DESARROLLO WEB EN ENTORNO SERVIDOR

CAPÍTULO 2:

Inserción de código en páginas web

Marcos López Sanz
Juan Manuel Vara Mesa
Jenifer Verde Marín
Diana Marcela Sánchez Fúquene
Jesús Javier Jiménez Hernández
Valeria de Castro Martínez

Ra-Ma®

Lenguajes y tecnologías de servidor

- Servidor web:
  - "*Programa cuya misión última es servir datos en forma de documentos HTML codificados en este lenguaje*".
  - La configuración de este componente software vendrá determinada por la utilización de un lenguaje u otro.
  - El intercambio de datos entre un cliente y un servidor web se hace por medio un protocolo determinado, generalmente HTTP.

Lenguajes y tecnologías de servidor

- Secuencia de comunicación cliente-servidor:

1. El navegador solicita, como cliente DNS, la traducción de una URL a una IP.
2. Se realiza la petición HTTP al servidor web que tenga la IP obtenida.
3. El servidor procesa la solicitud realizada por el cliente.
4. Tras ejecutar el código asociado al recurso solicitado en la URL, el servidor responde al cliente enviando el código HTML de la página.
5. El navegador del cliente, cuando recibe el código, lo interpreta y lo muestra en pantalla.

Lenguajes y tecnologías de servidor

- Tipos de servidores web:
  - **Servidores basados en procesos**. Su funcionamiento se basa en la obtención de un paralelismo de ejecución mediante la duplicación del proceso de ejecución.
  - **Servidores basados en hilos**. La creación de un hilo por parte de un proceso servidor no es tan costosa como la duplicación de un proceso completo.
  - **Servidores dirigidos por eventos.** Utilización de *sockets* (espacios de memoria para la comunicación entre dos aplicaciones que permiten que un proceso intercambie información con otro proceso estando los dos en distintas máquinas, en este caso cliente y servidor).
  - **Servidores implementados en el núcleo del sistema** (kernel). Situan el procesamiento de cada una de las ejecuciones del servidor web en un espacio de trabajo perteneciente al sistema operativo (*kernel*) y no en un nivel de usuario (sobre el sistema operativo).

Lenguajes y tecnologías de servidor

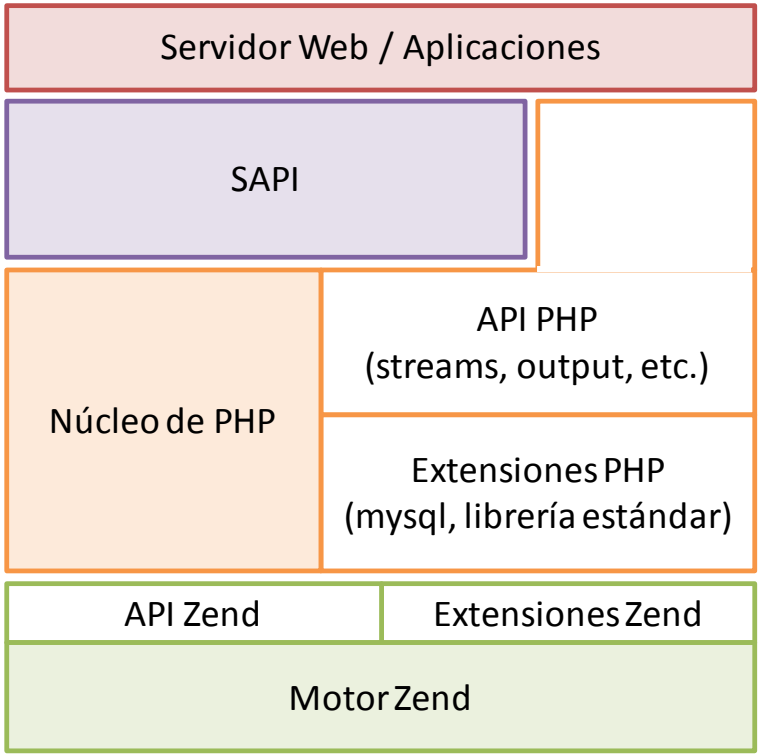
- Ejemplos de servidores web:
  - **Apache Server**. Se trata de un servidor HTTP diseñado para ser utilizado en múltiples plataformas y sistemas operativos.
  - **Microsoft IIS**. Es el servidor web de Microsoft©.
  - **Sun Java System Web Server**. Se trata de un servidor web de alto rendimiento, de escalabilidad masiva y seguro que ofrece contenido dinámico y estático.
  - **Ngnix**. Es un servidor HTTP que ha ganado cuota de mercado en los últimos años (de < 1% en 2007 a casi el 10% a finales de 2011).
  - **Lighttp**. Es un servidor web especializado para entornos en los que se requieren respuestas rápidas.

