ESCUELA DE INFORMÁTICA

PROGRAMA: TÉCNICO EN DESARROLLO DE SOFTWARE
SUBMÓDULO
PROGRAMACION PARA LA WEB II

DP03 Ver. 02

NOMBRE Y APELLIDOS ESTUDIANTE

_____

CONTENIDO

pág.

PLANEADOR DE CLASES

SEMANA/FECHA PROGRAMADA	CONOCIMIENTOS, COMPRENSIONES Y RECOLECCIÓN DE EVIDENCIAS	ACCIONES A DESARROLLAR	FORMAS DE ENSEÑANZA Y APRENDIZAJE
Semana 1 **Febrero 4-10**	Pacto Pedagógico y diagnóstico de aprendizajes previos	Al finalizar la clase estará en capacidad de: *Identificar y comprender los acuerdos y reglas de juego durante el desarrollo del submódulo *Identificar los aspectos en que se deben fortalecer algunos conceptos previos que se requieren en el semestre	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 2 **Febrero 11-17**	**Scripts avanzados del lado del cliente con DHTML** 1. Identificación de las características avanzadas de los scripts del lado del cliente 1. Uso avanzado de DHTML: Javascript - CSS • Modelo de Objetos de Documento (D.O.M.) • Editores WYSIWYG para DHTML • Tratamiento de formularios	Al finalizar la clase estará en capacidad de: *Identificar las características avanzadas de DHTML *Identificar y utilizar editores que le faciliten el trabajo con DHTML *Aplicar técnicas avanzadas de DHTML en sitios web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

	• Tratamiento de imágenes		
Semana 3 Febrero 18-24	• Control de frames y ventanas. Uso de ventanas para mostrar informes. • Lista de estilos, propiedades y valores aplicables en CSS • Estilos de capas • Maquetación de sitios web con CSS	Al finalizar la clase estará en capacidad de: *Identificar las características avanzadas de DHTML *Aplicar técnicas avanzadas de DHTML en sitios web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 4 Febrero 25-Marzo 3	2. Introducción a las expresiones regulares • Que es una expresión regular • Ventajas y desventajas de las expresiones regulares • Funciones para el tratamiento de expresiones regulares • Comparaciones entre las expresiones regulares y los métodos tradicionales • Ejemplos con	Al finalizar la clase estará en capacidad de: *Identificar patrones de expresiones regulares *Utilizar expresiones regulares para el tratamiento de texto	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

	expresiones regulares		
Semana 5 Marzo 4-10	3. Cookies en Javascript • Que es una cookie • Uso de cookies en sitios web • Funciones comunes para el tratamiento de cookies	Al finalizar la clase estará en capacidad de: *Identificar y tratar las cookies de un sitio web *Utilizar cookies en sitios web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 6 Marzo 11-17	4. Framework para Javascript: jQuery • Conceptos fundamentales sobre los frameworks • Frameworks para Javascript • Que es y para que sirve jQuery • Ventajas de utilizar jQuery • Uso de jQuery • Editores para la creación de archivos en Jquery • Aplicaciones con jQuery	Al finalizar la clase estará en capacidad de: *Identificar un framework *Utilizar el framework jQuery en el desarrollo de aplicaciones web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

Semana 7 Marzo 18-24	• Aplicaciones con jQuery • Recolección de evidencias	Al finalizar la clase estará en capacidad de: *Utilizar el framework jQuery en el desarrollo de aplicaciones web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 8 Abril 1-7	• Recolección de evidencias Hasta esta semana Primera Recolección de Evidencias	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Recolección de pruebas de conocimiento, desempeño y producto
Semana 9 Abril 8-14	**Scripts avanzados del lado del servidor con el lenguaje PHP** 1. Identificación de las características avanzadas de los scripts del lado del servidor • Variables del Servidor web y de sesión • Expresiones regulares en PHP • Manejo de cookies en PHP	Al finalizar la clase estará en capacidad de: *Identificar variables del servidor y de sesión *Utilizar expresiones regulares para el tratamiento de texto *Utilizar cookies en un sitio web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 10 Abril 15-21	Semana Académica (Reunión de Colectivos-Elaboración Mapa de Riesgo)		

Semana 11 Abril 22-28	2. Implementación de clases en PHP • Conceptos sobre POO (Programación Orientada a Objetos) • Creación de clases en PHP • Implementar clases en páginas web 3. Manejo de archivos con PHP 4. Manejo de informes en pdf con PHP	Al finalizar la clase estará en capacidad de: *Identificar una clase y un objeto en PHP *Utilizar clases en un sitio web *Manipular archivos con PHP	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 12 Abril 29 - Mayo 5	5. Algunas aplicaciones • Creación de elementos dinámicos con PHP (listas, listas dependientes, galerías de imágenes, plantillas, y contenido aleatorio entre otros)	Al finalizar la clase estará en capacidad de: *Crear aplicaciones dinámicas involucrando nuevos elementos	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 13 Mayo 6-12	6. Frameworks para PHP • Escoger un framework para PHP • Instalación del framework • Explorando el interior del framework • Modelo vista - controlador • Introducción a la creación de aplicaciones con un framework • Uso de un framework para la creación de una aplicación web	Al finalizar la clase estará en capacidad de: *Identificar los frameworks existentes para PHP *Descargar e instalar un framework para PHP *Utilizar un framework para el desarrollo de una aplicación	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

Semana 14 **Mayo 13-19**	Hasta esta semana Segunda Recolección de Evidencias	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Recolección de pruebas de conocimiento, desempeño y producto
Semana 15 **Mayo 20-26**	Gestión avanzada de información con el motor de base de datos MySQL. 1. Utilización de características avanzadas del motor de Bases de datos MySQL para la gestión de bases de datos a) Diseño e implementación de bases de datos con MySQL • Gestores de bases de datos para MySQL: PHPMyAdmin, MySQL-Front, Consola de MySQL • Normalización de bases de datos (repaso) • Tipos de entidades • Índices • Claves principales y foráneas • Propiedades de los campos • Manejo de integridad referencial b) Características avanzadas de MySQL • Importar y exportar	Al finalizar la clase estará en capacidad de: *Identificar características avanzadas de administración de bases de datos	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

	bases de datos • Uso de los programas (funciones) MySQL • Establecer contraseña de seguridad en MySQL • Establecer privilegios en MySQL • Introducción a la administración de bases de datos con MySQL		
Semana 16 Mayo 27 - Junio 2	• Procedimientos almacenados en MySQL • Desencadenadores en MySQL	Al finalizar la clase estará en capacidad de: *Crear procedimientos almacenados para utilizarlos en las aplicaciones web *Crear desencadenadores en MySQL analizando el costo-beneficio en cuanto al rendimiento	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

		e implementación	
Semana 17 Junio 3-9	Utilización de Ajax y XML en sitios web 1. Identificación de las características de AJAX a) Introducción a AJAX • Que es AJAX • ¿Quién utiliza AJAX? • Fundamentos básicos de Ajax • Patrones de Ajax b) Introducción a XML • Que es XML • Soporte XML en los navegadores • Uso de XML en sitios web • Uso de XML para configuraciones de sistemas • Aplicación de Ajax y XML en sitios web	Al finalizar la clase estará en capacidad de: *Identificar las tecnologías Ajax y XML y su relevancia en los sitio web actuales *Crear aplicaciones que utilicen Ajax y XML	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
Semana 18 Junio 10-16	• Aplicación de Ajax y XML en sitios web	Al finalizar la clase estará en capacidad de: *Crear aplicaciones que utilicen Ajax y XML	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

Semana 19 Junio 17-23	Recolección de evidencias Hasta esta semana Tercera Recolección de Evidencias	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Recolección de pruebas de conocimiento, desempeño y producto
Semana 20 Junio 24-30	Planes de Mejoramientos	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Realización de pruebas de conocimiento, desempeño y producto

PACTO PEDAGÓGICO

CLASE No. 1 PACTO PEDAGÓGICO

Cada docente que recibe un grupo cuenta con la posibilidad, y lo que puede entenderse a su vez como una herramienta pedagógica, de definir los límites de su clase y de pactar, de entrada, las condiciones que permitirán el logro de los objetivos propuestos, la adquisición de la responsabilidad individual y colectiva frente a éstos, y la regulación de los vínculos e interacciones grupales para una favorable convivencia al interior del grupo.

