Unidad Didáctica VIII.- La web como plataforma de desarrollo

CASO PRÁCTICO

El equipo completo de SI Andalucía está trabajando en el último encargo de El PC Feliz, la empresa de mantenimiento hardware que ha confiado en ellos varias veces ya. En esta ocasión les han solicitado el diseño, desarrollo e implantación del sitio web de la empresa. Con él pretenden dar un mejor servicio a sus clientes y captar otros nuevos, además de servir como herramienta interna para los técnicos de la empresa.

Víctor se puso enseguida a investigar sobre este mundo nuevo para él. Carmen se le unió pronto, ya que fue una de las cosas que más le atrajeron en su etapa de estudiante en el IES Aguadulce. Las orientaciones de Carmen fueron una gran ayuda para Víctor, que empezó a darse cuenta de la gran cantidad de facetas que involucra el poner en marcha un sitio web con la funcionalidad requerida en el proyecto.

Una de las recomendaciones de Carmen fue la de montar un sencillo servidor web de pruebas en SI Andalucía para poder ir probando su diseño antes de ponerlo en Internet. Cuanto más se documentan, más claro empiezan a ver que crear un sitio web de calidad implica mucho más trabajo del que pensaban en un principio y además se necesita un equipo multidisciplinar que va desde personas expertas en diseño gráfico hasta especialistas en redes y comunicaciones pasando por programadores y administradores de sistema.

La primera conclusión es que si quieren presentar un buen trabajo tendrán que trabajar duro y repartir las funciones, hasta se han planteado contratar temporalmente los servicios de alguna persona con formación específica en diseño gráfico digital. José dice que puede tener a la persona adecuada, un recién titulado en el Ciclo Formativo de Grado Superior de Imagen que es conocido suyo.

Acuerdan repartir las funciones y nombran a José coordinador del grupo; el equipo está muy ilusionado por este nuevo reto, aunque también están un poco nerviosos por su inexperiencia en este tema.

La unidad anterior te ha servido como introducción a muchos conceptos y tecnologías relacionadas con la utilización de Internet, y más concretamente con la web, para ofrecer a los usuarios prestaciones similares a las que tradicionalmente sólo ofrecían las aplicaciones de escritorio.

Muchas cosas han evolucionado, y en muy poco tiempo, **desde la antigua web** concebida como un sistema de publicación de contenidos enlazados (pero estáticos), **hasta el momento actual** en el que términos como Web 2.0 están haciéndose familiares a cualquier persona. **Jamás una tecnología había cambiado tanto y en tan poco tiempo nuestra forma de entender y relacionarnos con el mundo que nos rodea.**

En esta unidad seguirás aprendiendo cómo dotar de funcionalidad a la web, y sobre todo practicarás lo introducido en la unidad anterior, **además de aprender nuevos conceptos**. Por eso es importante que tengas presentes los contenidos de la unidad anterior para afrontar ésta con seguridad. A continuación te ofrecemos un repaso de lo esencial que debes tener claro para comprender y asimilar lo expuesto aquí.

Generación de aplicaciones para la web (II)

WWW

La World Wide Web, WWW o, familiarmente, "la Web" nació en el año 1990 en el CERN (Consejo Europeo para la Investigación Nuclear). **Su desarrollador,** un físico nuclear llamado **Tim Berners-Lee**, buscaba una forma de publicar y compartir documentos entre sus compañeros científicos basada en hipertexto.Con esa idea creó el primer navegador web y sentó las bases en las que se apoya la actual web.

Estándares que hoy nos resultan familiares como URL, protocolo HTTP **o** lenguaje HTML **tienen su origen en aquellos tiempos** y no han cambiado, en lo sustancial, hasta nuestros días. Sin embargo la web ha evolucionado muchísimo desde que la ideó Tim Berners-Lee. **En sus principios consistía en un sistema de publicación y enlazado por medio de hipertexto de documentos estáticos**, pero en la actualidad a la web se le están dando **nuevos usos y funcionalidades basados en la interacción con el usuario.**

PARA SABER MÁS

En la actualidad Tim Berners-Lee dirige el W3C, consorcio internacional que se ocupa de establecer los estándares en los que se basa la web. Consulta los siguientes enlaces para conocerlo mejor.

World Wide Web Consortium – Wikipedia

Oficina española del W3C

Como curiosidad te interesará saber que Tim Berners-Lee fue nombrado Caballero Comandante de la Orden del Imperio Británico (KBE), por su Majestad la Reina Isabel II el 16 de julio de 2004.

Tim Berners-Lee es nombrado Caballero del Imperio Británico

El estado actual de la informática no se entendería sin la utilización de la web para implementar aplicaciones basadas en ella.

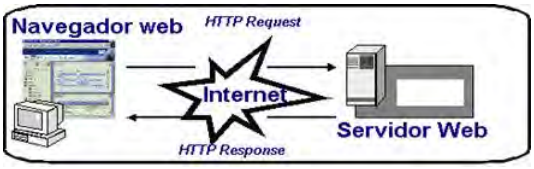
Una aplicación web es un conjunto de programas que residen en un servidor web y a los que se accede desde un navegador web como Internet Explorer o Firefox. En una aplicación web los usuarios van más allá de la simple visualización de documentos estáticos, sino que interactúan con las aplicaciones web desde el navegador y ven los resultados en él.

En este mismo instante estás utilizando una aplicación web diseñada para facilitar la enseñanza a distancia. ¿Te habías dado cuenta?

¿Sabes cómo funciona el sistema web que utilizas a diario? ¿Te has preguntado qué ocurre cuando tecleas http://www.juntadeandalucia.es/educacion/adistancia/ **en la barra de direcciones de tu navegador web?**

Veámoslo de una forma sencilla, es importante que entiendas y conozcas lo que ocurre "detrás" de tu navegador cuando lanzas una petición de página web a un servidor web de Internet.

Como todos los protocolos de Internet, **el protocolo HTTP está basado en la** arquitectura cliente-servidor. Cuando un usuario requiere una página web tecleando una dirección (URL) en su navegador, este lanza una solicitud al servidor donde reside la página deseada (REQUEST) y el servidor contesta a su petición enviándole (RESPONSE) la página solicitada. Puedes ver un esquema simple de este modo de funcionamiento en la siguiente imagen.



PARA SABER MÁS

No es objeto de este módulo profundizar sobre el funcionamiento de los protocolos de Internet, pero es muy interesante que conozcas bien el protocolo HTTP para comprender cómo ocurren las cosas en la web. Te facilitamos un par de enlaces que te darán más información útil sobre el protocolo que utiliza la web

Funcionamiento del protocolo HTTP

Funcionamiento del protocolo HTTP

Autoevaluación

El protocolo que se utiliza en la web es...

a) HTTP
b) HTML
c) URL
d) Ninguno de los anteriores

Comprobar

Es importante que entiendas cómo funciona la arquitectura cliente-servidor (Navegador Web-Servidor Web) del protocolo HTTP, puesto que las aplicaciones web, objeto de esta unidad, están basadas en él.

Para terminar, citaremos un pensamiento de Tim Berners-Lee en su discurso de investidura como Caballero del Imperio Británico:

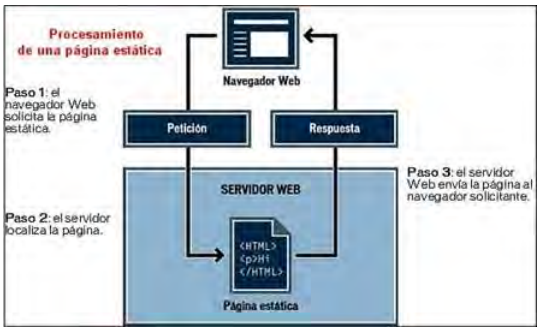
"La Web debe permanecer como un medio universal, abierto a todos y que no sesgue la información que transmite. Ya que la tecnología se vuelve cada vez más potente y, al mismo tiempo, está cada vez más disponible, con el uso de una gran variedad de dispositivos, espero que aprendamos a usarla como medio para trabajar juntos y resolver desacuerdos a todos los niveles".

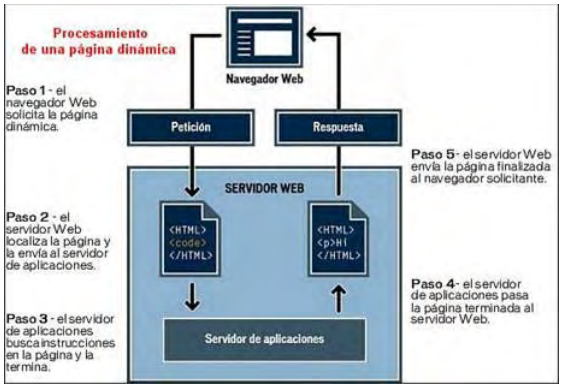
Páginas estáticas vs. Páginas dinámicas

Ya sabes que **en un principio la web sólo servía** (y no era poco) **para almacenar, relacionar y recuperar documentos estáticos** que estaban almacenados en servidores. Esos documentos **estaban escritos en un lenguaje de marcas de hipertexto llamado HTML** (HyperText Markup Language). **El lenguaje HTML no es un lenguaje de programación, sino de formateo de texto;** no permite emplear estructuras de control, ni permite definir variables, ni cualquier otro concepto de los que utilizamos cuando programamos en un lenguaje de programación como Java o C++.

