50 ANIVERSARIO

IES ANTONIO SEQUEROS ALMORADÍ

TEMA

0

Introducción

2020/2021

Versión: 200913.2108

IES Antonio Sequeros

ÍNDICE

1. INTRODUCCIÓN

1.1. TIPOS DE PÁGINAS WEB

Una sencilla clasificación de los tipos de páginas web podría ser esta:

- Páginas estáticas
- Páginas dinámicas

Páginas estáticas

Diremos que una página es estática cuando sus contenidos **no** pueden ser modificados - ni desde el *servidor* que la aloja (ordenador remoto) ni desde el *cliente* (navegador) - mediante ninguna intervención del usuario ni tampoco a través de ningún programa.

Páginas dinámicas

Llamaremos dinámicas a las páginas cuyos contenidos **sí** pueden ser modificados -de forma automática o mediante la intervención de un usuario- bien sea desde el *cliente* y/o desde el *servidor*.

Para que esas modificaciones puedan producirse es necesario que algo o alguien especifique: qué, cómo, cuándo, dónde y de qué forma deben realizarse, y que exista otro algo o alguien capaz de acceder, interpretar y ejecutar tales instrucciones en el momento preciso.

Igual que ocurre en la vida cotidiana, las especificaciones y las instrucciones requieren: un lenguaje para definirlas; un soporte para almacenarlas y un intérprete capaz de ejecutarlas.

Somos capaces de entender unas instrucciones escritas en castellano pero si estuvieran escritas en búlgaro las cosas seguramente serían bastante distintas, y, por supuesto, a un búlgar@ le pasaría justamente lo contrario.

Igual ocurre con los programas intérpretes de los lenguajes de script. Ellos también requieren órdenes escritas en su propio idioma.

1.2. SCRIPTS

Se llama *script* a un conjunto de *instrucciones* escritas en un *lenguaje* determinado que van **incrustadas** dentro de una *página WEB* de modo que su *intérprete* pueda acceder a ellas en el momento en el que se requiera su *ejecución*.

Cuando se **incrustan** *scripts* en una *página WEB* empiezan a *convivir* en un mismo documento informaciones destinadas a distintos *intérpretes*.

Por una parte, el *código HTML* que ha de ser *interpretado* por el *navegador*, y por la otra, los *scripts* que han de ser *ejecutados* -dependiendo del lenguaje en el que hayan sido escritos- por **su** *intérprete* correspondiente.

La manera de diferenciar los contenidos es delimitar los scripts *marcando* su comienzo con una etiqueta de *apertura* <script> y señalando el final con una etiqueta de *cierre* </script>.

Lo que no está contenido entre esas etiquetas se considerará *código HTML*.

La posibilidad de **insertar** en un mismo documento *scripts* desarrollados en distintos *lenguajes* obliga a especificar cuál se ha utilizado en cada caso, para que en el momento en el que vayan a ser *ejecutados* se invoque el *intérprete* adecuado.

Para ello, dentro de la propia etiqueta de apertura (<script>) se inserta *una referencia* al tipo de *lenguaje* con esta sintaxis:

```
language="nombre"
```

Por ejemplo, el siguiente código indicaría que las instrucciones están escritas con la sintaxis de **PHP**.

```
<script language="PHP">
......
...... instrucciones …
......
</script>
```

Por el contrario, en este otro estaríamos señalando que en las instrucciones contenidas en el *script* se ha utilizado sintaxis de **JavaScript**.

```
<script language="JavaScript">
......
...... instrucciones ..
......
</script>
```

Para el caso concreto de **PHP**, existe una *sintaxis alternativa*, mucho más cómoda y que es la que se usa habitualmente.

Es la siguiente:

```
<?php
......
......instrucciones...
......
?>
```

`<?php` hará la misma función que `<script language="PHP">` y `?>` será equivalente a `</script>`

1.3. LENGUAJES

Hay múltiples posibilidades en cuanto a lenguajes de *script*. Pero antes de hacer mención a algunos de ellos es conveniente hacer una clasificación previa.

Los lenguajes de *script* pueden clasificarse en dos tipos:

- Del lado del cliente
- Del lado del servidor

Lenguajes del lado del cliente

Diremos que un lenguaje es *del lado del cliente* cuando el *intérprete* que ha de *ejecutar* sus *scripts* es **accesible** desde éste -*el cliente*- sin que sea necesario hacer ninguna *petición* al *servidor*.