Al establecimiento explícito de dichas condiciones en el aula de clase se le puede denominar PACTO PEDAGÓGICO, el cual se define como el acuerdo inaugural del curso, pues consiste principalmente en aquel compromiso que de antemano se "PACTA" entre el docente y sus estudiantes, y recíprocamente entre los estudiantes y el docente desde el primer día de clase, lo que determina simbólicamente el devenir futuro del curso.

Es importante que el docente dedique la primera hora de clase de su submódulo, al inicio de cada semestre, para el establecimiento del PACTO PEDAGÓGICO, ya que es el momento oportuno para sentar las reglas de juego. Se espera entonces, que el docente una vez se presente al grupo, de inicio a la determinación de ciertos aspectos, dentro de los cuales deben estar contemplados los siguientes:

1. **Presentación del docente**
2. **Conducta de entrada:** El primer día los estudiantes quieren básicamente tres cosas:

- Relaciones:

Los estudiantes quieren sentirse cómodos con sus compañeros de clase. Sentir que hacen parte del grupo y que pueden desarrollar una red de amigos. Por lo tanto lo invitamos a:

a. Desarrollar una actividad corta donde ellos realicen una significativa presentación de lo que son, lo que hacen y quieren ser, para que conozcan a sus compañeros.
b. Reconocer a los estudiantes a través del aplauso, la felicitación o unas palabras motivadoras, teniendo en cuenta a aquellos que trabajan y estudian, que son padres de familia; o que tienen una característica específica de acuerdo con la dinámica desarrollada en el grupo; hacerlos sentir que son valorados por la decisión de transformar su vida a través de la formación técnica. Algunas frases motivadoras pueden ser: HOY EMPIEZAS A TRANSFORMAR TU VIDA; BIENVENIDO, TE ESTÁBAMOS ESPERANDO.

- Visión de Programa:

Este es el motivador más poderoso de todos. Parte de esa visión es el mercado laboral, y ese día los estudiantes están muy abiertos en mirar lo que ellos conocen, y lo que es y no es realista con respecto al programa que eligieron y al plan de estudios del mismo. Ellos quieren saber qué habilidades y competencias necesitarán tener para lograr una carrera

laboral exitosa. Por tal razón esta es la oportunidad de dar a conocer de manera general el perfil del programa técnico.

- Saber cómo Triunfar:

Los estudiantes quieren saber que se requiere para ser exitosos con su programa académico. Quieren saber qué tan diferente va a ser de su experiencia en los colegios donde estuvieron u otras instituciones de educación superior a las que hayan asistido antes.

Para lograr una alta motivación en los estudiantes, el primer día es el mejor para que el docente llegue a la base de su motivación. El primer día tenemos su total atención, se empieza de cero y hay que aprovecharlo. Dado que estamos hablando de educación para el trabajo y el desarrollo humano, la oportunidad para crear una relación de por vida es altísima ya que los estudiantes verán el aprendizaje aplicado desde el primer día.

3. **Dimensión disciplinaria:** la puntualidad y la asistencia tendrán seguimiento por medio de la "llamada a lista" los primeros cinco minutos de la clase. No se permite ingerir alimentos, comer chicles o realizar ventas durante el desarrollo de la clase. También interferir negativamente con el adecuado desenvolvimiento del submódulo, manifestando conductas o comportamientos que atenten contra el logro de los objetivos del trabajo propuesto.

4. **La asistencia:** aclarar la importancia de la asistencia a clase o en el caso de que se presente la inasistencia del alumno, justificarla con una excusa escrita (médica o laboral) o en caso de ser verbal, demostrar su validez.

5. **La puntualidad**: definir la hora en que se comenzará y finalizará la clase regularmente. Los estudiantes que tengan dificultades para cumplir estos horarios pueden aclararlo desde el principio y comprometerse a mantenerse al día en el desarrollo del submódulo.

6. **La evaluación**: oficializar la metodología de la recolección de evidencias, los momentos empleados para este proceso y las dimensiones que serán evaluadas: conocimiento, desempeño y producto, así como el porcentaje respectivo. De igual forma, anunciar el tiempo de devolución de las evidencias, ser muy claro en los plazos acordados para la entrega de trabajos por parte de los estudiantes y fijar las pautas y derroteros con los cuales deben ser presentados. Aclarar el tipo de sanciones cuando se presenten eventuales fraudes.

7. **El conducto regular:** establecer las instancias regulares a las cuales el estudiante podrá recurrir en el caso de presentarse algún inconveniente o desacuerdo significativo.

8. **Los recursos logísticos:** velar por el uso adecuado de los materiales didácticos, los equipos, los muebles y elementos del salón de clase.

9. Los estudiantes del primer semestre tendrán una inducción especial en la cual se les presentará el perfil, los objetivos, la justificación y las áreas de estudio del programa que van a realizar, igualmente se clarificarán las dudas que surjan al respecto.

10. MOMENTOS DE RECOLECCIÓN DE EVIDENCIAS: Los estudiantes tendrán 3 momentos de recolección de evidencias de conocimiento, desempeño y producto, con sus respectivos mejoramientos:

    - La primera en las primeras 6 semanas
    - La segunda entre la 7 y la semana 12
    - La tercera entre la 13 y la semana 18

11. Los estudiantes pueden habilitar en los siguientes casos:

    - Si pierden una instancia de un submodulo(conocimiento, desempeño o producto)
    - Si pierde dos instancias de un submódulo
    - Si pierde dos instancias en submódulos distintos
    - Si pierde tres instancias en dos submodulos distintos(dos en uno y uno en otro)
    - Si pierde cuatro instancias en dos submódulos distintos(dos en uno y dos en otro)

En los demás casos debe repetir los submódulos en donde haya perdido las instancias de evaluación.

INTRODUCCIÓN

Este documento tiene como finalidad ser un material de apoyo para los estudiantes de programación web y además servir como una referencia de consulta, en donde encontrará, además de los conceptos teóricos acerca de la creación de sitios web, ejemplos resueltos y ejercicios propuestos al final de cada capítulo.

Es la continuación del texto Programación para la Web I, abordando las mismas temáticas desde un enfoque más avanzado.

El texto se divide en varias partes, partiendo del diseño de páginas y sitios web con *DHTML*, en donde se estudia lo relacionado a las interfaces de usuario más importantes a emplear en la creación de sitios. Posteriormente se estudian aspectos avanzados de PHP para la creación de páginas web dinámicas. Se estudia además AJaX, una tecnología que combina Javascript con XML de forma asincrónica con el servidor.

Si el estudiante desea, puede leer el documento en el orden indicado, aunque puede estudiar primero los capítulos de PHP y MySQL y luego el tema relacionado con DHTML (Javascript y CSS), dependiendo del interés.

Otro aspecto importante a tener en cuenta, tiene que ver con la tarea de hacer tanto los ejemplos resueltos como los ejercicios propuestos, por parte del estudiante, pues hacerlos, fortalecen los conceptos teóricos y permiten ganar experiencia al enfrentarse a diferentes problemas, además, allí encontrará explicaciones y notas adicionales de los conceptos definidos.

Cabe anotar además que el enfoque de este texto está orientado a la utilización de herramientas de *software libre* (*free software*), por una parte, por lo apasionante de este mundo, y por otro, por las enormes ventajas que trae el desarrollo con este tipo de tecnologías. También emplearemos herramientas de libre uso y gratuitas (*Freeware o Software gratuito)*, y que no son software libre[1].

[1] No sobra decir que este texto también es libre, esto es, puede ser copiado, modificado y redistribuido según la GNU GFDL (Licencia de Documentación Libre de GNU, una “hermana” de la Licencia Pública General GNU/GPL que protege el software de intentos de apropiación que restrinjan sus libertades a los usuarios). Esto, siempre y cuando se conserve la autoría del mismo (Jaime Montoya) y las referencias bibliográficas que allí aparecen.

