eje01.php">
    <p>
        <label>Nombre:</label>
        <!-- un input que no tiene name="" no se incluye como variable -->
        <input name="nombre">
    </p>
    <p>
        <label>Apellido:</label>
        <input name="apellido">
    </p>
    <p><input type="submit"></p>
```

```
</form>

<?php

//siempre debemos chequear que exista al menos una de las variables esperadas, para evitar un Warning undefined index

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

	echo '<p>Su Nombre es: '.$_POST['nombre'].'</p>';

	echo '<p>Su Apellido es: '.$_POST['apellido'].'</p>';

}

?>


</body>

</html>
```

Resumen

Los Formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML (o sea que se ejecutan del lado del cliente) y su principal uso es para que los usuarios puedan cargar datos y a través de la orden “Submit” enviar todos los datos contenidos en el formularios al servidor Web quien los cargara en variables capaces de ser manipuladas por PHP. Los métodos para hacer esta transferencia de datos son GET y POST. El primero pasa los valores de las variables como parámetros junto con la URL. El segundo para los datos de manera invisible.