Pronto se vio que esto no era suficiente, y **se empezaron a buscar alternativas y añadidos técnicos que dotaran** a la idea estática inicial de la web **de interactividad con el usuario,** de forma que **lo mostrado en el navegador web** en un momento dado tuviera un carácter **dinámico,** es decir, dependiese de los datos de la petición reproduciendo el **esquema básico de la programación: ENTRADA-PROCESO-SALIDA.**

Para aclararte la **diferencia de procesamiento entre páginas estáticas y dinámicas** te presentamos lo que ocurre en ambos casos en forma de esquema de bloques.





Fíjate bien en un detalle importante, en los dos casos la petición y la respuesta se basan en el mismo y antiguo protocolo HTTP. Los navegadores web sólo entienden lenguajes de presentación (HTML y sus añadidos), el código de programación se encuentra en el servidor web y una parte de éste (el servidor de aplicaciones web) es quién procesa las instrucciones contenidas en la página web para generar el resultado (página web) que posteriormente se envía de vuelta al navegador. Todo el proceso queda encapsulado en el servidor web, y más concretamente en el servidor de aplicaciones web que forma parte de él. Al navegador sólo llega HTML, nunca código de programa.

Autoevaluación

Señala qué afirmaciones son correctas:

a) HTML es un lenguaje de formateo
b) HTML permite estructuras de control de programación
c) HTML se diseñó para poder crear páginas dinámicas
d) HTML significa HyperText Markup Language

Comprobar

Con un ejemplo lo entenderás mejor

Vamos a mostrarte con ejemplos reales la diferencia entre páginas estáticas y dinámicas. Seguro que después de ver y practicar el ejemplo que te proponemos entenderás mucho mejor todo lo anterior. Te proponemos además que analices los sitios web que visitas a menudo y distingas aquellos que son dinámicos y los que no.

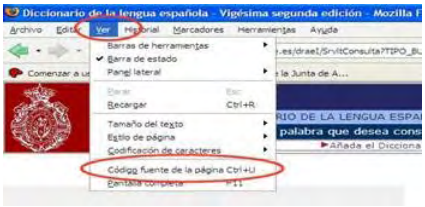
¿A que cuesta trabajo encontrar hoy día un sitio web que no sea dinámico?

Veamos... si tecleas en la barra de direcciones de tu navegador **la dirección** http://www.un.org/spanish/aboutun/hrights.htm **obtendrás el texto de la Declaración Universal de los Derechos Humanos.** Por más veces que accedas a esta página siempre la verás igual, y además no encontrarás ningún elemento que te permita interactuar con ella, **es un ejemplo de página estática**. Haz clic sobre la imagen, o sobre el enlace anterior, para ver la imagen ampliada.

Si ahora tecleas **la dirección** http://www.rae.es/ **accederás al sitio web de la Real Academia Española de la Lengua en Internet,** salta a la vista que en este caso sí encontramos elementos que permiten interactuar con ella, en concreto disponemos de una caja de entrada de texto que permite introducir una palabra y... ¡Obtener su definición según el diccionario! **En este caso tenemos una página dinámica.** Haz clic sobre la siguiente imagen para verla ampliada, o directamente usa el enlace anterior para ver la página.

Habitualmente los servidores de aplicaciones web están conectados a bases de datos, de otra forma sus posibilidades serían muy pobres. **En el ejemplo anterior es obvio que la página de la Real Academia Española de la Lengua accede a una base de datos en la cual se encuentra almacenado el diccionario**, pero lo que llega al navegador web es solamente código de presentación HTML. Si viésemos ese código no lograríamos apreciar ni rastro de acceso a tablas de una base de datos, y sin embargo es así como funciona.

¿Sabías que podemos ver el código HTML recibido por nuestro navegador? Todos los navegadores web lo permiten, en el caso de Firefox se puede conseguir con la opción <Código Fuente de la Página> del menú <Ver>. **Pruébalo y comprueba cómo aparecen las etiquetas HTML** de la página que estás viendo, el navegador las interpreta para producir el resultado que ves en la ventana del navegador.



Cuando termine esta unidad serás capaz de desarrollar tus propias aplicaciones web con acceso a base de datos.

PARA SABER MÁS

Te facilitamos un artículo que profundiza sobre el tema de las páginas web estáticas y dinámicas, encontrarás en él información que complementa lo anterior.

Web estática vs Web dinámica

Autoevaluación

Señala qué afirmación es correcta

a) Las páginas dinámicas no necesitan HTML
b) Las páginas estáticas no se pueden cambiar una vez que están escritas
c) Tim-Berners Lee diseñó las primeras páginas dinámicas en el CERN
d) Las páginas dinámicas permiten al usuario interactuar con ellas

Comprobar

CASO PRÁCTICO.

*Después del reparto de tareas, **Jesús**, por su experiencia como administrador de sistemas, ha resultado ser el encargado de montar un servidor web de pruebas en la red local de **SI Andalucía**. La primera decisión ha sido elegir cuál montar. Después de estudiar las distintas posibilidades se decide por un servidor web Apache en un PC con Linux que estaba en desuso.*

Pronto se dan cuenta que también necesitarán un servidor FTP y otro de correo electrónico. Cada vez tienen más claro la web es un campo muy prometedor, pero que involucra gran cantidad de tecnologías muy diversas y diferentes, y que exigen mucha especialización por parte de las personas que trabajan en ella.

Ahora ya pueden empezar a dar sus primeros pasos como desarrolladores web, y cuanto más conocen, más sorprendidos quedan de todo lo que descubren. Hasta ahora no habían pasado de usar Internet como usuarios avanzados, pero esta oportunidad les está haciendo ver unas posibilidades que tan sólo intuían vagamente.

En la unidad anterior has tomado contacto con la tecnología web, tan importante en la actualidad. Ya sabes que **en el principio sólo existían páginas estáticas** que proveían contenidos con escasa o nula interacción con el usuario.

Mucho ha cambiado desde entonces lo que las tecnologías web pueden ofrecer al profesional informático y por lo tanto al usuario. Si hoy día podemos operar con nuestra cuenta bancaria, contratar las vacaciones, comprar en nuestro supermercado favorito o estudiar formación profesional sin movernos de casa, es gracias a la web y en concreto

gracias a las aplicaciones basadas en la web.

Hasta ahora hemos hablado mucho de la web y las aplicaciones basadas en ella, el protocolo HTTP, el lenguaje de marcas HTML, las páginas dinámicas conectadas a bases de datos y un buen número de tecnologías puestas a trabajar de forma conjunta para alcanzar el objetivo de dotar de funcionalidad a la web más allá de la simple publicación de contenidos, **pero hemos pasado casi de puntillas por la pieza clave que sustenta a todo lo demás, nos estamos refiriendo a los servidores web.**

Seguramente habrás oído hablar en multitud de ocasiones de servidores web, pero... **¿Conoces de verdad lo que son y cómo funcionan?**

En este apartado entraremos en detalle a estudiarlos y te enseñaremos a instalar uno para que puedas poner en práctica todo lo aprendido hasta ahora en esta unidad y en la anterior.

Debemos dejar claro sin embargo, que **el estudio en profundidad de la instalación, puesta en marcha, configuración y mantenimiento de un servidor web para servir páginas de manera profesional e intensiva con requerimientos de seguridad, robustez y eficiencia, es una tarea compleja reservada a administradores de sistema**, y está fuera de los objetivos de este módulo profesional.

Por lo tanto nos conformaremos aquí en darte las pautas para poder poner en funcionamiento un servidor web de pruebas para ejecutar las prácticas que te iremos proponiendo a lo largo esta unidad y te animamos a que profundices en el tema de servidores web, es apasionante y muy actual.

PARA SABER MÁS

Si deseas empezar a saber más sobre servidores web, su instalación, configuración y explotación te proponemos algunos enlaces de interés.

Servidor web casero

Como montar un servidor con XAMPP

¿Qué es un hosting?

Instalación de un servidor web

Seguro que en este momento estás deseando empezar a poner en práctica todo lo que has estudiado hasta ahora. Como ya sabes, lo primero que necesitamos para empezar a practicar es disponer de un servidor web. **Podríamos utilizar alguno de los muchos hostings que podemos encontrar en la red hoy día** (tanto gratuitos como de pago), estos servicios están muy extendidos y son fáciles de utilizar. **Pero creemos que es mucho mejor que lo montes en tu propio equipo para que comprendas todo lo que hay detrás de un sitio web de Internet y qué queda oculto al usuario común.**

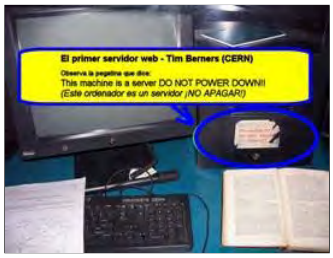
¿Empezamos?

El servidor web más extendido. Apache

Como ya sabes, el primer servidor web fue el desarrollado por Tim Berners en el CERN, se albergaba en su propio ordenador personal. **Eran otros tiempos pero en realidad no hace tanto.** Te mostramos una foto de aquel ordenador corriendo el primer servidor web, curioso ¿verdad?

Hoy día los servidores se montan en ordenadores diseñados específicamente para ello y en unas instalaciones diseñadas para responder 365x24**, con infraestructura adecuada de comunicaciones, seguridad, fiabilidad, robustez, tolerancia a fallos, etc**.

Compara las dos fotos siguientes, entre ellas hay un abismo tecnológico pero sólo unos pocos años de diferencia.