HTML DINÁMICO (DHTML – DYNAMIC HTML)

CLASE No. 2 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

DHTML

Figura 1. Algunos navegadores web que se ajustan a los requerimientos del DOM

El HTML Dinámico o *DHTML* (del inglés Dynamic HTML) designa un conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación del lenguaje *HTML* estático, un lenguaje interpretado en el lado del cliente (como *JavaScript*), el lenguaje de hojas de estilo en cascada (*CSS – Cascade Style Sheet*) y la jerarquía de objetos del *DOM (Document Object Model)*.

Una página de un sitio HTML Dinámico es cualquier página web en la que los scripts en el lado del cliente cambian el HTML del documento, después de que éste haya cargado completamente, lo cual afecta la apariencia y las funciones de los objetos de la página. La característica dinámica del DHTML, por tanto, es la forma en que la página interactúa con el usuario cuando la está viendo, siendo la misma página para todos los usuarios que la visitan[2].

[2] A diferencia de una página PHP, cuyo script se encuentra del lado del servidor; aquí, cada usuario ve una página distinta, ya que el servidor devuelve páginas HTML de acuerdo a parámetros suministrados por éstos o por un programa.

En contraste, el término más general "página web dinámica" lo usamos para referirnos a alguna página específica que es generada de manera diferente para cada usuario, en cada recarga de la página o por valores específicos de variables de entorno. Este término no debe ser confundido con DHTML. Estas páginas dinámicas son el resultado bien de la ejecución de un programa en algún tipo de lenguaje de programación en el servidor de la página web (como por ejemplo ASP.NET, PHP o Perl), el cual genera la página antes de enviarla al cliente, o bien de la ejecución en el lado del cliente de un código que crea la página completa antes de que el programa cliente (navegador) la visualice.

En una página DHTML, una vez ésta ha sido cargada completamente por el cliente, se ejecuta un código (como por ejemplo en lenguaje JavaScript) que tiene efectos en los valores del lenguaje de definición de la presentación (por ejemplo CSS), logrando así una modificación en la información presentada o el aspecto visual de la página mientras el usuario la está viendo.

Entre los usos más habituales del DHTML están el hacer menús desplegables, imágenes que cambian al pasar el cursor sobre ellas, objetos en movimiento, botones que permiten desplazar el texto que se está mostrando, textos explicativos que aparecen al situar el cursor sobre ciertas palabras clave, cronómetros, etc.

Otro uso interesante de esta tecnología es la creación de juegos de acción que utilizan el navegador web para funcionar, aunque tradicionalmente este tipo de desarrollos han sido complicados debido a las diferencias en el lenguaje y las características soportadas por los distintos navegadores existentes. Recientemente los navegadores más populares han empezado a soportar estándares comunes, como el DOM, lo cual ha facilitado mucho la creación de este tipo de aplicaciones.

DOCUMENT OBJECT MODEL (DOM)

Document Object Model o *DOM* (Modelo de Objeto de Documento o Modelo en Objetos para la representación de Documentos) es esencialmente una interfaz de programación de aplicaciones (API – Application Programming Interface) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

El responsable del DOM es el *World Wide Web Consortium (W3C)*[3].

[3] El World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce recomendaciones para la World Wide Web. Es dirigido por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), el protocolo HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de HiperTexto) y el Lenguaje de Marcado de HiperTexto HTML (HyperText Markup Language) que son las principales tecnologías sobre las que se basa la Web. Además, Barners-Lee junto con su equipo de trabajo diseñó y construyó el primer navegador (llamado WorldWideWeb y desarrollado con NEXTSTEP) y el primer servidor Web al que llamó httpd (HyperText

Figura 2. Logotipo World Wide Web Consortium (W3C).

En efecto, el DOM es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (ahora conocido como JavaScript).

Desarrollo del DOM

La primera vez que el DOM se utilizó, fue con el navegador Netscape Navigator en su versión 2.0. Este DOM se conoce también como "modelo básico", o "DOM nivel 0". Internet Explorer 3.0 fue el primer navegador de Microsoft que utilizó este nivel. Netscape 3.0 empezó a utilizar rollovers. Microsoft empezó a usar rollovers en Internet Explorer 4.0. Netscape 4.0 agregó la capacidad de detectar eventos ocurridos en el ratón y el teclado. Una característica de este navegador fue el uso de capas.

En Internet Explorer 4.0 todos los elementos de una página web se empezaron a considerar objetos computacionales con la capacidad de ser modificados. Debido a las diferencias en estos navegadores, el World Wide Web Consortium emitió una especificación denominada "DOM nivel 1" en el mes de octubre de 1998 en la cual se consideraron las características y manipulación de todos los elementos existentes en los archivos HTML y XML. En noviembre del año 2000 se emitió la especificación del "DOM nivel 2". En esta especificación se incluyó la manipulación de eventos en el navegador, la capacidad de interacción con CSS y la manipulación de partes del texto en las páginas web. "DOM nivel 3" se emitió en abril de 2004; utiliza la Definición de Tipo de Documento (DTD – Data Definition Type) y la validación de documentos.

Transfer Protocol Daemon). El en texto Guía Académica Programación para la Web I, podrá encontrar mas detalles sobre el W3C, así como en el sitio http://www.w3c.org.

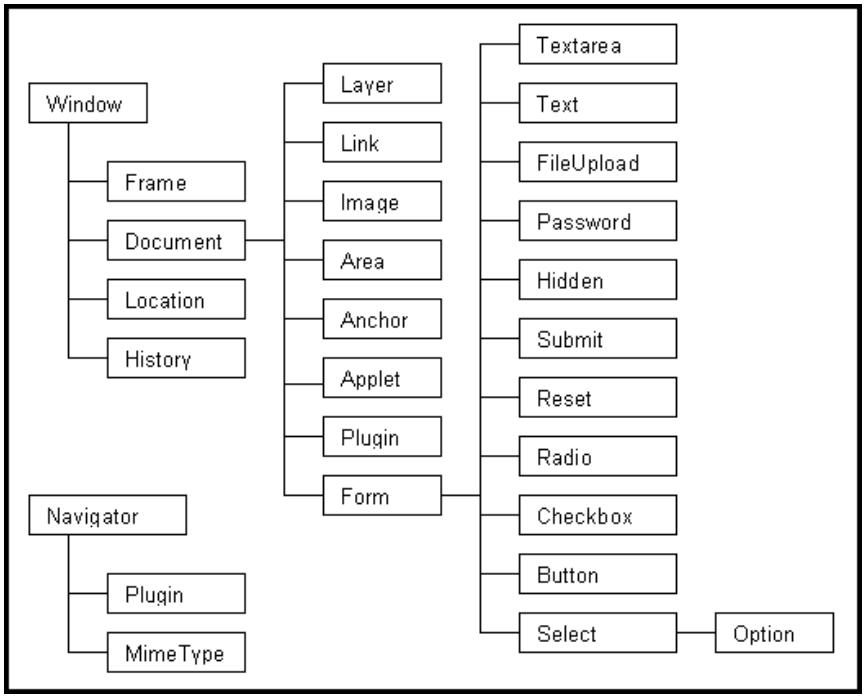

Figura 3. Jerarquía del DOM.

Problemas de compatibilidad

La guerra de navegadores que existió entre Netscape Navigator (hoy, abandonado el proyecto) e Internet Explorer de Microsoft y otras compañías sigue creando graves problemas para los programadores de páginas web, ya que, aunque todos los navegadores utilizan Javascript como uno de los lenguajes de programación, los objetos no se comportan de la misma forma, lo que obliga con frecuencia a programar las páginas en más de una versión, una para Netscape, o Mozilla Firefox, otra para Internet Explorer, otra para Safari, Opera, etc; sumado a esto, no todas las versiones de un mismo navegador se comportan igual.

El World Wide Web Consortium (W3C), el consorcio encargado de definir los estándares de la web, decidió crear un modelo de objetos único, el DOM, para que todos los fabricantes pudieran adoptarlo, facilitando la compatibilidad plena entre ellos.

No obstante, Microsoft ha añadido su propia extensión al DOM, creando problemas de interoperabilidad para los navegadores web.