Seguramente te ha ocurrido alguna vez que al intentar acceder a una *página web* ha aparecido un mensaje diciendo que *la correcta visualización de la página requiere un **plug-in** determinado*, y que, a la vez, se te haya ofrecido la posibilidad de *descargarlo* en ese momento.

Eso ocurre porque cuando el *navegador* -que en el caso de las *páginas web* es el *cliente*- trata de *interpretar* la página, encuentra **incrustado** en ella *algo* (un fichero de sonido, una animación *Flash*, etcétera) que -de forma muy similar a lo que ocurre con los *scripts*- requiere un *intérprete adecuado* del que no dispone en ese momento.

Cuando los *scripts* contenidos en un documento son de este tipo, el *servidor* lo *entrega al cliente* si efectuar ningún tipo de modificación.

Lenguajes del lado del servidor

Un lenguaje es *del lado del servidor* cuando la ejecución de sus *scripts* se efectúa, por *instancia* de este -*el servidor*-, **antes** de *dar respuesta a la petición*, de manera que el *cliente* no recibe el documento original sino el resultante de esa interpretación previa.

Cuando se usan estos tipos de lenguaje el *cliente* recibe un documento en el que cada *script* contenido en el original habrá sido *sustituido* por los resultados de su ejecución.

Esto es algo a tener muy en cuenta, porque, en este caso, los usuarios **no tendrán** la posibilidad de **visualizar el código fuente**, mientras que cuando se trata de *lenguajes del lado del cliente* siempre es posible visualizar los *scripts*, bien sea de forma directa -mirando el código fuente de la página recibida- o *leyendo* el contenido de ficheros externos -vinculados a ella- que son bastante fáciles de encontrar en la *caché* del navegador.

La utilización de este tipo de *scripts* requiere que el *intérprete* del lenguaje sea *accesible* -esté *del lado*- desde el propio *servidor*.

¿Cómo resuelve sus dudas el servidor?

Dado que en unos casos el servidor debe *entregar* el **documento original** -páginas estáticas o páginas dinámicas en las que se usan *lenguajes* del *lado del cliente*- mientras que en otros casos -páginas dinámicas usando lenguajes del *lado del servidor*- tiene que devolver el **resultado** de la ejecución de los *scripts*, es razonable que te preguntes: ¿cómo *sabe* el servidor lo que debe hacer en cada caso?

La respuesta es simple. Eso hay que *decírselo*. Y se le dice de una forma bastante simple. Se indica al poner la extensión al documento.

Si en la *petición* se alude a un documento con extensión **.htm** o **.html** el servidor entenderá que esa página no requiere la intervención previa de ningún *intérprete* de *su lado* y entregará la página *tal cual*.

Si en esa petición se aludiera a una extensión distinta -**.php**, por ejemplo- el servidor entendería que *antes de servir la página* debe *leerla* y requerir al intérprete de PHP que ejecute los scripts desarrollados en ese lenguaje (en caso de que los contuviera) y devolvería al cliente el **documento que resultara** de las eventuales ejecuciones de tales scripts.

Algunos lenguajes con nombre y apellidos

Sin pretender hacer una enumeración exhaustiva, los *lenguajes* de *script* más populares son los siguientes:

Del lado del cliente

- DHTML
- JavaScript

- VBScript
- CSS
- XML

DHTML no es exactamente un lenguaje de programación. Se trata más bien de una serie de capacidades que se han ido añadiendo a los navegadores *modernos* mediante las cuales las páginas pueden contener *hojas de estilo* y/o organizarse en *capas* susceptibles de ser redimensionadas, modificadas, desplazadas y/o ocultadas.

JavaScript es uno de los lenguajes más populares. Cada navegador incluye su propio intérprete y es frecuente que los resultados de visualización sean *algo* distintos según el navegador y la versión que se utilice.

Parece ser que las versiones más recientes de los distintos navegadores se aproximan a un estándar -*ECMA Script-262*- que ha sido desarrollado por la **ECMA** (Asociación Europea de Normalización de Sistemas de Información y Comunicación), lo que hace suponer que en un futuro muy próximo todos los navegadores se ajustarán a esa especificación y que, con ello, las páginas web ya se visualizarán de forma idéntica en todos ellos.

VBScript es un lenguaje de *script* derivado de *VisualBasic* y diseñado específicamente para los navegadores de *Microsoft*.

CSS utilizado para especificar la apariencia que tendrá nuestra página.