PARA SABER MÁS

Un muy interesante artículo de una de las personas que más pronto vislumbró en España las posibilidades de la web, el profesor de la Universidad Jaume I Jordi Adell

Arqueología digital: Los primeros servidores web de España

No olvides que lo que ves en las fotos anteriores es sólo la parte hardware del servidor web **¿Qué pasa con el software del servidor?**

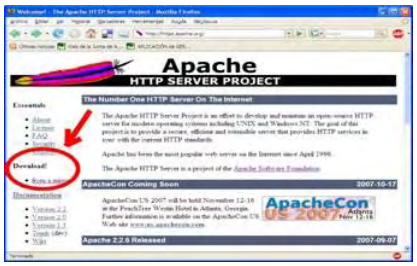
A principio de la década de los 90 en el NCSA (National Center for Supercomputing Aplications) **se empezó a desarrollar el servidor web NCSA HTTPd. Más tarde,** tomando como base ese servidor, **se empezó a desarrollar el Apache HTTP Server**. Éste se convirtió pronto en **el servidor más popular**, y todavía hoy sigue siendo el de mayor implantación. Las características que le han hecho imponerse a los demás son:

1. **Arquitectura Modular:** Sobre la base del servidor se pueden montar módulos complementarios que ofrecen funcionalidades extra.
2. **Software de código abierto (Open Source):** Con todo lo que ello significa en cuanto a transparencia, extensibilidad y soporte. Además de hacerlo gratuito.
3. **Multi-plataforma:** Existen versiones de Apache para casi todos los sistemas operativos y arquitecturas disponibles en el mercado.

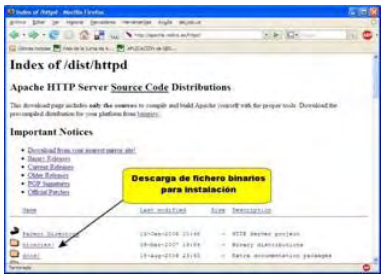
> ***PARA SABER MÁS***
>
> ***Sigue estos enlaces sobre el vetusto servidor NCSA y su sucesor Apache, junto con una interesante comparativa de la implantación de actual de los diferentes servidores web en el mercado.***
>
> ***Servidor NCSA***
>
> ***Servidor web Apache***
>
> ***Comparativa de implantación de servidores***

Para nuestras prácticas utilizaremos Apache ¿Nos ponemos manos a la obra?

Lo primero es descargar de la página oficial del Proyecto Apache http://www.apache.org/ la versión adecuada a nuestra plataforma, observarás que existe entre otras para Linux y para Windows.



Podemos descargar el código fuente o la versión para instalar (binary), además de documentación, versiones anteriores y parches oficiales.



¿Lo has descargado ya? Pues vamos a instalarlo...

Autoevaluación

Señala qué afirmaciones son correctas.

a) Apache es open source
b) Apache puede aumentar su funcionalidad con módulos externos
c) Apache es un programa para el sistema operativo Windows
d) Apache nació a partir del NCSA HTTPd

Comprobar

Instalación y configuración de Apache

Si ya has descargado el fichero con la instalación para tu sistema puedes empezar con la instalación. **Hacer una instalación básica no es nada complicado, aunque se pueden configurar múltiples parámetros y ajustes para adaptarlo a nuestras necesidades y requerimientos**. Te facilitamos una presentación que recorre todos los pasos del asistente de instalación en Windows.

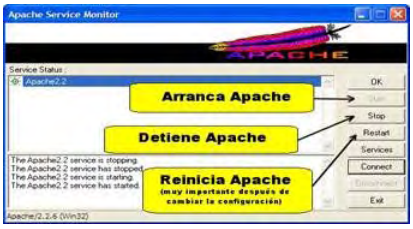
Instalación del servidor web Apache

¿Todo bien? ¿Has comprobado que apache está instalado en tu sistema? ¿Ya has comprobado que **"hay algo" detrás de tu dirección de bucle local? ¡Enhorabuena!**

Habrás observado también que aparece un pequeño icono en la barra de direcciones indicando que Apache está funcionando y "escuchando" en el **puerto 80 del TCP/IP** de tu sistema. Desde ese icono podemos abrir el monitor de Apache.

También puedes comprobar que dispones de una nueva entrada en el grupo de programas. Desde ella también es posible arrancar

(Start), parar (Stop) o reiniciar (Restart) el servicio de Apache.



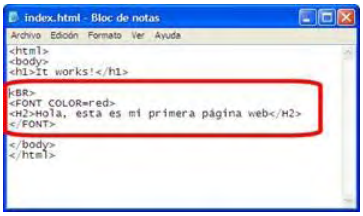
Puedes ver y editar el fichero de configuración httpd.conf. No hay más que verlo para darse cuenta de la cantidad de parámetros configurables que existen y la complejidad que podemos llegar a alcanzar, pero... **¡Cuidado! No introduzcas cambios sin saber lo que haces, podrías estropear la instalación de Apache.**

Nuestra primera página

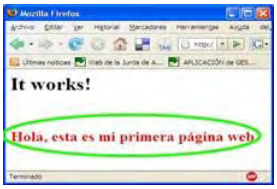
Vamos a empezar a practicar creando nuestra primera página web servida desde nuestro recién instalado servidor web. **Apache sirve los ficheros situados en el directorio indicado por la variable DocumentRoot del fichero de configuración httpd.conf.** En nuestro caso DocumentRoot es "C:/Archivos de programa/Apache Software Foundation/Apache2.2/htdocs".

Si examinas el contenido ese directorio comprobarás que **existe un fichero llamado index.html. Apache sirve por defecto este fichero cuando requerimos en un navegador la URL del sitio web**, si lo abres verás que contiene marcas HTML (recuerda la unidad anterior).

Hagamos una prueba simple, **edita el fichero index.html con algún editor de texto plano y añade algún contenido** (no olvides que debe ser HTML). Por ejemplo si añadimos las siguientes líneas:



Grabamos los cambios, tecleamos como URL **http://localhost** en nuestro navegador y...



Etiquetas, etiquetas y más etiquetas

En la unidad anterior te presentamos los aspectos esenciales de HTML, en particular muchas de las etiquetas más utilizadas y su función. **Recuerda que HTML es un lenguaje de marcas y formateo, orientado a definir la presentación de documentos en un navegador web**. Existen multitud de etiquetas con sus correspondientes parámetros, te proponemos que practiques con ejemplos el lenguaje HTML hasta que te familiarices bien con él.

Ya tienes todo lo necesario a tu alcance, un servidor de páginas web (Apache) y un editor de texto plano, no se necesita más ¿O sí...? Pronto te darás cuenta de que **resulta casi imposible**, y tremendamente farragoso, **componer documentos web** de cierta complejidad tecleando directamente las etiquetas HTML **en un editor de texto plano**. Pero eso lo veremos después, ahora te toca practicar.

> ***PARA SABER MÁS***
>
> ***En los siguientes enlaces tienes un buen tutorial de HTML y una gran cantidad de ejemplos que puedes probar en tu propio servidor Apache. Es muy importante que te acostumbres al proceso de edición-visualización, puesto que todo lo que se mueve por la web es HTML y las aplicaciones web lo utilizan para construir el interface de usuario en el navegador web.***
>
> ***Tutorial de HTML***

Muchos ejemplos de HTML clasificados en categorías

¿Has probado todas las etiquetas? Son muchas, ¿verdad?

Te proponemos a continuación un ejercicio que seguro que resolverás sin problema, si has estudiado con atención el lenguaje HTML.

Intenta escribir un documento HTML que produzca la siguiente página web al cargarla en un navegador web. Haz que funcione en el servidor Apache que tienes instalado.

Recuerda que Apache sólo puede servir páginas y ficheros que estén dentro del directorio indicado por la variable DocumentRoot del fichero de configuración **httpd.conf**. Por lo tanto en ese directorio deberás situar el documento HTML y las imágenes que formen parte de la página. Para ello puedes utilizar las imágenes contenidas en el siguiente enlace:

Descarga las imágenes

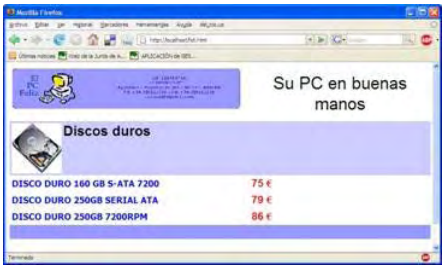
¿Lo has conseguido? No es difícil, por si has tenido algún problema te facilitamos un enlace con el código HTML para generar la página anterior.

Descarga el html

¿Te has dado cuenta de lo tedioso que resulta manejar las etiquetas HTML directamente sin ninguna ayuda, tan sólo con un editor de texto plano? Sería mucho más fácil si dispusiésemos de algún **editor especializado en manejar las etiquetas**; o mucho mejor, poder concentrarnos en el diseño, y que el código HTML se generase automáticamente.

Como ya estarás imaginando, ese tipo de editores existen y además con una oferta muy amplia, incluso los procesadores de texto actuales, como OpenOffice, suelen incluir la posibilidad de generar documentos en ese formato. **Trataremos esas herramientas a continuación**, mientras tanto te proponemos un nuevo ejercicio.

Modifica el fichero HTML anterior, de manera que cada imagen sea un enlace a una nueva página donde aparezca la lista de componentes de cada categoría, pero conservando la cabecera de página.



Aquí tienes la solución. Era fácil ¿verdad?

Descarga la solución

Autoevaluación

¿Cómo se llama la variable de configuración de Apache que indica el directorio donde residen las páginas que pueden servirse?

a) Documentroot.
b) httpd.conf
c) www
d) URL

Comprobar

Herramientas para edición web

El apartado anterior terminó tratando la posibilidad de utilizar herramientas diseñadas específicamente para editar código HTML de una manera más sencilla y potente que escribir las etiquetas directamente en un editor de texto plano, en realidad sería imposible hacer de esta manera cualquier sitio web medianamente sofisticado.