Como el navegador de Microsoft, Internet Explorer, hasta el año 2002 fue el navegador web mas empleado, esto llevo a un problema real a los desarrolladores de navegadores más comprometidos con los estándares, como Mozilla. Si adoptan las extensiones de Microsoft al DOM, se arriesgan a perder credibilidad en sus llamadas a que los sitios web respeten el estándar, y si no lo hacen, se arriesgan a alinear a sus usuarios por la pérdida de compatibilidad con casi todos los sitios web que utilizan las extensiones de Microsoft. Los críticos han observado esta actitud como otro caso de aplicación de la táctica de Microsoft de “adoptar, extender y extinguir”. Esto puede ser considerado irónico, debido a que tanto Microsoft como Netscape fueron responsables de proporcionar extensiones no estándares en la “carrera armamentística” por el control del estándar, y Mozilla surgió como una iniciativa de Netscape.

La opinión general parece indicar que esto cambiará sólo si nuevos navegadores que respeten los estándares ganan una cuota de mercado significativa en la Web, de forma que el uso de extensiones no estándares se convierta en un problema comercial para los autores de los sitios web que las usen. Esto ya está pasando con Mozilla Firefox, el cual desde hace varios años, viene incrementando su utilización en computadores personales junto a la novedad de Google Chrome, y sin dejar de mencionar a Opera, quien también tiene un buen número de adeptos y a los que les debemos la innovación en el sistema de pestañas en los navegadores.

Estableciendo referencias a objetos

El DOM define la manera en que los objetos y elementos se relacionan entre sí en el navegador y en el documento. Cualquier lenguaje de programación adecuado para el diseño web del lado del cliente puede ser utilizado. En el caso de JavaScript, cada objeto tiene un nombre y un identificador, los cuales son únicos. Cuando existe más de un objeto del mismo tipo en un documento web, estos se organizan en un arreglo (vector).

Es posible asignarle una identificación a un objeto, y luego usarla para hacer referencia a éste, por ejemplo:

```
<div id="Capa">....</div>
```

Para hacer referencia a elementos de una página usamos el método getElementById del objeto document.

```
document.getElementById("Capa")
```

Manipulando las propiedades y funciones de objetos

Los objetos computacionales de la misma forma que cualquier objeto de la vida real, tienen propiedades (atributos). Algunos ejemplos de propiedades de objetos de la vida real son dimensiones, color y peso entre otras.

En la mayoría de los objetos computacionales, algunas propiedades se pueden determinar de la siguiente manera:

Objeto.propiedad = valor;

Por ejemplo para el objeto “motocicleta”
motocicleta.color = “azul”;

La manipulación de objetos sigue los mismos principios que en el lenguaje de programación que se esté utilizando. Una de las características de estos objetos es la función para la cual están diseñados, de hecho la mayoría de las ocasiones tienen más de una función. En JavaScript, muchas funciones (métodos) para cada uno de los objetos, incluyendo el navegador y la ventana que lo contiene, han sido definidas previamente; adicionalmente, el usuario puede definir funciones asociadas a manejadores de evento de acuerdo a sus necesidades.

Eventos
Un evento, desde el punto de vista computacional, ocurre cuando alguna situación cambia en el sistema, como por ejemplo, mover el cursor del mouse (ratón), presionar alguna tecla, la condición de la pantalla, etc. En la creación de páginas web estos eventos representan la interacción de la máquina con el usuario y los programas.

Cuando algunos de estos eventos ocurren, como por ejemplo la presión de algún botón del mouse, es deseable que el computador responda de alguna manera. Esta es la razón por la que existen los *event handlers o* “manejadores de evento”, que son objetos que responden a eventos. Una manera de añadir eventos en el DOM utilizando javascript es:

<***etiqueta*** *onevento="script">....</****etiqueta***>

Ejemplo

```
<div id="midivision" onClick="alert('Mira el mensaje');">
  Aquí puede ir contenido
<div>
```

Otra forma de manipular eventos en JavaScript al crear páginas web, es tratándolos como propiedades de los elementos que forman la página, por ejemplo:

objeto.evento = función;

document.midivision.onclick = acción_a_ejecutar;

O también

document.getElementById("midivision").onclick = acción_a_ejecutar;

En el DOM, se considera que un evento se origina en el exterior de la página web y se propaga de alguna manera hasta los elementos internos de ésta. Un ejemplo de esta propagación es:

EVENTO → Ventana → Document → HTML → BODY → DIV → DESTINO

RESPUESTA → DIV → BODY → HTML → Document → Ventana → EVENTO

Siguiendo esta idea, se establecen tres etapas: captura, la cual se da cuando el evento se está trasladando a su destino. Blanco, que es cuando llega a su destino; este destino es el objeto en el cual se va a crear una reacción a este evento. Finalmente la etapa de burbujeo que es cuando el evento "regresa" a su posición original.

Ciertos objetos pueden estar pendientes de ciertos eventos. Para hacer esto, el objeto añade un "oyente de eventos" con la función addEventListener. Cuando el evento ocurra, alguna función determinada se lleva a cabo; en este proceso se indica en que momento debe producirse, ya sea en la etapa de captura o en la etapa de burbujeo. Este momento se indica con la palabra true si debe ocurrir en la etapa de captura o false si debe ocurrir en la etapa de burbujeo. En JavaScript se escribe de la siguiente manera:

objeto.addEventListener(evento, funcion, momento);

Ejemplo

```
document.getElementById("midivision").addEventListener("click", mifuncion, false);
```

EDITORES DE PÁGINA WEB

Para escribir el código HTML o de otro lenguaje para la web, puede emplearse cualquier editor de texto (pero no un procesador de textos); algunos que pueden utilizarse son:

Editores avanzados WYSIWYG[4]: Ofrecen características avanzadas para el desarrollo de págianas y aplicativos web, algunos son: Aptana Studio (basado en el popular IDE Eclipse), Dreamweaver, Expression Web, PHP Designer, OpenOffice.org.
Editores semi avanzados: Ofrecen algunas funcionalidades que ayudan en la edición del código, entre ellos tenemos el Programmer's Notepad, Notepad++, Textpad, Crimson, Geany.
Editores sencillos: Ofrecen muy pocas opciones de edición de código o ninguna, algunos de éstos son: *gedit* de Linux, *Bloc de notas* de Windows, Edit del D.O.S.

Algunos de estos editores son productos de software libre y otros no, y no todos son gratuitos, por lo que tendrá que pagar por el uso de algunos de ellos. Decidir con cual

[4] ***WYSIWYG*** es el acrónimo de ***W****hat* ***Y****ou* ***S****ee* ***I****s* ***W****hat* ***Y****ou* ***G****et* (en inglés, "Lo Que Ves Es Lo Que Obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final.

editor trabajar es una cuestión de gusto que solo se define al experimentar con varios de éstos. Estos editores también pueden ser utilizados para editar código PHP, Javascript y CSS entre otros. Cabe aclarar que las páginas HTML son archivos con extensión *.html*, la cual debe especificar a la hora de guardar por primera vez.

Figura 4. Logos de algunos editores para página web: Geany, Programmers Notepad, Texpad, Notepad++, Crimson, Dreamweaver y Aptana Studio respectivamente.

El trabajo con scripts avanzados requiere de un buen conocimiento del lenguaje que se esté usando. Para facilitar esta tarea, es muy recomendable trabajar con los editores tipo WYSIWYG o con un IDE, ya que ofrecen herramientas avanzadas que ayudan en el desarrollo de las aplicaciones web.

Para el trabajo con PHP y DHTML, se recomienda utilizar un IDE como Aptana Studio, ya que esta herramienta, aparte de ser un software libre (basado en Eclipse), está orientada a los programadores. Para PHP también se puede utilizar el editor PHPDesigner, que es una herramienta desarrollada para tal fin.

MANEJO DE IMÁGENES CON JAVASCRIPT

Javascript permite acceder a las propiedades de las imágenes y manipular los eventos que soportan, facilitando construir páginas interactivas con el usuario. Desarrollaremos aplicaciones para crear efectos como los rollovers (muy utilizados por los diseñadores para incluirlos en sus páginas y que consisten en establecer imágenes de sustitución), a modificar propiedades de una imagen en tiempo de ejecución. Se revisará además, la forma como JavaScript accede a los objetos y a sus propiedades y métodos.