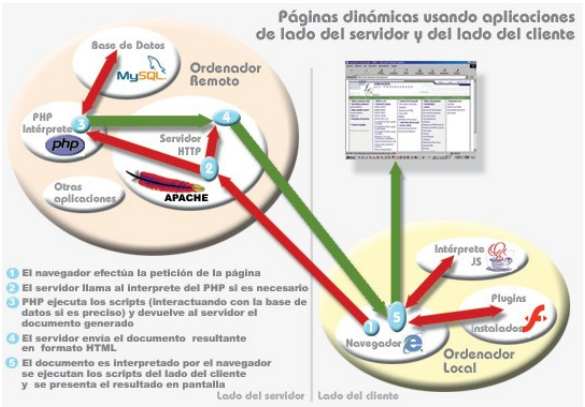
XML, un formato para datos y documentos estructurados, igual que HTML utiliza etiquetas.

Del lado del servidor, los más populares de este tipo son:

- PHP
- ASP
- Perl
- Java
- Python

Cada uno de ellos tiene sus propias peculiaridades. Pero dado que aquí tratamos sobre **PHP** quizá sea conveniente -a modo de recordatorio- hacer algunas precisiones sobre los requisitos imprescindibles para trabajar con este lenguaje.

Requisitos para el uso del lenguaje PHP



De acuerdo a lo comentado en los párrafos anteriores y en los esquemas que tenemos a la derecha, el uso del lenguaje PHP requiere tener *instalado* y *configurado*:

- Un **software de servidor** -configurado para interactuar con el intérprete de **PHP-** que soporte el protocolo HTTP y que en nuestro caso será el denominado servidor **Apache**.
- El intérprete de **PHP**.
- Un software de servidor de **bases de datos** capaz de ser gestionado mediante funciones propias de **PHP**. Utilizaremos el servidor de bases de datos conocido como **MySQL**.

2. SOFTWARE

Para trabajar con Php necesitaremos un servidor Web (Apache), Php y Servidor Bases de Datos (MySQL)

También un editor de textos para escribir el código.

2.1. EDITOR PARA PHP

Va a sernos de gran utilidad el uso un editor de textos que nos permita identificar los números de línea de nuestros scripts y que a la vez nos ayude -mediante resaltado de textos- a depurar la sintaxis.

Existen multitud de editores:

- Php Coder Pro
- Visual Studio Code
- Notepad++
- PhpEdit

Instalación: La instalación de cualquiera de estos editores no presenta ninguna dificultad. Se instala -y se desinstala- de la forma habitual en que se realizan estos procesos en Windows.

2.2. XAMPP

¿Qué es un servidor WEB?

Podríamos definir un servidor WEB como una aplicación que permite acceder a los recursos contenidos en algunos de los directorios del ordenador que la alberga a usuarios remotos que realizan sus peticiones mediante el protocolo HTTP.

Por tanto, *instalar un servidor web* no es otra cosa que *instalar* y *configurar* un *programa* en una unidad o directorio de un ordenador cualquiera.

¿Qué es «Apache»?

Bajo este nombre suele hacerse referencia a **Apache Software Foundation**, organización norteamericana que se autodefine con el objetivo de «... *facilitar ayuda organizativa, legal y financiera para los proyectos de desarrollo de software tipo* **Open Source** (código abierto)».

Uno de los proyectos más populares de **Apache** es el desarrollo y suministro -de forma gratuita y libre- de un **software de servidor HTTP**, conocido también como el **servidor Apache**.

La configuración del Servidor Apache se verá en Servicios de Internet

2.3. PHP

¿Qué es PHP?

El lenguaje **PHP** -acrónimo de **H**ypertext **P**re **P**rocessor- suele definirse como: *interpretado* y de *alto nivel*. El código de sus instrucciones va insertado en páginas HTML (es un lenguaje de script) y es interpretado, siempre, en el servidor.

Se trata un lenguaje de estilo clásico, cercano en su sintaxis a C++.

2.4. Probando el entorno

Ejemplo

Vamos a probar el entorno:

- Crea un documento html (index.html) con un párrafo que muestre el texto «Hola mundo!!»
- Muéstralo con el navegador

Extensión php

- Renombra el documento index.html a index.php y vuelve a mostrarlo con el navegador.
- ¿Hay alguna diferencia en la visualización del documento?
- ¿Y en la forma que el servidor lo procesa?

Nuestro primer código PHP

- El código php siempre va entre los simbolos <?php y ?>.
- Las instrucciones php terminan siempre con ;
- Para generar código html desde php podemos utilizar el método echo pasándole el texto del código que queremos generar.
- Sustituye el texto que hay dentro del párrafo por un bloque de código php que muestre el mismo texto.
- Vuelve a mostrarlo con el navegador.