En la actualidad existen multitud de herramientas que vienen en nuestra ayuda **para diseñar y desarrollar webs cada vez más ricas y funcionales**, mucho más atractivas y fáciles de utilizar por el usuario, **produciendo interfaces web cada vez más amigables y espectaculares.**

Vamos a hacer un breve recorrido por **algunos de los editores web más representativos que podemos encontrar en la actualidad.** Como verás, es tal la variedad que existe, que merece la pena que instales algunos de ellos y los pruebes con tranquilidad comparando sus capacidades y posibilidades.

Aunque con grandes diferencias, **en casi todos los productos podemos encontrar las siguientes características principales:**

1. **Edición WYSIWYG**. Consiste en editar de forma visual sobre una representación del documento final que se obtendrá, el editor generará el código HTML necesario para conseguir el resultado que deseamos. Las siglas WYSIWYG significan What You See Is What You Get (Lo Que Ves Es Lo Que Obtienes).
2. **Cliente FTP.** Resulta muy útil para poner las páginas editadas localmente en un servidor web remoto.
3. **Control de versiones.** Permite realizar un seguimiento de las modificaciones y es especialmente interesante en el caso de que un equipo de personas trabajen sobre el mismo proyecto.
4. **Auto completado de código.** Dada la cantidad y variedad de etiquetas HTML resulta una gran ayuda que el propio editor colabore con nosotros en la escritura de las etiquetas.
5. **Resaltado de sintaxis**. Permite tener una visión clara y estructurada del código.
6. **Soporte de lenguajes de servidor.** Como veremos más adelante HTML no permite ningún tipo de programación, por lo tanto hay que integrar otros lenguajes que le aporten esa característica como PHP, ASP o JSP.
7. **Depuración de código.** Esta facilidad resulta de gran ayuda para descubrir y corregir errores.

> ***PARA SABER MÁS***
> ***Una buena definición sobre el WYSIWYG podemos encontrarla en el siguiente diccionario informático, así como enlaces interesantes sobre este tema.***
> ***WYSIWYG***

Los primeros editores web sólo permitían la edición de código HTML, en la actualidad este tipo de productos deben integrar soporte para otras tecnologías como **CSS, JavaScript o PHP** entre otras.

Algunos de los productos más interesantes en este momento son:

- **Dreamweaver.** Es un producto **muy maduro con buenas prestaciones y edición WYSIWYG**, soporta integración con lenguajes de servidor (por ejemplo PHP y JSP) y muy buenas capacidades para manejar CSS. No es open source y por lo tanto hay que pagar por él. Se puede descargar una versión de prueba de su sitio web http://www.adobe.com/es/products/dreamweaver/

- **Amaya.** Es un navegador y editor web **desarrollado por el W3C**. Es un software de fuente libre (open source). Permite navegar por una página y editarla al mismo tiempo. http://www.w3.org/Amaya/

- **Firebug.** Una extensión o **plugin para Firefox** que añade funcionalidad de edición y depuración WYSIWYG mientras navegamos por la página. Naturalmente sólo está disponible para Firefox, por lo que no se puede utilizar para depurar código para Internet Explorer u otros navegadores.http://getfirebug.com/

- **Aptana.** Tiene un muy buen asistente de código con un **gran soporte de CSS y Javascript**. Existe versión open source y permite depurar para Firefox e Internet Explorer. http://www.aptana.com/

El aprendizaje en profundidad de HTML, CSS y Javascript excede las pretensiones de esta unidad, pero para que aprecies la potencia de estas tres tecnologías juntas y te introduzcas en el manejo de un editor web **vamos a desarrollar un pequeño ejemplo que te servirá de pauta para iniciarte en este inmenso mundo del diseño web.**

Vamos a modificar la página de productos de "El PC Feliz" que hemos diseñado anteriormente utilizando el editor web Aptana. Haremos uso de CSS y Javascript, seguro que el resultado te impresiona. Te facilitamos una presentación del proceso que hemos seguido.

Introducción a la edición web utilizando el editor web Aptana

También te proporcionamos los ficheros generados para que puedas colocarlos en tu servidor Apache y probarlos en tu equipo.

Descarga los archivos

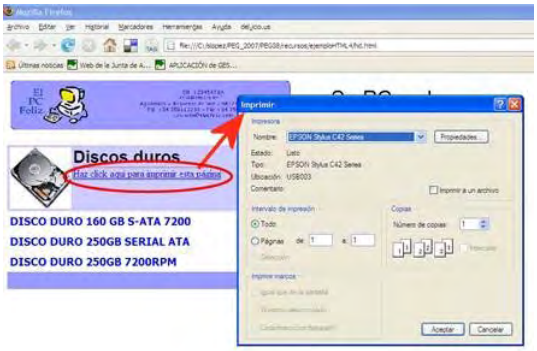
Ahora te proponemos un ejercicio que a buen seguro no tendrás problemas en resolver

Modifica la página web de "El PC Feliz" que te hemos facilitado de manera que cada imagen sea un enlace a una nueva página donde aparezca la lista de componentes de cada categoría, pero conservando la cabecera de

página. En esta ocasión utiliza una hoja de estilo CSS y haz varias pruebas.

Añade el siguiente código Javascript para insertar un enlace que al pulsarlo abra la ventana de impresión.

```
<a href="javascript:window.print()">Haz click aquí para imprimir esta página</a>
```



PARA SABER MÁS

No tenemos espacio ni tiempo aquí para tratar Javascript más profundamente, pero te recomendamos que te esfuerces en intentar aprender más sobre él. A continuación te facilitamos un enlace sobre Javascript, seguro que tú encontrarás muchos más en Internet.

Tutorial sobre Javascript

Autoevaluación

Sobre WYSIWYG...

a) Se utiliza siempre junto a CSS.
b) Permite editar visualmente una página web
c) Es incompatible con Javascript
d) Proporciona un método fácil de corregir los errores al teclear las etiquetas HTML

Comprobar

¿Qué se puede programar con todo esto?

La respuesta es **nada. ¿Te sorprende?**

Si recuerdas en el principio de esta unidad distinguimos entre páginas web estáticas y dinámicas, **hasta ahora sólo hemos escrito páginas estáticas**. Es cierto que les **hemos añadido algunas funcionalidades utilizando Javascript**, pero sólo para mejorar la estética y usabilidad. **Siempre se comportan de la misma manera y producen el mismo resultado.**

Entonces... te preguntarás: ¿cuál es el siguiente paso? ¡Eso es, lo que estás pensando!...

Hay que colocar programas en el lado del servidor web que generen sus salidas en formato HTML, el cual se enviará a los navegadores de los clientes. Es el momento de presentarte a los lenguajes de servidor que nos permitirán dar el salto a las aplicaciones web.

CASO PRÁCTICO.

*Después de instalar su servidor web de pruebas y haberse iniciado en HTML, Javascript y CSS, además de haber instalado varios editores web y haberlos probado, el equipo de **SI Andalucía** está empezando a plantearse el siguiente paso para culminar el proyecto que les encargó **El PC Feliz**.*

*Saben que deben decidirse por algún lenguaje de programación web y un gestor de bases de datos, también deben valorar qué herramientas les proporciona el mercado actual para no empezar desde cero. **Víctor,** como siempre, está deseando lanzarse de lleno, pero **Carmen,** con prudencia, le advierte que es mejor tomarse un tiempo de pruebas y ensayos antes de tomar una decisión.*

***María** aporta la idea de que quizás sea ventajoso utilizar algún framework como MVC o alguno de los muchos sistemas de gestión de contenidos (CMS) de código abierto que existen en el momento actual y utilizarlos como punto de partida para la aplicación de encargo de **El PC Feliz**. Después podrían añadir aquellas funcionalidades que fuesen propias del proyecto que están desarrollando.*

PARA SABER MÁS

En el caso práctico anterior ha aparecido el concepto de framework (marco de trabajo). ¿Quieres saber algo más sobre este concepto cada vez más utilizado?

¿Qué es un framework?

Empezaremos con un pequeño ejemplo que te servirá para entender mejor el papel de los lenguajes de servidor. **En muchas ocasiones habrás utilizado el buscador Google ¿Te has preguntado qué hay detrás de cada búsqueda que haces? ¿Qué pasa en el servidor web de Google cuando lanzas una petición de búsqueda?**

Recuerda esta imagen que ya te presentamos en esta misma unidad didáctica:

Cuando escribimos una lista de palabras clave en la caja de texto de la página de Google y le damos al botón buscar se lanza un proceso en el servidor web de Google cuyo resultado conoces muy bien, lo que no es tan conocido es qué pasa dentro del servidor de Google.

Resulta evidente es que es imposible que Google tenga una base de datos con todas las combinaciones de búsqueda que se puedan utilizar, puesto que éstas son infinitas. Lo que ocurre es que de alguna manera...

- **el servidor de Google recibe nuestra petición,**
- **ésta se procesa consultando sobre la base de datos de direcciones web y**
- **finalmente se compone una página web dinámica** con el resultado de la búsqueda que es enviada a nuestro navegador web.

¿No te recuerda eso al paradigma de la programación? Entrada-Proceso-Salida.

Si observas detenidamente la barra de direcciones de tu navegador lo comprenderás claramente. Fíjate en la siguiente presentación.