El tratamiento de imágenes en JavaScript puede ser breve o extenso. Si nos centramos en las propiedades esenciales del objeto *Image* a las que podemos acceder con este lenguaje, el texto puede resultar corto, pero si nos adentramos en explicaciones de los múltiples scripts que se pueden construir basándose en estos elementos, el manual quizá sería algo grande.

Se explorarán las propiedades, métodos y eventos propios; con los ejemplos, el tema se refuerza para su comprensión.

El lenguaje JavaScript posee un objeto propio asociado a cada una de las imágenes de un documento HTML, el objeto *Image*. Además, también posee un “array” (arreglo, matriz o vector) particular, el array *images*, que contiene todas las imágenes presentes en la

página. El orden o posición de cada imagen dentro del array corresponderá al que se haya dado a las imágenes al añadirlas a la página.

Dentro de la jerarquía propia de este lenguaje, tanto el objeto Image como la matriz images se encuentran incluidos dentro del objeto document, que a su vez se encuentra incluido en el objeto principal window.

Por lo tanto, al ser estos objetos propios del lenguaje, podemos referirnos a ellos y acceder a sus propiedades de forma directa, sin tener que recurrir a capas ni otros elementos externos. Así, podemos acceder directamente a una propiedad cualquiera de una imagen del documento de dos formas diferentes:

1) Mediante el objeto *Image*, cuya sintaxis general es:

document.nombre_imagen.nombre_propiedad

Donde *nombre_imagen* es el nombre asignado a la imagen mediante su atributo *name* (es condición indispensable para usar este método el haber asignado a la imagen un nombre único mediante el atributo name). *nombre_propiedad* define la propiedad a la que queremos acceder.

2) Mediante la matriz *images[ ]*. Esta matriz (unidimensional) contiene todas las imágenes del documento, empezando su índice en 0, como en todos los arreglos en JavaScript. La sintaxis general es:

document.images[índice].nombre_propiedad

La equivalencia entre el índice y la imagen que le corresponde se establece de forma secuencial, de tal forma que images[0] representará a la primera imagen insertada en el cuerpo (*body*) de la página, images[1] a la segunda, y así sucesívamente, salvo que se haga una declaración explícita al respecto.

El array images posee la propiedad *length*, que almacena el número de imágenes presentes en el documento. Para obtener éste, basta con escribir:

document.images.length.

Ejemplo

```
…
<script type=”text/javascript”>
function verancho_cambiaralto ()
{
  // Acceso a la propiedad width de la imagen cuyo nombre es img1
  alert(document.images.img1.width);
```

```
  //o bien:
  alert(document.img1.width);
  //También lo podemos hacer así:
  alert(document.images[0].width);//siempre y cuando esta imagen sea la primera en la
página
  // Acceso a la propiedad height de la imagen cuyo nombre es img1
 document.img1.height="500";
}

</script>
…
<img src="imagenes/img1.jpg" name="img1" id=”img1” width="200" height="200"
border="0" onclick=”verancho_cambiaralto();” />

…
```

De esta forma podemos acceder y/o cambiar cada una de las propiedades de una imagen.

El objeto Image posee una serie de propiedades, a las que podemos acceder mediante JavaScript, permitiendo este lenguaje leerlas e incluso cambiar muchas de ellas. La forma de acceder a éstas es:

document.nombre_imagen.propiedad

Las principales propiedades del objeto Image son:

name: referencia el nombre único de la imagen asociada al objeto. La forma de acceder a esta propiedad es:

document.images[indice].name

Por ejemplo, para ver el nombre (name) de todas las imágenes de nuestra página podemos escribir un código como éste:

Ejemplo

```
…
<script language="JavaScript" type="text/javascript">
function ver_nombres()
 {
  var nombres = "";
    for (i=0;i<document.images.length;i++)
     {
```

```
      nombres += document.images[i].name + "\n";
    }
  alert(nombres);
 }
</script>

…

<form>
<input type="button" value="Obtener los nombres de las imágenes"
onClick="ver_nombres();">
</form>
…
```

Como resultado de esta acción, sólo aparecerán aquellas imágenes a las que hemos nombrado por medio del atributo name.

src: este atributo de la etiqueta image, establece la ruta de la imagen asociada al objeto. De esta forma se puede obtener o asignar la ruta de una imagen a una variable. Utilizando esta propiedad, podemos construir rollovers, en los que cambiamos dinámicamente su valor. El atributo src trata la ruta como una cadena de texto (String).

width y **height**: estas propiedades, que representan el ancho y el alto de la imagen respectivamente, se puden leer y modificar de forma dinámica.

border: esta propiedad está asociada al atributo respectivo que establece el borde de la imagen. Puede accederse a esta propiedad utilizando cualquiera de estas formas:

document.images[índice].border
document.nombre_imagen.border

hspace y **vspace**: estas propiedades definen el espacio horizontal y vertical de una imagen flotante respectivamente, respecto al texto que la rodea. El acceso a esta propiedad se logra mediante una sentencia como esta:

document.nombre_imagen.hspace

lowsrc: esta propiedad, que fija la ruta de una imagen accesoria, generalmente una versión de menos peso en KB de la imagen principal, que se debe cargar en pantalla mientras se recibe ésta. Su sintaxis es:

document.nombre_imagen.lowsrc

prototype: propiedad muy importante desde el punto de vista del programador web, ya que permite añadir propiedades y métodos adicionales a un objeto Image, es decir,

permite ampliar lo que es el objeto en sí, aumentando sus propiedades por defecto. Para su uso es necesario un conocimiento profundo del lenguaje.

complete: propiedad del objeto Image de tipo booleano; cuando su valor es verdadero (true) indica que la imagen se ha acabado de cargar en memoria, y cuado es falso, indica lo contrario.

Ejemplo

```
…
<script language="Javascript" type="text/javascript">
var ruta = document.imagen1.src;
 </script>
…

<!--En el código HTML es:-->

<img name="imagen1" src="images/img1.jpg" width="200" height="200" border="1">

<form>
<input type="button" value="Ver ruta de la imagen" onClick="alert(ruta);">
<input type="button" value= "Ver el alto de la imagen"
onClick="alert(document.imagen1.height)">
<input type="button" value="Cambia borde de imagen"
onClick="document.imagen1.border=10;">
</form>
…
```

Las propiedades border, name y vspace no son accesibles para objetos creados mediante el constructor Image().

Veamos algunos eventos soportados por una imagen.

onAbort: este evento se activa cuando se aborta la carga de una imagen en pantalla, por ejemplo porque el usuario presiona el botón "Detener" del navegador.

onError: se dispara cuando por algún motivo externo no se ha podido realizar la carga de la imagen en pantalla, por ejemplo porque la ruta de la misma esté mal especificada.

onLoad: se desencadena cuando se ha acabado de cargar la imagen y se presenta en pantalla.

onClick: se activa cuando se presiona clic con el puntero del mouse en la imagen.

onmouseOver: se desencadena cuando se el puntero del mouse pasa sobre la imagen.

onmouseOut: se dispara cuando el puntero del mouse, luego de estar sobre la imagen, pasa a otro punto de la página.

Ejemplo

```
…
<img name="img1" src="imagenes/dibujo.jpg" onAbort="alert ('Se canceló la carga de la imagen');" />

<img name="img1"src="imagenes/dibujo.jpg" onError="alert('No se pudo cargar la imagen');" />

<img name="img1" src="imagenes/dibujo.jpg" onLoad="alert('Se cargó la imagen correctamente');" />

<img name="img1" src="imagenes/dibujo.jpg" onClick="alert('Has presionado clic sobre la imagen');" />

<img name="img1" src="imagenes/dibujo.jpg" onmouseOver="alert('Has pasado el mouse sobre la imagen');" />

<img name="img1" src="imagenes/dibujo.jpg" onmouseOut="alert('Has quitado el mouse de la imagen');" />

…
```

El objeto Image posee en JavaScript un método constructor, de tal forma que podemos declarar con él un objeto de este tipo al inicio de la página.