Obtención del código enviado al cliente

- PHP (*Hypertext Preprocessor*):
  - Lenguaje de *scripting.*
  - De propósito general y de código abierto.
  - Especialmente diseñado para el desarrollo de aplicaciones web.
  - Puede ser embebido (intercalado) en código HTML.

Obtención del código enviado al cliente

- Arquitectura genérica de PHP con Zend:



Obtención del código enviado al cliente

- Arquitectura genérica de PHP con Zend:
  - **Servidor web.** Recibe las peticiones que el cliente hace a una dirección determinada.
  - **Capa SAPI (*Server Abstraction API*).** Donde PHP interactúa con el servidor u otras aplicaciones.
  - **Núcleo de PHP.** Maneja la configuración del entorno de ejecución.
    - Ofrece interfaces de programación (API PHP) tales como la interfaz de entrada/salida estándar, transformación de datos (data parsing).
    - Ofrece un interfaz para cargar extensiones.
  - **Motor Zend**. Componente encargado del análisis y ejecución de las porciones de código PHP (*scripts*).
    - Admite extensiones que permiten modificar completamente su funcionalidad.

Obtención del código enviado al cliente

- Proceso de ejecución de PHP con Zend:

1. El *script* es analizado por un analizador léxico (*lexer*) que transforma el código escrito por el desarrollador en un conjunto de piezas (*tokens*) entendibles por la máquina.

2. Estos *tokens* son pasados después al analizador sintáctico (*parser*).

3. El *parser* toma el conjunto de *tokens* y genera un conjunto de instrucciones (o código intermedio) que es ejecutado por el motor Zend.

4. El componente ejecutor de Zend ejecuta una por una las instrucciones indicadas en el código intermedio.

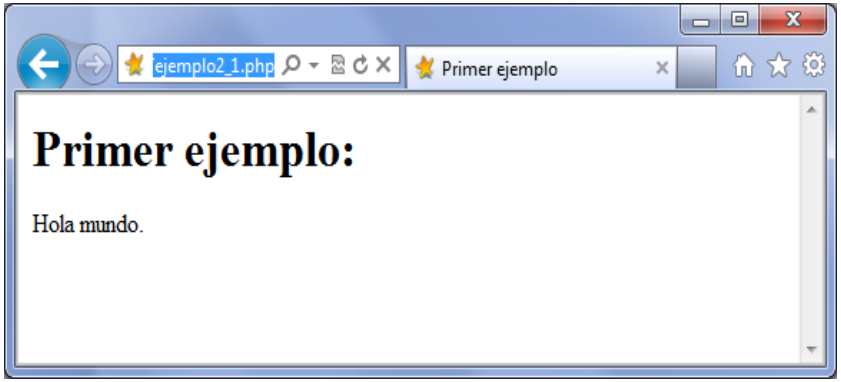
Etiquetas para la inserción de código

- Cada lenguaje que se puede utilizar para insertar código dentro de una página web utiliza una serie de etiquetas para delimitar los fragmentos de código que han de ser procesados por el servidor web.

- El componente del servidor encargado de procesar el código ignorará el código HTML que se encuentra fuera de dichas etiquetas.

Etiquetas para la inserción de código

```
<html>
  <head>
    <title>Primer ejemplo</title>
  </head>
  <body>
    <h1>Primer ejemplo: <br/></h1>
    <?php
      echo "Hola mundo.";
    ?>
  </body>
</html>
```



Etiquetas para la inserción de código

- Características básicas de los lenguajes de *scripting* de código embebido:
  - Todo *script* en comienza y termina con una etiqueta de inicio y otra de fin.
  - Podemos configurar otros estilos de etiquetas en estos lenguajes.
  - Los espacios en blanco que escribamos dentro del código embebido no tienen ningún efecto.
  - El código de servidor embebido en páginas HTML está formado por un conjunto de sentencias que deben estar claramente separadas.
  - Los *scripts* embebidos pueden situarse en cualquier parte del recurso web ejecutado.
  - El número de *scripts* que podemos tener dentro de un fichero HTML es indefinido.
  - Cuando se ejecuta un código embebido, el *script* entero se sustituye por el resultado de dicha ejecución, incluidas las etiquetas de inicio y fin.