Comprobación intuitiva de que hay un programa detrás de una web dinámica

---

Añadiendo PHP a nuestro servidor web Apache

Hemos hablado mucho sobre web dinámicas y aplicaciones web, vamos ahora a poner en práctica lo aprendido. **Vamos a dotar a nuestro servidor Apache de lo que le falta para poder hacer programas para la web.**

Existen muchos lenguajes de programación que se pueden incorporar a un servidor web para generar aplicaciones que utilicen como interface de usuario un navegador web, algunos de los más utilizados son:

1. **ASP.** Active Server Pages, es la tecnología ofrecida por **Microsoft.**
2. **JSP.** Java Server Pages, lenguaje basado en **Java.**
3. **Perl.** Lenguaje interpretado muy usado en **entornos UNIX y Linux.**
4. **PHP.** Independiente de la plataforma y de **código abierto. Muy usado en la actualidad.**

PARA SABER MÁS

Un interesante enlace que te proporcionará más información sobre los lenguajes anteriores y más detalles relacionados con éstos.

Lenguajes del lado servidor o del cliente

Vamos a usar PHP para nuestras primeras prácticas de programación web. **Presenta muchas ventajas para iniciarse en la programación web** puesto que es multiplataforma, **casi todos los servidores web lo soportan**, existe mucha documentación, es de **código abierto** y **se integra fácilmente con sistemas gestores de base de datos** como MySQL o PostgreSQL hasta el punto de que la combinación Apache-PHP-MySQL es la más utilizada por los servidores en la actualidad.

Para instalar PHP en nuestro servidor web podríamos hacerlo descargándolo de la web de php http://www.php.net/. Pero en esta ocasión **utilizaremos una aplicación que nos hará este proceso**

mucho más fácil. La aplicación a la que nos referimos es **AppServ**.

AppServ instala y configura de una manera muy sencilla el servidor web **Apache** con soporte para **PHP** y el gestor de base de datos **MySQL**. Podemos descargar AppServ desde el siguiente enlace:

> ***ZONA DE DESCARGA***
> ***Utiliza el siguiente enlace para descargar AppServ***
> ***Descarga AppServ***

En la siguiente presentación te detallamos el proceso de instalación de AppServ que, como comprobarás, es muy sencillo y automatizado. Aunque no es imprescindible, te recomendamos que desinstales el servidor Apache que habíamos instalado anteriormente para dejar que AppServ pueda configurar la instalación de Apache junto con PHP de forma automática.

Instalación de Apache, PHP y MySQL con el instalador AppServ

Autoevaluación

¿Cuál no es un lenguaje para programación web?

a) Javascript
b) ASP
c) JSP
d) PHP

Comprobar

Generación de aplicaciones para la web (II)

Ahora sí que podemos programar

¿Todo listo para nuestra primera página web dinámica en PHP? Pues no esperemos más...

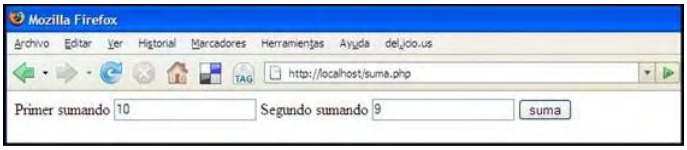
Antes de empezar debemos aclarar que **el estudio en profundidad de PHP cae fuera de los límites de esta unidad didáctica**, esperamos que lo expuesto aquí te sirva **para empezar a profundizar** en el estudio y la práctica de este lenguaje, merece la pena que lo intentes. Para ello **puedes comenzar por cualquiera de la mucha y buena documentación que puedes encontrar en la web**.

Lo primero que debes saber es que **las páginas dinámicas en PHP utilizan la extensión .php**. Para empezar utilizaremos un simple editor de texto plano, aunque ya sabes que un buen editor web como Aptana se hace imprescindible en grandes proyectos.

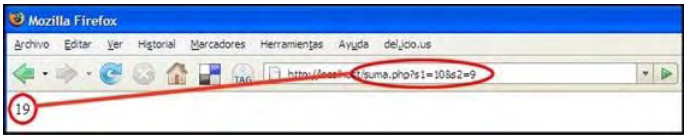
Edita el siguiente código PHP y guárdalo en un fichero llamado suma.php, asegúrate de que lo **colocas en el directorio que tu servidor Apache está utilizando como DocumentRoot**. Si no has alterado la instalación de AppServ, este directorio será **el subdirectorio www** dentro del directorio de instalación de AppServ.

```
<HTML>
<BODY>
<? if (empty($s1) && empty($s2)) { ?>

        <form action=suma.php method=get>
          Primer sumando <input type=text name=s1>
          Segundo sumando <input type=text name=s2>
          <input type=submit value=suma>
        </form>

<? } else {
        print $s1+$s2;
   } ?>
</
</HTML>
```

Ahora **abre un navegador y carga la página** que has escrito tecleando: http://localhost/suma.php



Si escribes dos números en las cajas de texto definidas por el formulario y pulsas el botón de envío...



¡En el navegador aparece una nueva página con el resultado de la suma! (observa los parámetros de la página de resultados y recuerda: **Entrada-Proceso-Salida**).

¡Atención! Si ves el **código fuente de la página** anterior no verás **ni rastro de PHP, sólo HTML.** El programa se ejecuta en el servidor Apache y lo que se envía es HTML puro.

Analicemos el código:

El programa se basa en la conocida **estructura de control `if-then-else` que incorporan todos los lenguajes de programación** (recuerda que HTML no es un lenguaje de programación y carece de ella). En la primera línea establecemos **la condición ¿Están vacías las variables `$s1` y `$s2`?:**

```
<? if (empty($s1) && empty($s2)) { ?>
```

Si están vacías entonces presentamos al usuario el formulario HTML para que pueda **introducir valores:**

```
<form action=suma.php method=get>

Primer sumando <input type=text name=s1>

Segundo sumando <input type=text name=s2>

<input type=submit value=suma>

</form>
```

Y cuando el **usuario pulsa el botón `submit`** el formulario ejecuta su acción y **vuelve a llamar a la página `suma.php`**, pero está vez con los valores introducidos por el usuario como argumentos:

http://localhost/suma.php?s1=10&s2=9

Al volver a cargar la página, **la estructura condicional evalúa la condición de bifurcación a falso** y se ejecuta la parte *else*:

```
<? } else {

        print $s1+$s2;

   } ?>
```

Imprimiendo el valor de la suma de las variables pasadas desde el formulario.

> **Hay que hacer notar que "imprimir" en el mundo de las páginas dinámicas significa generar un resultado visualizable en un navegador web (HTML).**

Peculiaridades de la programación web

¿Qué tal tu primer encuentro con PHP? Ya habrás comprobado alguna de las sigularidades que tiene la **programación web:**

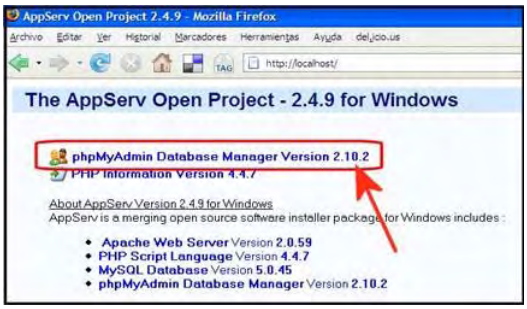
1. **El interface de usuario es un navegador web**, por lo tanto el programa PHP "debe preparar" la **salida en formato HTML.** Esto obliga a combinar como mínimo el lenguaje HTML con el PHP, pero piensa que también pueden entrar en escena otras tecnologías como Javascript o CSS.
2. **Se entremezclan el lenguaje HTML con las líneas de código PHP**, éstas están delimitadas por los símbolos **<?** y **?>**. Esto es un **problema en proyectos grandes** porque hace que los **programas sean difíciles de mantener**. La solución consiste en utilizar técnicas como el patrón MVC (Model-View-Controller) y la programación por capas que ya conoces de unidades anteriores.

¿Y el acceso a bases de datos?

Hemos repetido muchas veces que para obtener **páginas verdaderamente dinámicas debemos conectar nuestros programas de servidor web con un gestor de bases de datos** que proporcione los contenidos que se presentarán en el navegador. Recordarás que cuando instalaste AppServ se instaló MySQL, vamos ahora a presentarte un ejemplo de página PHP que obtiene su contenido de una consulta SQL a una base de datos.

Nos basaremos en el ejemplo anterior de la web de El PC Feliz, **haremos una página dinámica que muestre los artículos disponibles con sus precios. Lo primero** que debemos hacer es **crear una base de datos con una tabla que contenga los datos de los artículo**s, para ello disponemos de otra herramienta que también se ha instalado con AppServ, se trata de **phpMyAdmin.**

PhpMyAdmin es una aplicación web que proporciona un interface web para gestionar MySQL. Para lanzar esta aplicación disponemos de un enlace en la página por defecto del sistema instalado por AppServ.



Vamos con nuestro ejemplo, **crearemos una base de datos muy sencilla con una sola tabla** que contendrá los **datos de los artículos** que queremos presentar. Por simplificar, la tabla constará tan sólo de **tres columnas:**

1. **Categoría.** Expresará la categoría a la que pertenece el artículo: HD, Placa Base, DVD, etc.
2. **Descripción.** Una breve indicación del artículo.
3. **Precio.** El precio de venta.

Vamos a explicar **cómo utilizar phpMyAdmin** con una presentación, te darás cuenta que su manejo es muy intuitivo. **Es importante que sepas que phpMyAdmin es una aplicación web desarrollada en PHP,** eso te dará una idea de las posibilidades de este lenguaje.

Uso de phpMyAdmin para gestionar una base de datos

Un ejemplo de PHP accediendo a BD

Ahora probaremos a escribir una **página PHP muy simple que acceda a la base de datos** y muestre en el navegador la lista de productos de una determinada categoría.