La sintaxis es la siguiente:

nombre_imagen = new Image (width,height);

Donde los parámetros width y height corresponden a los atributos análogos de la imagen definida. Si no especificamos estos parámetros, la imagen aparecerá con el tamaño por defecto. Sin embargo, como no se ha establecido una imagen, es preferible utilizar el constructor de esta forma:

nombre_imagen = new Image ( );

nombre_imagen.src = "ruta_imagen";

Y con esto el navegador ya sabe que hemos declarado un nuevo objeto imagen y sabe también qué imagen en concreto es la asociada al objeto, pudiendo cargarla en memoria y tenerla así disponible para poder usarla cuando queramos. Ahora se pueden definir objetos tipo imanes (imágenes específicas) con este constructor declarando su tamaño mediante los parámetros width y height.

Ejemplo:

```
dibujo = new Image(249,28);
dibujo.src="imagenes/dibujo.jpg";
```

La declaración de los objetos Image que figurarán en nuestra página es muy importante, sobre todo en el caso de que en ella vayamos a cambiar dinámicamente la ruta de alguna de las imágenes inicialmente presentes en el cuerpo de la página. Este es el caso de los famosos rollovers en los que, si no hemos declarado en la cabecera el objeto Image correspondiente a la nueva imagen a pintar en pantalla con su ruta correcta, el navegador no sabrá a qué nos referimos, con lo que nos dará el típico error de JavaScript "document.nombre_imagen no es un objeto".

Declarar un objeto Image dentro del encabezado de la página, dando la ruta de su imagen asociada, se conoce también con el nombre de precarga de imagen, ya que efectívamente, éste es el efecto que se consigue con esta declaración.

Ejemplo

```
<html>
<head>
<title>Imágenes con Javascript</title>

<script type="text/javascript">
<!--

function cambiar_tamano()
 {
  document.img1.width=400;
  document.getElementById("img11").height=500;
  alert("La imagen cambiará");
  document.images[0].src="images/img_2.jpg";
  document.images[1].height=500;

   for(i=0;i<document.images.length;i++)
     alert(document.images[i].name + "  " + document.images[i].id + "\n");
```

```
 }

function ocultar()
 {

  alert("La imagen se ocultará");
  document.images[1].style.display='none';

 }

function cambiar_imagen()
 {
   document.getElementById("img33").src="images/img_Fish.jpg";
 }

function restablecer_imagen()
 {
   document.getElementById("img33").src="images/img_Dragon.jpg";
 }

//-->
</script>

</head>

<body>

<img src="images/img_1.jpg" width="100" height="100" title="Imagen" alt="" name="img1"
id="img11" onclick="cambiar_tamano()"/>

<br />

<img src="images/img_2.jpg" width="100" height="100" title="Imagen" alt="" name="img2"
id="img22" onclick="ocultar()"/>

<br />

<img src="images/img_Dragon.jpg" width="100" height="100" title="Imagen" alt=""
name="img3" id="img33" onmouseover="cambiar_imagen()"
onmouseout="restablecer_imagen()"/>

</body>
</html>
```

Ejemplo

```
<html>
<head>
<title>Imágenes con Javascript</title>

<script type="text/javascript">
<!--

function mostrar_imagen()
 {
        img1 = new Image();
        img1.src = "images/img_Dragon.jpg";
        img1.title="Hola";
        img1.width="500";
        img1.height="500";
        //img1.style.dysplay='block';
        document.images[1].src=img1.src;
        document.images[1].title=img1.title;
        document.images[1].width=img1.width;
        document.images[1].height=img1.height;

}
//-->
</script>

</head>

<body>
<img src="images/img_s.jpg" name="img" id="img" width="200" height="200" title="" alt=""
onclick="mostrar_imagen()" />
<img src="" name="img0" id="img0" width="0" height="0" title="" alt=""
onclick="mostrar_imagen()" />

</body>
</html>
```

MAPAS DE IMÁGENES HTML

Mediante el uso de la etiqueta **<map>…</map>,** es posible crear áreas dentro de la imagen que se asocian a enlaces. El mapeo de imágenes permite crear hipervínculos en partes de la imagen asociadas a una forma o área, la cual se define mediante la etiqueta **<area />** que se anida en la etiqueta de mapa; algunos atributos de esta etiqueta son.

shape: indica la forma del mapa o el tipo de área. Los valores para este atributos son: rect (rectángulo), circle (círculo) y poly (polígono).

coords: especifica las coordenadas donde se posiciona un área específica dentro de la imagen, separadas por comas. Para un rectángulo se establecen cuatro coordenadas, para un círculo se indican tres coordenadas, y para un polígono hay que especificar 6 coordenadas.

href: especifica el recurso de URL que se vinculará al mapa o zona interactiva.

title: muestra un texto alternativo al pasar el mouse sobre el mapa.

Ejemplo

```
<html>
<head>
<title>Mapas de imágenes</title>
</head>

<body>

<map  name="mapa1" id="mapa1">

<area title="Pulsa sobre este círculo" shape="circle" coords="44,36,29" href="#"
onclick="alert('Hola, esta zona corresponde al círculo');"/>

<area title="Pulsa sobre este rectángulo" shape="rect" coords="140,35,31,50" href="#"
onclick="alert('Hola, esta zona corresponde al rectángulo');"/>

<area title="Pulsa sobre este polígono" shape="poly" coords="70,30,77,33,80,25,70,20"
href="#" onclick="alert('Hola, esta zona corresponde al polígono');"/>

<area title="Visitar Google" shape="circle" coords="33,36,31"
href="http://www.google.com"/>

</map>

<img src="images/img_1.jpg" width="380" height="72" title="Mapa de imágenes"
border="0" usemap="#mapa1"/>

</body>
</html>
```

CLASE No. 3 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

TRATAMIENTO DE FORMULARIOS CON JAVASCRIPT

Javascript permite acceder a los formularios y cada uno de sus objetos y manipular los eventos que soportan, facilitando construir páginas interactivas con el usuario y realizando diferentes tipos de tareas, como por ejemplo validaciones de datos, manejo de listas, cálculos, etc. En el curso anterior vimos cómo acceder a ciertas propiedades, métodos y eventos de los objetos de los formularios. Vamos a extender este conocimiento para aprender a manipular listas fijas y desplegables, tan útiles en cualquier aplicación.

Ejemplo

Tratamiento de listas fijas o desplegables creadas con la etiqueta select. La solución la implementaremos con dos páginas, la primera, html, que contendrá el formulario con sus elementos; la segunda, js, que contendrá los scripts para pasar datos de una lista a otra con Javascript. Se hará la validación para no repetir elementos en la lista donde se agrega y también se tendrá la posibilidad de quitar elementos no deseados. Las tareas de agregar o quitar se pueden realizar para un elemento seleccionado o para varios al tiempo.

manejo_listas.html

```
<html>
<head><script type="text/javascript" src="informacion_adicional.js">
</script>

</head>

<body><form name="form1" id="form1" method="post" onSubmit="return
validar_informacion_adicional()">
<select name="lstentretenimiento[]" size="3" id="lstentretenimiento" multiple=”multiple”
style="width:200px;">
        <option>Deporte</option>
          <option>Instrumentos musicales</option>
        <option>Cine</option>
        <option>Literatura</option>
        <option>Viajar</option>
        <option>Escuchar música</option>
          <option>Deportes extremos</option>
          <option>Televisión</option>
          <option>Ver deportes</option>
          <option>Rumba</option>
```

```
      </select>
        <br/>     <input name="btnagregar" type="button" id="btnagregar" value=">"
onClick="adicionar_elemento()" title="Agrega uno o varios elementos">
        <br/>
        <input name="btnquitar" type="button" id="btnquitar" value="<"
onClick="quitar_elemento()" title="Quita uno o varios elementos">
     <br/>
      Seleccionados
          <select name="lsentretenimientoseleccionado[]" size="3"
id="lstentretenimientoseleccionado" multiple style="width:200px;">
     </select>      <br/>
     <input name="btnguardar" type="submit" id="btnguardar" value="Guardar">
     <input type="reset" name="Submit" value="Restablecer"></td>

</form>
</body>
</html>
```

