PRÁCTICA 3.

PHP (I)

OBJETIVOS:

- *Realizar una primera toma de contacto con PHP*
- *Instalar XAMPP (Windows) o LAMPP (Linux)*
- *Reestructurar la página web para que funcione en el lado servidor*

Inicio: 04/04/2015
Entrega: 25/04/2015
Ponderación nota final de prácticas: 30%

DESARROLLO DE LAS PRÁCTICAS

- Sesión 1. Análisis de la estructura arquitectónica.
- Sesión 2. Resolución de dudas en clase.
- Sesión 3. Proposición de mejoras. Resolución de dudas.

EN EL AULA DE PRÁCTICAS

En el aula de prácticas, iniciando la imagen *lsiweb*, tendréis funcionando el servidor Apache, que es el encargado de interpretar las órdenes de los scripts php.

En vuestra unidad **U:** tenéis el directorio u:\xampp\htdocs donde se colgarán los archivos que posteriormente podrán verse en el navegador utilizando la url http://localhost

EN VUESTRO ORDENADOR

Deberéis instalar tanto Apache como PHP, y pensando en prácticas posteriores, también MySQL. La forma más cómoda en Windows es descargando e instalando XAMPP: www.apachefriends.org/es

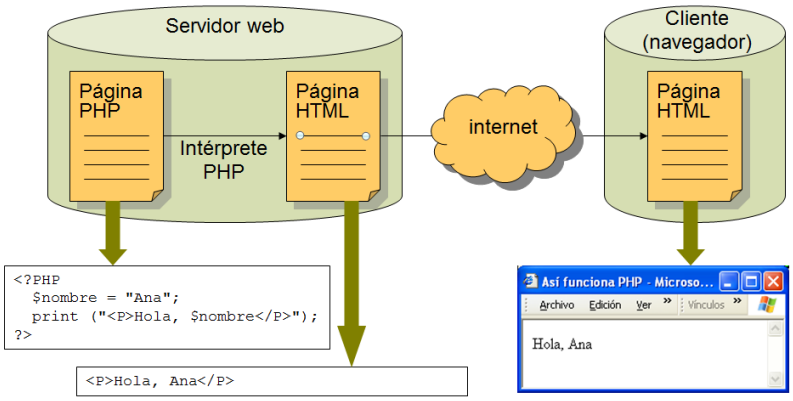
De este software, incluso existe una versión portable para llevarlo siempre en una unidad USB.

En Linux, la versión correspondiente se llama LAMPP, y es fácilmente accesible con Ubuntu u otras distribuciones.

INTRODUCCIÓN

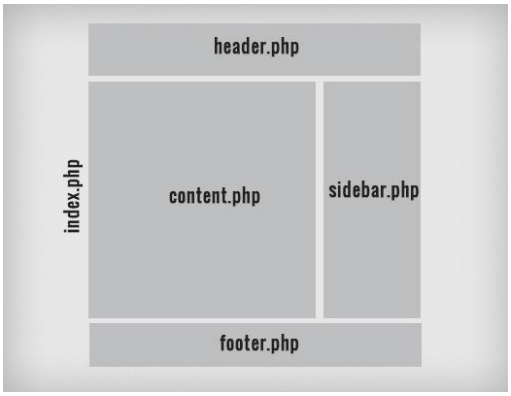
PHP es un lenguaje de script del lado del servidor. Esto quiere decir que no es código compilado, sino que es interpretado.

Los archivos .php son reconocidos como tales por el servidor http (normalmente Apache), que se encarga de ejecutarlos y generar como resultado de la petición HTTP la salida de los script. El cliente no ve el código PHP, sino la salida que los script generan.



PROPUESTA DE DESARROLLO DE LA PRÁCTICA

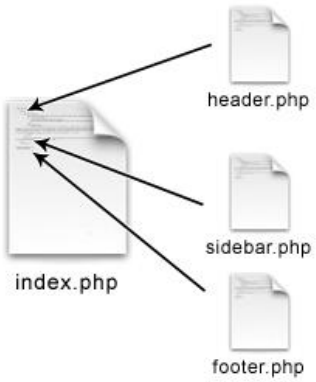
En vuestras páginas webs generadas en la práctica 1 hay numeroso código que se repite, y que se puede delimitar bastante bien. De hecho, en algunas de las prácticas entregadas podemos diferenciar cuatro zonas como las que se muestran en la figura de la derecha.



Si no existe una barra lateral, la incluiremos. En ella colocaremos un menú contextual (que cambiará según en la opción del menú principal que estemos) y el slider de promociones y ofertas, así como la información de contacto con el hotel.

La mejor forma de generar contenido de webs complejas sin tener que repetir código, es hacer que nuestro archivo **index.php** delegue la generación del contenido HTML en distintos scripts, es decir, que index.php u otro script en quien este delegue, actúe como CONTROLADOR:

- **header.php** se encargará de generar la cabecera y el menú;
- **footer.php** se ocupará de generar el pie de página;
- **sidebar.php** se usará para manejar las opciones laterales;
- **content.php** se utilizará para mostrar el contenido principal.



Esto no quita que no haya más scripts para diversas funciones, como envío por e-mail del formulario de contacto, el control de los banner, etc.

¿Cómo se hace que con un único código se generen los distintos tipos de habitaciones o de actividades? En esta práctica 3 lo vamos a hacer de forma estática, sin base de datos.

Para ello hay que hacer uso de las variables GET, tal y como se ve en clase de teoría: http://localhost/index.php?secc=habitaciones invoca al script `index.php` con la variable `secc=habitaciones`. Esto mostraría, por ejemplo, todos los tipos de habitaciones disponibles. `http://localhost/index.php?secc=habitaciones&tipo=3` mostraría exclusivamente la información de las habitaciones triples.

¿Cómo se recupera en PHP el valor de una variable pasada por GET?

Si tenemos un script `saludo.php` y lo invocamos como `saludo.php?name=Javi`

```
<?php
      echo '<p>Hola '.htmlspecialchars($_GET["name"]).'!</p>';
 ?>
```

La ejecución del script devuelve el siguiente código HTML: `<p>Hola Javi!</p>`
Detalles interesantes:

- La salida se genera con la llamada a la función `echo`, que no necesita paréntesis.
- Los string se pueden delimitar con comilla simple o doble (‘ o “).
- El carácter “punto” (.) sirve para concatenar string.
- `$_GET[“variable”]` obtiene el valor de la variable con ese nombre pasado por GET.

Si en lugar de GET usáramos POST, la variable se retomaría con `$_POST[“name”]`.

INCLUDE & REQUIRE

¿Cómo se hace eso de llamar a otros scripts dentro de uno? Utilizando las directivas include y require (más información en http://www.w3schools.com/php/php_includes.asp <-- **Es recomendable leerlo**).

Al contrario que en C, los include suponen la ejecución de ese código (salvo que sean funciones, que se ejecutarán sólo cuando se invoquen). Por ejemplo, podemos tener el archivo `index.php` con lo siguiente:

```
<html>
<body>
<?php
   include 'header.php';
?>
   <h1>Esta es mi página de inicio</h1>
   <p>Hola.</p>
</body>
</html>
```

En este caso, lo que hará será poner justo después del <body> y antes del <h1> la salida que genere `header.php` con sus sentencias `echo`.

Todo código que haya de ser interpretado deberá estar entre **<?php** y **?>**

BIBLIOGRAFÍA BÁSICA

http://www.w3schools.com/php/