Escribe el código siguiente con un **editor de texto plano** y guárdalo como **`ListaxCategoria.php`** en el directorio raíz de Apache (**`DocumentRoot`**).

```
<HTML>
<BODY>
<H1>Categoría: <?= $categoria ?></H1>
<?
        $dblink=mysql_connect('localhost','root','123456');
        $resultado=mysql_db_query("pcfeliz","SELECT descripcion,precio FROM articulos WHERE categoria='$categoria'");
        $numero_articulos=mysql_num_rows($resultado);
?>
         <TABLE BORDER=1>
           <TR>
             <TD><H2>DESCRIPCIÓN</H2></TD>
             <TD><H2>PRECIO</H2></TD>
           </TR>
<?
        for($i=0;$i<$numero_articulos;$i++)
```

```
                {
                $articulo=mysql_fetch_array($resultado);
              echo "<TR>";
                  echo "<TD>$articulo[0]</TD>";
             echo "<TD>$articulo[1]</TD>";
              echo "</TR>";
                }
    ?>
    </TABLE>
    </
    </HTML>
```

Abrimos un navegador, **tecleamos la URL** http://localhost/ListaxCategoria.php, y...

¿Qué ha pasado? Algo no va bien, **aparece una página sin datos. ¿Sabes el motivo?** Claro, **no hemos pasado el parámetro de la categoría.**

Veamos, si tecleamos http://localhost/ListaxCategoria?categoria=HD , ahora sí.

Comentando el ejemplo...

Vamos ahora comentar **lo más significativo del programa PHP:**

Si se quiere **sustituir una variable por su valor** se emplea **<?=**

```
 <?= $categoria ?>
```

Para abrir una **conexión con el gestor de base de datos MySQL** que está en "localhost" e **identificarnos** como "root" con contraseña "123456" se emplea la **función `mysql_connect()`**, ésta **devuelve un descriptor de conexión** que será usado más tarde.

```
$dblink=mysql_connect('localhost','root','123456');
```

Para lanzar una sentencia SQL sobre la conexión abierta se utiliza la función **`mysql_db_query()`, ésta devuelve el resultado de la consulta.**

```
$resultado=mysql_db_query("pcfeliz","SELECT descripcion,precio FROM articulos WHERE categoria='$categoria'");
```

`mysql_num_rows($resultado)`, devuelve el número de filas contenido en $resultado.

PHP cuenta con una estructura de control de tipo bucle repetitivo que nos es útil para recorrer el $resultado y extraer los valores de las columnas de cada fila.

```
for($i=0;$i<$numero_articulos;$i++)
                {
                $articulo=mysql_fetch_array($resultado);
```

```
echo "<TR>";
    echo "<TD>$articulo[0]</TD>";
  echo "<TD>$articulo[1]</TD>";
echo "</TR>";
  }
```

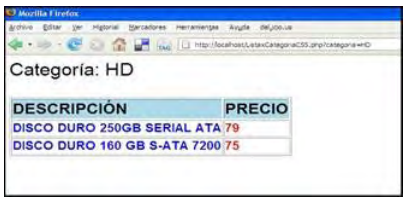
Como ves, la mezcla de un lenguaje de servidor como PHP con HTML genera un código que puede complicarse mucho si no se separa adecuadamente. El uso de patrones como MVC se hace entonces imprescindible.

Una de las técnicas para poder separar la presentación del contenido es usar hojas de estilo en cascada. Vamos a ver con un ejemplo cómo podríamos mejorar la presentación de la página anterior utilizando CSS.

Si incluimos la siguiente hoja de estilo:

```
<style type=text/css>
.titulo1 {color:black; font-size:30px; font-family:arial;}
.titulo2 {color:black; font-size:25px; font-family:arial; font-weight:bold;}
.descripcion {color:blue; font-size:20px; font-family:arial; font-weight:bold;}
.precio {color:red; font-size:20px; font-family:arial; font-weight:bold;}
.fila_cabecera {background-color:LightBlue;}
</style>
```

El resultado podría ser el siguiente:



Y aquí tienes el código completo del ejemplo:

Descarga el código completo

Ahora te toca practicar a ti, te proponemos el siguiente ejercicio que seguro resolverás sin problema:

Utilizando como base las páginas estáticas de "El PC Feliz" que desarrollaste anteriormente en esta misma unidad, reescríbelas de forma que su contenido se obtenga de la base de datos. ¿Te atreves a añadir una página que solicite un precio al usuario y facilite los artículos que tienen un valor igual o inferior a éste?

Autoevaluación

Señala la afirmación falsa

a) phpMyAdmin es un sistema gestor de bases de datos
b) AppServ es un instalador de PHP
c) AppServ es un instalador de MySQL
d) AppServ es un instalador de Apache

Comprobar

Generación de aplicaciones para la web (II)

Una aplicación web completa. El gestor de contenidos Joomla

Ya has podido comprobar las posibilidades de PHP en conjunción con el servidor Apache, y tal como te anunciamos su estudio en profundidad no puede abarcarse desde esta unidad didáctica. Nuestro propósito era mostrarte el camino

que debes seguir para avanzar en el campo de la programación web con PHP.

Existen innumerables sitios web que utilizan PHP como lenguaje de servidor, el propio Moodle que estás utilizando ahora como plataforma de elearning está escrito en PHP. Para terminar vamos a instalar un gestor de contenidos (CMS, Content Management System) escrito en PHP y que es muy utilizado en la actualidad, nos referimos a Joomla.

> *ZONA DE DESCARGA*
> *Joomla es un gestor de contenidos de código abierto que puede descargarse libremente desde Internet.*
> *Sitio web oficial de Joomla en español*

Como siempre, lo mejor es mostrarte los pasos de instalación con una presentación.

Instalación del gestor de contenidos Joomla

Tienes ante ti una herramienta muy potente y versátil desarrollada íntegramente en PHP, deberías ahora tomarte un tiempo para probar los diferentes ajustes de configuración de este gestor de contenidos. Recuerda que muchos sitios web que visitas a menudo están desarrollados utilizando Joomla.

Generación de aplicaciones para la web (II)

> *CASO PRÁCTICO.*
> *Después de evaluar las posibilidades del servidor web Apache en conjunción con PHP el equipo de SÍ Andalucía decide hacer una prueba más. José ha asistido a un congreso sobre tecnologías web organizado por Sun Microsystems y ha descubierto que también se puede utilizar Java para programar aplicaciones para la web.*

Esto revoluciona las expectativas de nuestros amigos, ya que todos tienen una buena formación en Java y piensan que pueden utilizar esa formación adquirida para aplicarla en este proyecto.
José les cuenta a todos los demás que para poder utilizar Java en su desarrollo deberán cambiar de servidor web y empezar a utilizar un servidor de aplicaciones Java. Afortunadamente descubren que existen varias alternativas posibles y algunas de ellas de código abierto.
Se ponen manos a la obra con el nuevo enfoque pensando que deberán empezar de cero, pero pronto se dan cuenta que el camino que llevan recorrido y los conocimientos adquiridos son muy aprovechables y les harán avanzar más rápidamente.

Ya has podido ver en la práctica cómo podemos utilizar un servidor web para servir páginas web estáticas y también como añadir funcionalidad a las páginas web apoyándote en PHP para dotar de capacidad de programación a nuestro servidor web Apache.

También has podido comprobar cómo se complica la separación de la lógica de presentación y la lógica de negocio al mezclar el lenguaje de marcas HTML con el código de programación.

La solución nos viene de la mano de la arquitectura J2EE y los servidores de aplicaciones, pero antes de eso es necesario que te familiarices con otros conceptos y técnicas.

Generación de aplicaciones para la web (II)

Java, servlets y JSP

Hasta ahora has utilizado Java como un lenguaje de programación para desarrollar aplicaciones Stand Alone, ahora aprenderás a utilizar Java para generar páginas web dinámicas.

Lo primero que necesitamos es conocer los conceptos de servlet y JSP:

1. Servlet. Programa Java que se ejecuta en el contexto de un servidor web. Esto es, capaz de procesar datos y generar la respuesta utilizando el protocolo HTTP.
2. JSP. JavaServer Pages, es una tecnología propiedad de Sun Microsystems que permite generar páginas web de forma dinámica utilizando el lenguaje de programación Java. Es una extensión de la API de Java Servlet.

Como siempre, algunos ejemplos nos ayudarán a entender mejor los conceptos presentados. Pero para poder empezar a practicar con ejemplos de Servlets y JSP necesitamos un servidor web que soporte dichas tecnologías.

Un servidor que soporte la ejecución de servlets y páginas JSP se denomina contenedor de servlets (Servlets Container), un servidor de aplicaciones J2EE es un paso más en las capacidades de un simple contenedor de servlets aportando además la posibilidad de los Enterprise JavaBeans (EJBs), de manera que se pueden implementar las diferentes capas de la aplicación, como la lógica de presentación, la lógica de negocio y el acceso a bases de datos de una forma más sencilla, estructurada y escalable.

En la actualidad disponemos de muchas opciones de aplicaciones comerciales y open source tanto como contenedores de servlets y JSP como de servidores de aplicaciones J2EE completos (EJBs). A continuación te presentamos una tabla con los más utilizados:

Servidor	Tipo	Empresa desarrolladora	Distribución del Software	Sitio web
Apache Tomcat	Contenedor de Servlet y JSP	Proyecto Apache	Open Source	http://tomcat.apache.org/
Apache Geronimo	Servidor J2EE	Proyecto Apache	Open Source	http://geronimo.apache.org/
WebSphere	Servidor J2EE	IBM	Propietario	http://www-306.ibm.com/software/webservers/appserv/express/
Oracle Application Server	Servidor J2EE	Oracle Corporation	Propietario	http://www.oracle.com/technology/products/ias/index.html
WebLogic	Servidor J2EE	BEA Systems	Propietario	www.bea.com/
JBoss	Servidor J2EE	Red Hat	Open Source	http://labs.jboss.com/
JOnAS	Servidor J2EE	Consorcio OW2	Open Source	http://jonas.objectweb.org/

Como ves el mercado actual nos ofrece muchas posibilidades, lo que da una idea de la importancia de esta tecnología hoy en día.