Etiquetas para la inserción de código

- Comentarios:
  - Forma de mejorar la legibilidad del código y que se recomienda siempre como una buena práctica a la hora de programar.

- Comentarios de una línea:
  - `// esto es un comentario de una línea`
  - `# esto es otra forma de comentario de una línea`

- Comentarios de múltiples líneas:
  - ```
    /* esta es la forma de
       escribir comentarios
       de varias líneas */
    ```
    ```

Etiquetas para la inserción de código

- Inclusión de código en páginas HTML (PHP):
  - La instrucción `echo` sirve tanto para cadenas de caracteres como para imprimir variables.
    - `echo "ejemplo de impresión de cadena de caracteres";`
  - El resultado de usar `print` sería el mismo:
    - `print "ejemplo de impresión de cadena de caracteres";`
  - También se pueden imprimir números directamente:
    - `echo 215062;`
  - Y mostrar el contenido de una variable:
    - `echo $variableResultado;`
    - `print $variableResultado;`
  - La diferencia entre `echo` y `print` es que la sentencia echo puede imprimir más de un argumento.
    - `echo "imprimir primer argumento", " y el segundo";`

Etiquetas para la inserción de código

- Inclusión de código en páginas HTML (PHP):
  - Otra forma de mostrar el contenido de una variable es utilizando el símbolo igual junto a la etiqueta de inicio.
    - `<?=$variableResultado; ?>`
  - Las sentencias echo y print son funciones y pueden usarse con los parámetros entre paréntesis.
    - `echo ("mensaje");`
  - Cuando utilizamos paréntesis con `echo` y `print` es necesario tener en cuenta el número máximo de parámetros admitidos por cada una de estas sentencias. En el caso de `echo`, el máximo es 1.

Etiquetas para la inserción de código

- Inclusión de código en páginas HTML (PHP):
  - Tanto las comillas simples (‘) como dobles (“) se pueden usar para crear cadenas de caracteres.
    - `echo 'Este texto se mostrará…';`
    - `echo " exactamente igual que éste.“;`
  - Si queremos mostrar comillas simples (o dobles) como parte de la salida, lo que debemos hacer es utilizar comillas dobles (o simples) como delimitadores.
    - `echo "This string has a ': a single quote!";`
    - `echo 'This string has a ": a double quote!';`
  - La impresión de caracteres reservados se hace indicándole al intérprete de PHP que el carácter a continuación de la barra invertida (\) es un carácter que debe imprimirse sin interpretarlo.
    - `echo "Este texto contiene una \" (una comilla doble)";`
    - `echo 'Este texto contiene una \' (una comilla simple)';`
    - `echo "Este texto contiene una barra invertida: \\";`
    - `echo “ y este un símbolo del dólar: \$”;`

Variables

- Definición y uso:
  - "*Almacenes temporales de datos que permiten gestionar los datos utilizados por la aplicación web durante el flujo de ejecución de una página determinada*".
  - Se identifican por el símbolo del dólar (`$`) seguido del nombre de una variable.
  - En PHP, las variables no necesitan ser declaradas explícitamente.
  - Además, no tienen un tipo definido hasta que no se les asigna un valor.
    - `$var = 15;`
    - `$var = "cambio de tipo";`

Variables

- Definición y uso. Reglas de nombrado:
  - El nombre debe comenzar con una letra o con un guión bajo ("_").
  - El nombre únicamente puede contener caracteres alfanuméricos y guiones bajos (a-z, A-Z, 0-9 y _).
  - El nombre de una variable no debe contener espacios en blanco.
  - PHP es *case-sensitive* → las variables `$Variable`, `$variable`, `$Variable` y `$VARIABLE` son variables completamente diferentes.

Variables

- Tipos de datos y variables:
  - Tipos escalares: `boolean`, `integer`, `float` y `string`.
  - Tipos compuestos: `array` y `object`.
  - Tipos especiales: `NULL` y `resource`.
  - Pseudo-tipos: `mixed`, `number` y `callback`.

Variables

- Tipos de datos y variables.
  - Operadores:
    - `$var = (($var - 3) * 4) / 2;`
  - Concatenación:
    - `$var = "cadena" . " unida";`
  - Varias formas de añadir 1 a la variable $var:
    - `$var = $var + 1;`
    - `$var += 1;`
    - `$var++;`
  - Varias formas de multiplicar o dividir:
    - `$var = $var * 2;` → `$var *= 2;`
    - `$var = $var / 2;` → `$var /= 2;`

Variables

- Conversiones entre tipos de datos:
  - Funciones (algunas):
    - `string strval(mixed variable)` → transforma a `string`.
    - `integer intval(mixed variable)` → transforma a `integer`.
    - `float floatval(mixed variable)` → transforma a `float`.
  - Genérica:
    - `settype(mixed variable, string type)`.
    - Parámetro “variable”: valores de tipo `array`, `boolean`, `float`, `integer`, `object` o `string`.
    - Parámetro “type”: cadena de caracteres que indica el tipo al que queremos transformar el parámetro “variable”.

Variables

- Conversiones entre tipos de datos:

Sentencia	Resultado
(int) $var (integer) $var	Conversión a tipo integer.
(bool) $var (boolean) $var	Conversión a tipo boolean.
(float) $var (double) $var (real) $var	Conversión a tipo float.
(string) $var	Conversión a tipo string.
(array) $var	Conversión a tipo array.
(object) $var	Conversión a tipo object.

Variables

- Conversiones entre tipos de datos. Ejemplos:

Valor de $var	(int) $var	(bool) $var	(string) $var	(float) $var
null	0	false	""	0
true	1	true	"1"	1
false	0	false	""	0
0	0	false	"0"	0
3.8	3	true	"3.8"	3.8
"0"	0	false	"0"	0
"10"	10	true	"10"	10
"6 metros"	6	true	"6 metros"	6
"hola"	0	true	"hola"	0

Variables

- Conversiones entre tipos de datos.

Conversión automática:

- Cuando combinamos en una misma expresión dos variables que inicialmente tienen tipos diferentes o cuando pasamos una variable como argumento a una función que espera un tipo de dato diferente.

- $var se convierte a tipo integer con valor 35:
  - `$var = "20" + 15;`

- $var se convierte a string con valor = "20 años“:
  - `$var = 20 . " años";`

- $var se convierte a tipo integer con valor = 20:
  - `$var = 20 + " años";`

- $var se convierte a tipo integer con valor = 42:
  - `$var = 40 + "2 razones";`

Variables

- Comprobación del tipo de una variable:
  - `boolean is_int(mixed variable).`
  - `boolean is_float(mixed variable).`
  - `boolean is_bool(mixed variable).`
  - `boolean is_string(mixed variable).`
  - `boolean is_array(mixed variable).`
  - `boolean is_object(mixed variable).`

Variables

- Estado de una variable:
  - Una variable puede estar en un estado indeterminado (no tener un valor asignado) e incluso puede no haber sido definida (estado indefinido).
  - Funciones para comprobar el estado de una variable:
    - `boolean isset(mixed var)` → comprueba si a una variable se le ha asignado un valor no nulo.
    - `boolean empty(mixed var)` → comprueba si esa variable tiene un valor.
  - Función para pasar una variable a estado “indefinido”:
    - `unset()`.

Variables

- Estado de una variable:

Contenido de $var	isset($var)	empty($var)	(bool) $var
$var = null;	false	true	false
$var = 0;	true	true	false
$var = true	true	false	true
$var = false	true	true	false
$var = "0";	true	true	false
$var = "";	true	true	false
$var = "foo";	true	false	true
$var = array( );	true	true	false
unset ($var);	false	true	false

Variables

- Ámbito de las variables:
  - "*Contexto dentro del que la variable está definida, es decir, la zona del programa en la que puede ser accedida*".
  - **Ámbito local**: las variables internas a una función única y exclusivamente pueden ser utilizadas dentro de dicha función.

```
function duplicar($var){
  $temp = $var * 2;
}
$variable = 5;
duplicar($variable);
echo "El valor de la variable \$temp es: $temp";
```

→ Salida: El valor de la variable $temp es: … y ningún valor para `$temp`.

Variables

- Ámbito de las variables:
  - **Ámbito global**:
    - Sentencia “global”: declara que una variable dentro de una función es la misma que la variable que hemos utilizado (o utilizaremos) fuera de esa función.

```
function duplicar($var){
  $temp = $var * 2;
}
$variable = 5;
duplicar($variable);
echo "El valor de la variable \$temp es: $temp";
```

    - → Salida: El valor de la variable $temp es: 10.
    - La utilización de variables globales sin control puede resultar en un código difícil de mantener y propenso a dar errores.