manejo_listas.js

```
// JavaScript Document
function validar_informacion_adicional()
  {
    seleccionar_lista();
    return true;
  }

 function seleccionar_lista()
  {

        for(i=0; i<document.getElementById("lstentretenimientoseleccionado").length; i++)

document.getElementById("lstentretenimientoseleccionado").options[i].selected=true;

  }

 function encontro(e)
  {
    var sw=false;
        var i=0;
        var elem;

          while(i<document.getElementById("lstentretenimientoseleccionado").length &&
!sw)
            {
```

```
elem=document.getElementById("lstentretenimientoseleccionado").options[i].text;
                    if(elem==e)
                      sw=true;
                    else
                      i++;
             }
   return sw;
 }

 function adicionar_elemento()
 {
   var elemento;

        for(var i=0; i<document.getElementById("lstentretenimiento").length; i++)
         {
           if(document.getElementById("lstentretenimiento").options[i].selected)
                 {
                   elemento=document.getElementById("lstentretenimiento").options[i].text;

                        if(!encontro(elemento))
                         {

document.getElementById("lstentretenimientoseleccionado").options[document.getElemen
tById("lstentretenimientoseleccionado").length]=new Option(elemento);
                         }
                        else
                          alert("El ítem " + elemento + " ya se encuentra en la lista");
                 }
         }
 }

 function quitar_elemento()
 {
  var i=0;
        while(i<document.getElementById("lstentretenimientoseleccionado").length)
         {

if(document.getElementById("lstentretenimientoseleccionado").options[i].selected)
                   document.getElementById("lstentretenimientoseleccionado").remove(i);
                 else
                  i++;
    }

 }
```

Ejemplo

Vamos a ilustrar como combinar Javascript con CSS en la validación de formularios, lo que nos permitirá observar la potencia y las posibilidades que se tienen con DHTML en los sitios web. Javascript en realidad puede manipular los atributos de las etiquetas HTML que contienen su atributo *id* definido, en este caso vamos a acceder al atributo *style* que contiene la mayoría de etiquetas, y por consiguiente, utilizamos las distintas propiedades CSS.

formulario_datos.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<script type="text/javascript" src="establecermensajeerror.js"></script>
</head>
<body>
<form action="" method="post">
<input type="text" name="txterror" id="txterror" size="70" style="display:none;text-
align:center;"  readonly="readonly"/>
Nombre
<input type="text" name="txtnom" id="txtnom" />
<br />
<div id="men1" style="display:none;text-align:center;background-
color:orange;color:green;width:200px;">
Bien ingresado
</div>
<br />
<input type="button" value="Validar" onclick="validar_campos()"/>
<input type="button" value="Restablecer"
onclick="parent.location.href='pagina_mensajevalidacion.html'" />
</form>
</body>
</html>
```

establecermensajeerror.js

```
function establecer_mensaje(band,error)
 {
   if (band)
    {
            vista='none';
            vista2='block';
          }
```

```
  else
   {
          vista='block';
          vista2='none';
        }

 document.getElementById("txterror").style.display = vista;
 document.getElementById("txterror").value=error;
 document.getElementById("txterror").style.color="#FFFFFF";
 document.getElementById("txterror").style.background="#000000";

 document.getElementById("men1").style.display=vista2;
}

function validar_campos()
{
  var nom=document.getElementById('txtnom').value;
  if (nom.length>2)
   {
          establecer_mensaje(true,"");
        }
  else
   {
          establecer_mensaje(false,"Debe ingresar el nombre con 3 caracteres como
mínimo");
        }

}
```

MANIPULACIÓN DE VENTANAS

Dado que Javascript accede a toda la jerarquía del DOM, es posible manipular las ventanas del navegador, con lo que podemos abrirlas de acuerdo a ciertas acciones (eventos) del usuario o del sistema, modificar su tamaño y apariencia, entre otros aspectos.

Es común encontrar aplicaciones que manipulen ventanas del navegador, para esto es necesario que nuestros clientes web tengan habilitado el uso de Javascript, ya que de lo contrario no será posible. Por ejemplo los sistemas de educación en línea como Moodle, BlackBoard y otros requieren de este tipo de ventanas para desplegar ciertos contenidos. También pueden ser útiles para mostrar informes o solitar información de los usuarios

La manipulación de ventanas requiere del uso del objeto *window* y su método *open*. La sintaxis para su uso es:

window.open (<*página a mostrar*>, <*nombre ventana*>, <*propiedades de la ventana*>)

Donde cada parámetro indica lo siguiente:

página a mostrar
Especifica el sitio o página que se abrirá en la nueva ventana.

nombre ventana
Indica el nombre que tendrá la nueva ventana que se abrirá tras la ejecución de algún evento.

propiedades de la ventana
Especifica la configuración que tendrá la ventana que se creará. Las propiedades se separan por comas y los valores se indican después del signo igual (=) sin encerrarlos entre comillas. Algunas propiedades son:

fullscreen
Establece si la ventana estará maximizada o no. Los valores válidos son *yes* y *no*.

menubar
Establece si la ventana contiene o no, la barra de menús. Los valores válidos son *yes* y *no*.

width
Establece el ancho de la ventana en píxeles.

height
Establece el alto de la ventana en píxeles.

Al manipular ventanas, es posible que se requiera conocer en tiempo de ejecución la URL de la página o sitio. Para ello, se puede utilizar la propiedad *URL* del objeto *document*. La sintaxis para obtener la URL del sitio o página es:

document.URL

Ejemplo
Se ilustrará con algunos casos la manipulación de ventanas mediante el uso de la instrucción *window.open*. Así mismo se mostrará como obtener la URL del sitio con la instrucción *document.URL*.

```
<html>
<head>
<title>Ventanas con Javascript</title>
<script type="text/javascript">
```

```
<!--
function ver_url()
 {
        alert("La URL de esta página es: " + document.URL);
}

function abrir_ventana1(pag)
 {
        miventana=window.open(pag,'principal1','fullscreen=yes menubar=no');
 }