Ya habrás comprobado en la tabla anterior que Apache Tomcat no implementa J2EE al completo, pero por otra parte es muy utilizado por su sencillez, de manera que nos servirá para empezar a practicar nuestros primeros ejemplos. A continuación aprenderás a instalarlo, configurarlo y empezarás a usarlo programando tus primeras páginas JSP.

Autoevaluación

Un servlet es...

a) Es una página JSP
b) Un servidor de aplicaciones web.
c) Un programa Java
d) Un componente del J2EE

Comprobar

Generación de aplicaciones para la web (II)

Instalación y configuración de Apache Tomcat

Lo primero que tenemos que hacer es descargar de la página de Apache el servidor Tomcat, te facilitamos a continuación los enlaces. Existen versiones tanto para Linux como para Windows.

ZONA DE DESCARGA

Lo primero que debes hacer es visitar la página del proyecto Tomcat, te recomendamos su lectura atenta, ya que podrás obtener mucha información actualizada en ella.

Página oficial de Apache Tomcat

A continuación tienes los enlaces a las versiones para Linux y Windows

Apache Tomcat para Windows

Apache Tomcat para Linux

Es necesario tener instalado un JRE (Entorno de Ejecución de Java) en el ordenador

donde se instale Tomcat, seguro que en tu sistema hay ya alguno instalado de las anteriores unidades didácticas.

¿Ya has descargado Tomcat? Pues vamos a instalarlo. Nosotros lo instalaremos en Windows, pero tú puedes hacerlo en Linux si lo prefieres. A continuación tienes una presentación que te guiará en la instalación. ¡Vamos...! seguro que estás impaciente por verlo en funcionamiento.

Instalación del servidor JSP Tomcat

¿Qué tal ha ido? ¿Todo bien? ¿Has comprobado que puedes ver en tu navegador la página de prueba de Tomcat? Seguro que sí.

Ya habrás comprobado que el servidor Tomcat utiliza el puerto 8080, igual que hace la página web de gestión de Oracle Express. Esto puede provocar algún conflicto en tu sistema, de manera que alguno de los dos no funcione. Puedes arreglarlo cambiando el puerto de escucha de Tomcat, mientras tanto si en tu caso se da el conflicto de puertos mencionado puedes parar momentáneamente el servidor de Oracle para comprobar que tienes un servidor Tomcat instalado y funcionando en tu ordenador.

Observa que si tienes tu ordenador conectado en red con otros equipos, éste puede servir páginas a los demás. En los otros ordenadores de tu red deberías teclear la dirección IP del ordenador servidor seguido del puerto 8080, algo así como: http://192.168.1.1:8080. El realidad ya sabes que no tiene mucho sentido un ordenador servidor que sirve páginas a sí mismo salvo para hacer pruebas, en el "mundo real" nuestro servidor serviría sus páginas a través de Internet.

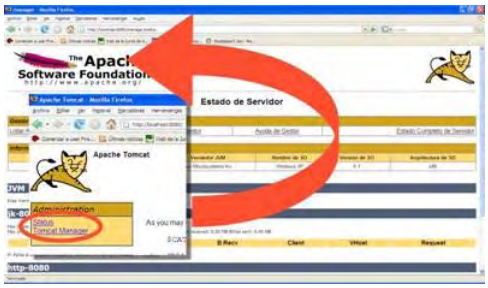
Autoevaluación

¿Por qué no podemos acceder a una página alojada en un servidor remoto tecleando http://localhost:8080?

a) Porque el puerto 8080 sólo sirve para trabajar de forma local
b) Porque HTTP es un protocolo que no está diseñado para acceder remotamente
c) Porque localhost se refiere al equipo local donde estamos situados
d) Es posible acceder de esa forma a un servidor remoto

Comprobar

Podemos acceder a la información sobre configuración y estado del servidor haciendo click sobre el enlace <Tomcat Manager> de la página principal de Tomcat, merece la pena que te observes con detalle los datos que suministra esa página.



Primeros ejemplos de servlets

Antes de empezar a programar nuestros primeros servlets nos familiarizaremos con ellos a través de los ejemplos que vienen con la instalación de Apache Tomcat. Recuerda que en la instalación pedimos instalar los ejemplos. Son muchos y variados, vamos a verlos.

Lo primero es localizarlos y aprender a verlos en acción. En la página principal de Tomcat encontramos un enlace que nos lleva directamente a ellos. Encontraremos tanto ejemplos de servlets como de páginas JSP. No te preocupes ahora si no los entiendes bien, concéntrate por el momento en comprobar cómo funcionan y el aspecto que tiene el código fuente, tanto de los servlets como de las páginas JSP.

¿Ya has visto el aspecto que tienen los servlets y las páginas JSP? Pronto empezarás a escribir tus propios ejemplos, pero antes vamos a conocer mejor qué son y cómo son manejados por Tomcat.

Un servlet es un programa ejecutado en un servidor de aplicaciones o contenedor de servlets (como Tomcat). Es un mecanismo para ejecutar aplicaciones en el lado del servidor. Reciben peticiones y mandan resultados utilizando para ello el protocolo

HTTP, siendo el formato más común de salida una página HTML (aunque puede ser cualquier otro tipo, como un archivo XML, un tipo MIME, una imagen, etc.)

¿Cómo ejecutar servlets?

El proceso de ejecución de un servlet es el siguiente:

1. Una petición HTTP llega al servidor web (por ejemplo Apache Tomcat) desde un cliente (por ejemplo, un navegador web como Firefox).
2. La petición se traslada al motor del servicio de Servlet/JSP del contenedor de aplicaciones.
3. El motor encapsula la petición en un objeto del tipo HttpServletRequest, además encapsula en un objeto HttpServletResponse el flujo de respuesta.
4. El motor crea por cada petición un hilo diferente sobre el que se invoca a la función service() del servlet. En función del método de la petición web (POST o GET), service() llamará al método correspondiente del servlet: doPost(), doGet(), pasándoles los objetos de HttpServletRequest y HttpServletResponse.
5. Cada clase del tipo servlet, tiene una única instancia, sobre la que corren los diferentes hilos (peticiones).

Cada petición genera un hilo independiente. Pero sólo la primera petición genera una instancia de la clase servlet. Podremos tener N peticiones al servlet, por tanto N hilos, pero únicamente una instancia de la clase (un objeto). Esto mejora mucho el esquema que se utilizaba en servidores web anteriores donde cada petición abría un nuevo proceso con la sobrecarga que eso supone para el servidor.

Vamos a programar un servlet sencillo y a desplegarlo en Tomcat. Escribiremos un servlet que muestre en el navegador cliente la fecha y hora del servidor donde está alojado el servlet y que además cuente las veces que el servlet es invocado. Los servlets son clases java que a su vez derivan de la clase javax.servlet.http.HttpServlet.

Generación de aplicaciones para la web (II)

¿Cómo queda nuestro servlet?

Nuestro servlet de ejemplo sería el siguiente:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.Date;

public class FechaHoraLlamadas extends HttpServlet {

  int llamadas=0;

  public void init(ServletConfig conf)
    throws ServletException {
    super.init(conf);
  }

  public void service(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException  {

    Date hoy = new Date();

    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
```

```
        out.println("<html>");

        out.println("<body>");

        out.println("<H3>La fecha y hora del servidor donde estoy alojado es:"+hoy);

        out.println("<BR>");

        out.println("Me has llamado:"+llamadas++ +" veces</H3>");

        out.println("</body>
        out.println("</html>");

      }

    }
```

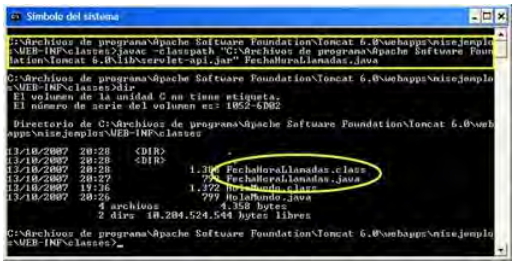
Cuando el servlet es llamado genera el flujo HTTP que es enviado al navegador desde el qué ha sido llamado, el navegador recibe ese flujo HTTP lo interpreta y produce la salida correspondiente (en este caso HTML).

Podemos escribir el código de nuestro servlet de ejemplo utilizando un editor de texto simple como el bloc de notas, después hay que compilarlo con java como ya aprendiste a hacer. Hay que tener en cuenta que hay que indicarle al compilador de java que utilice las clases de servlet que están en el directorio lib de la instalación de Tomcat, en concreto la más importante es servlet-api.jar. La orden para compilar quedaría así:

```
javac -classpath "C:\Archivos de programa\Apache Software Foundation\Tomcat 6.0\lib\servlet-api.jar" FechaHoraLlamad
```

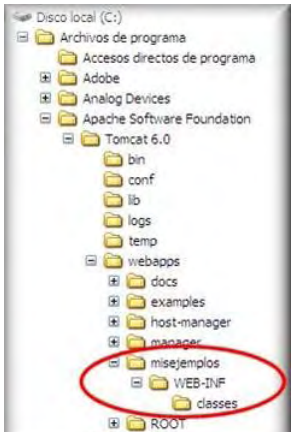
Si todo va bien obtendremos un fichero con la extensión .class, que es nuestro servlet compilado.

Haz clic sobre la imagen para ampliarla:



¿Cómo hacer funcionar nuestro servlet en Tomcat?

Para que funcione debemos desplegarlo en Tomcat, esto quiere decir que tenemos que incluir una serie de ficheros y crear una estructura de directorios en Tomcat para que los clientes puedan acceder a su funcionalidad. Normalmente, el desarrollo de un servlet forma parte de lo que se denomina una aplicación Web, que no es más que una colección de servlets, páginas HTML, JSP, clases y otros recursos, que ofrecen una determinada funcionalidad a la que los clientes acceden típicamente a través de un navegador. Las Aplicaciones Web, a partir de la especificación de Servlet 2.2, deben estructurarse según la siguiente jerarquía de subdirectorios:



1. Directorio raíz: En él se colocan ficheros estáticos (HTML, imágenes, hojas de estilo, etc.) y JSPs.
2. Directorio WEB-INF: Debe contener un fichero *web.xml*. Este fichero configura la aplicación. Por ejemplo, permite declarar servlets, asignarles parámetros de inicio, declarar alias y filtros, etc.

3. Directorio classes: En él se colocan los servlets compilados de las clases utilizadas por la aplicación web.
4. Directorio lib: Aquí se pueden poner otras bibliotecas de clases adicionales (comprimidas como jar).

Pues bien vamos a construir nuestra estructura de directorios dentro del directorio webapps de nuestra instalación Tomcat. Crearemos una nueva estructura para albergar nuestros ejemplos, le llamaremos misejemplos.

En cuanto al fichero web.xml que debe estar dentro de WEB-INF le indica a Tomcat el nombre y el lugar de nuestro servlet. Quedaría así:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    <hyperlink><name>http://java.sun.com/dtd/web-app_2_3.dtd</name>
    <url>http://java.sun.com/dtd/web-app_2_3.dtd</url></hyperlink>>

<web-app>

    <display-name>mis ejemplos</display-name>
    <description>
      Servlets de ejemplos de Servlets
    </description>

    <servlet>
        <servlet-name>FechaHoraLlamadas</servlet-name>
        <servlet-class>FechaHoraLlamadas</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>FechaHoraLlamadas</servlet-name>
        <url-pattern>/servlets/servlet/FechaHoraLlamadas</url-pattern>
    </servlet-mapping>

</web-app>
```

Por fin podemos probar nuestro servlet, escribiendo: http://localhost:8080/misejemplos/servlets/servlet/FechaHoraLlamadas en nuestro navegador web.

No te asustes, todo el proceso anterior queda bastante automatizado si utilizamos un IDE tipo JDeveloper, pero hemos querido que veas "por dentro" todo lo que se necesita para desplegar una aplicación web en un servidor de aplicaciones.

PARA SABER MÁS:

Te facilitamos dos enlaces que te ayudarán a comprender mejor todo lo relacionado con la tecnología de servlets.

Compilación y ejecución de servlets

Interesante enlace con extensa información sobre servlets

Ahora debes practicar todo lo anterior, te proponemos que escribas un

servlet que proporcione la dirección IP del ordenador cliente y lo despliegues en Tomcat.

Primeros ejemplos de páginas JSP

Programar directamente utilizando servlets es algo bastante complicado. La tecnología JSP viene en nuestra ayuda para hacernos las cosas más sencillas. En una página JSP se combinan las etiquetas HTML junto con código Java.

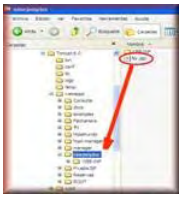
Cuando un usuario solicita una página JSP a través de un navegador web, el contenedor de servlets (Tomcat por ejemplo) reconoce la página por su extensión (.jsp) y lanza el proceso de compilación, fruto de esa compilación se obtiene una clase Java que es... ¿Lo adivinas? Exacto ¡Un servlet!. Como verás a continuación es mucho más sencillo escribir una página JSP que un servlet, puesto que el nivel de abstracción es mayor y nos permite obviar farragosos detalles de implementación. Pero no debemos olvidar que debajo de JSP siempre subyacen los servlets.

Vamos a empezar con un ejemplo sencillo:

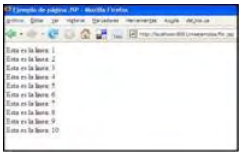
```
<HTML>

<HEAD>

<TITLE>Ejemplo de página JSP</TITLE>

</HEAD>

<BODY>

<%

for (int i=1; i<11; i++) {

  out.println("Esta es la línea: "+ i + "<BR>");

}

%>

</
</HTML>
```

Lo anterior es una página JSP. Lo primero que salta a la vista es la aparición entre las etiquetas HTML de un trozo de código escrito en sintaxis Java (lo hemos resaltado en rojo), concretamente un bucle FOR que realiza 10 iteraciones. Además es de destacar que ese código de programación está encerrado entre las marcas <% (apertura) y %> (cierre), el servidor de aplicaciones contenedor de servlets sabrá que esa parte de la página JSP es la que tendrá que compilar como código Java.

Si escribes la página JSP anterior en algún editor de texto plano y colocas el resultado en el directorio raíz de tu servidor Tomcat (si has seguido el esquema de los anteriores apartados sabrás que nos estamos refiriendo al directorio webapps), podrás ejecutar la página y ver el resultado en tu navegador. No olvides ponerle la extensión .jsp, si no lo haces así el contenedor de servlets no la reconocerá como una página JSP.



¿Lo has hecho? Si todo ha ido bien deberías ver algo parecido a lo siguiente:



Si observas el código HTML de la página, tal como aprendiste a hacer anteriormente, verás lo siguiente:

¿Qué ha pasado? ¿Dónde están las líneas de código Java que escribimos?

La respuesta es sencilla, recuerda que un navegador web no es capaz de interpretar código Java, sólo es capaz de interpretar etiquetas HTML, y eso justamente es lo que le envía el servidor de aplicaciones web a nuestro navegador. El código Java es compilado en una clase servlet y almacenado en el contenedor de servlets. Lo que llega al navegador es el resultado de la ejecución del servlet.

¿Quieres ver dónde está el servlet y la clase correspondiente?

Si buscas en el árbol de directorios de Apache Tomcat los encontrarás en *work/Catalina/localhost* . Observa la siguiente imagen:

¿Has visto el servlet (clase de java) que se ha generado?

Compáralo con la página JSP original ¿Qué te parece? De una simple página JSP se ha generado un complicado servlet. Te lo suministramos como recurso subrayando en rojo las partes más interesantes.

Descarga el código

¿Has comprendido la ventaja de utilizar páginas JSP?

Seguro que sí, ya ves como...

- son mucho más simples que los servlets correspondientes,
- nos permiten aumentar nuestra productividad como programadores,
- concentrarnos en los detalles que nos interesan y
- ocultando la complejidad subyacente a esta tecnología.

Generación de aplicaciones para la web (II)

Acceso a base de datos desde JSP

Ya nos hemos acercado a la tecnología JSP para desarrollo de aplicaciones web. Para terminar esta introducción a las posibilidades de JSP vamos a ver un ejemplo de acceso a base de datos. Nos aprovecharemos de la base de datos MySQL que creamos anteriormente y escribiremos un sencillo programa JSP que recupera los datos de la tabla y los muestra tabulados en el navegador.

Te facilitamos el programa ya escrito. Como sabes, debes colocarlo en el directorio *webapps* para que sea compilado y servido por el servidor Tomcat.

El resultado cuando llames a la página JSP debe ser algo así:

Descarga el código

Hay que hacer notar que para que un programa JSP tenga acceso a un gestor de base de datos debemos instalar el conector JDBC en el directorio lib del servidor Tomcat. El archivo conteniendo el conector JDBC podemos descargarlo del sitio web de MySQL.

Desarrollo web con JDeveloper

En esta unidad hemos pretendido mostrarte el estado actual de la tecnología web. Sin pretender, por falta de espacio, llegar a profundizar en todas y cada una de sus facetas, sino más bien iniciarte en este campo de la informática de indudables posibilidades y proyección profesional.

Por el ejemplo que te mostramos en el apartado anterior habrás comprobado que desarrollar una aplicación web Java utilizando como única herramienta un editor de texto plano no es una tarea viable. Es por esto que se utilizan entornos de desarrollo que nos facilitan la labor.

Ya conoces JDeveloper como entorno para desarrollar aplicaciones de escritorio. Ahora vamos a usar JDeveloper para desarrollar una aplicación web usando JSP y el patrón MVC.

Vamos a comprobar las posibilidades de JDeveloper desarrollando un sencillo ejemplo que consiste en una página JSP que mostrará los datos contenidos en una tabla de una base de datos MySQL. El detalle pormenorizado y comentado de este ejemplo lo puedes seguir en la siguiente presentación:

Ejemplo comentado

PARA SABER MÁS

Pero tú no debes quedarte aquí, es importante que practiques sobre el ejemplo mostrado y lo enriquezcas con nuevas funcionalidades, para ello también te sugerimos los siguientes enlaces:

La comunidad en español de JDeveloper (con vídeos muy interesantes)

Tutorial para realizar una página web maestro-detalle sobre dos tablas

Un interesante tutorial sobre la programación con JDeveloper

No podemos dejarnos atrás la documentación que nos proporciona Oracle, esto es sólo una muestra:

Cómo diseñar el flujo entre diferentes páginas de una aplicación

Cómo diseñar el flujo entre diferentes páginas de una aplicación

Utilización de Enterprise Java Beans