function abrir_ventana2(pag)
 {
 miventana=window.open(pag,'principal2','width=500, height=300, menubar=yes,
scrollbars=yes');
}
//-->
</script>
</head>
<body>
<a href="#" target="" onclick="ver_url()">Ver URL</a>
<br/>
<a href="#" target="" onclick="abrir_ventana1('pagina1.html');">Abrir ventana1 </a>
<br/>
<a href="#" target="" onclick="abrir_ventana2('pagina2.html');">Abrir ventana2 </a>
<br/>
<a href="http://www.google.com" target="principal1" >Abrir otra página</a>
<br/>
<a href="javascript:window.open('http://www.google.com','p1','width=300, height=300,
resizable=no');">Abrir mas página</a>
</body>
</html>
```

MAQUETACIÓN CON CSS

Las Hojas de Estilos en Cascada CSS ayudan a separar el contenido de la forma, es decir, los elementos que componen una página de la forma con la que se muestran. Además, CSS ayuda en gran medida a la definición de estilos en la página, ya que permite ajustar de una manera mucho más precisa cualquier aspecto de cualquier elemento de la página.

La maquetación con CSS lleva la utilización de las hojas de estilo a su grado máximo, de manera que cualquier definición del aspecto de la página se realiza en la declaración CSS que enlazamos con el documento HTML. Para definir la situación de los elementos en la

página se utilizan las capas (etiquetas div), a las que se aplica un posicionamiento a través de las diferentes propiedades de CSS.

En las capas se introducen los elementos que queramos que aparezcan en ciertas partes de la página. Los elementos que se encuentran dentro de las etiquetas quedan anidados, y así heredan las propiedades y el posicionamiento de la respectiva capa.

En la maquetación por capas se definen etiquetas ; las tablas sólo se utilizan para mostrar información tabulada, es decir, mostrada en filas y columnas. Cabe señalar que en la maquetación tradicional se utilizan las tablas para ajustar la posición de los elementos en la página. Es posible que alguno de los lectores estén familiarizados con el uso de tablas para maquetar una página web, y se habrán dado cuenta de lo poco práctica y las complicaciones que trae para el mantenimiento del código de las páginas web resultantes, dada la rigidez de las tablas, llevando a los diseñadores a realizar artificios en el código para lograr ciertos resultados que involucran construcciones complicadas y tediosas.

Ventajas e inconvenientes de la maquetación CSS
La maquetación por capas o maquetación CSS, permite crear páginas web más funcionales, acogiéndose a las recomendaciones del W3C.

Ventajas
La separación del contenido de la página y del estilo. Tener en cuenta que, cuanto más separemos estos dos elementos, más sencillo será el mantenimiento de las páginas y el procesamiento de la información. Con ello también podremos obtener páginas más limpias y claras.

Ahorro en la transferencia. Si todos los estilos y posiciones de los elementos se introducen en un documento externo, liberaremos el código de la página haciendo que sea mucho más fácil el mantenimiento gracias a que el código estará más claro y "limpio". Como la declaración de estilos se almacena en la caché del navegador, sólo se transfiere en la primera página que se visita del sitio, con lo que la segunda y posteriores páginas que se soliciten se cargarán mucho más rápido.[5]

Facilidad para alterar el aspecto de la página sin tocar el código HTML. Como toda la información de los estilos y el posicionamiento de las capas se encuentran en un mismo archivo, si deseamos cambiar cualquier elemento de la página, ya sea su posición o su aspecto, sólo tenemos que actualizar la hoja de estilos y los cambios se verán automáticamente en toda la página.

Desventajas

[5] Es muy importante tener en cuenta el aspecto de la transferencia, ya que éste generalmente es el servicio que más cuesta en la red, por lo que una optimización de recursos en nuestras aplicaciones, evitará gastos de más.

Compatibilidad con navegadores antiguos. Se necesita que el visitante disponga de un navegador actualizado. La mayoría de los visitantes disponen de navegadores que soportan características avanzadas de CSS, pero todavía hay usuarios que no han actualizado sus equipos o que navegan en sistemas operativos de sólo texto. Los navegadores que no soportan hojas de estilos, leen el código de la página y lo muestran sin ningún posicionamiento. Ello puede resultar incómodo, pero no genera problemas en la visualización de la información de la página, aunque no aparecerán en las zonas indicadas y sin el formato asignado.

Diferencias entre navegadores. Dependiendo del navegador, el comportamiento de las hojas de estilos puede variar, por lo que las páginas pueden no se verse exactamente igual en unos clientes que en otros. Al igual que ocurre con HTML, hay atributos no estándar o que tienen valores por defecto diferentes. Cuando se implementa la maquetación con CSS, puede resultar bastante complicado el ajuste a distintos navegadores, generando bastantes inconvenientes en el diseño al desarrollador o diseñador del sitio.

Dificultad. Sin duda, si estamos acostumbrados al HTML, pasar a CSS resulta más complicado y requiere un estudio más profundo. Sin embargo, este paso nos brindará un mayor control de los elementos de la página y ampliará nuestras fronteras a la hora de maquetar. Además, CSS es un lenguaje, que en general, es fácil de aprender e implementar, por lo que las personas que hayan trabajado con HTML, no deberían tener inconvenientes a la hora de emprender este nuevo estudio.

Ejemplo

```
/*Maquetación con CSS*/

#contenedor
 {
  position:relative;
  width:900px;
  height:1300px;
  left: 40px;
  top: 10px;
  background-color:green;
 }

#cabecera
 {
  /*position:absolute;relative*/
  width:900px;
  height:100px;
  left: 200px;
  top: 100px;
```

```
  color:blue;
  background-color:yellow;
 }

#vinculos
 {
  /*position:absolute;*/
  width:200px;
  height:1200px;
  left: 1000px;
  top: 100px;
  color:orange;
  background-color:black;
 }


#pie
 {
  /*position:absolute;*/
  width:900px;
  height:100px;
  left: 200px;
  top: 1100px;
  color:red;
  background-color:blue;
 }

#contenido
 {
  position:absolute;/**/
  width:700px;
  height:1200px;
  left: 200px;
  top: 100px;
  color:red;
  background-color:gray;
 }
```

CLASE No. 4 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

EXPRESIONES REGULARES EN JAVASCRIPT

Las expresiones regulares (*RegExp*) son modelos que describen las combinaciones de caracteres en el texto. Se podrían definir como una serie de caracteres que forman un patrón, que representan a otro grupo de caracteres mayor, de tal forma que podemos comparar dicho patrón con otros conjuntos de caracteres para ver las coincidencias. Las expresiones regulares pueden utilizarse en la mayoría de lenguajes de programación, entre ellos Javascript.

Las RegExp son utilizadas ampliamente en sistemas de búsquedas de textos y otras operaciones sobre éstos. Estas contienen una serie de símbolos que representan conjuntos de caracteres. La tabla siguiente contiene los caracteres especiales de las expresiones regulares.

Tabla 1. Caracteres especiales utilizados en las Expresiones Regulares

Carácter	Texto buscado
^	Principio de entrada o línea.
$	Fin de entrada o línea.
*	El carácter anterior 0 o más veces.
+	El carácter anterior 1 o más veces.
?	El carácter anterior una vez como máximo (es decir, indica que el carácter anterior es opcional).
.	Cualquier carácter individual, salvo el de salto de línea.
x|y	x o y.
{n}	Exactamente n apariciones del carácter anterior.
{n,m}	Como mínimo n y como máximo m apariciones del carácter anterior.
[abc]	Cualquiera de los caracteres entre corchetes. Especifique un rango de caracteres con un guión (por ejemplo, [a-f] es equivalente a [abcdef]).
[^abc]	Cualquier carácter que no esté entre corchetes. Especifique un rango de caracteres con un guión (por ejemplo, [^a-f] es equivalente a [^abcdef]).
\b	Límite de palabra (como un espacio o un retorno de carro).

\B	Cualquiera que no sea un límite de palabra.
\d	Cualquier carácter de dígito. Equivalente a [0-9].
\D	Cualquier carácter que no sea de dígito. Equivalente a [^0-9].
\f	Salto de página.
\n	Salto de línea.
\r	Retorno de carro.
\s	Cualquier carácter individual de espacio en blanco (espacios, tabulaciones, saltos de página o saltos de línea).
\S	Cualquier carácter individual que no sea un espacio en blanco.
\t	Tabulación.
\w	Cualquier carácter alfanumérico, incluido el de subrayado. Equivalente a [A-Za-z0-9_].
\W	Cualquier carácter que no sea alfanumérico. Equivalente a [^A-Za-z0-9_].

La tabla siguiente contiene algunos de los patrones más utilizados a la hora de validar formularios.

Tabla 2. Expresiones Regulares comunes

Cualquier letra en minuscula	[a-z]
Entero	^(?:\+|-)?\d+$
Correo electrónico	/[\w-\.]{3,}@([\w-]{2,}\.)*([\w-]{2,}\.)[\w-]{2,4}/
URL	^(ht|f)tp(s?)\:\/\/[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*(:(0-9)*)*(\/?)([a-zA-Z0-9\-\.\?\,\'\/\\\+&%\$#_]*)?$
Contraseña segura	(?!^[0-9]*$)(?!^[a-zA-Z]*$)^([a-zA-Z0-9]{8,10})$ (Entre 8 y 10 caracteres, por lo menos un dígito y un alfanumérico, y no puede contener caracteres espaciales)
Fecha	^\d{1,2}\/\d{1,2}\/\d{2,4}$ (Por ejemplo 01/01/2007)
Hora	^(0[1-9]|1\d|2[0-3]):([0-5]\d):([0-5]\d)$ (Por ejemplo 10:45:23)
Número tarjeta de crédito	^((67\d{2})|(4\d{3})|(5[1-

	5]\d{2})|(6011))(-?\s?\d{4}){3}|(3[4,7])\ d{2}-?\s?\d{6}-?\s?\d{5}$
Número teléfono	^[0-9]{2,3}-? ?[0-9]{6,7}$
Código postal	^([1-9]{2}|[0-9][1-9]|[1-9][0-9])[0-9]{3}$
Certificado Identificación Fiscal	^(X(-|\.)?0?\d{7}(-|\.)?[A-Z]|[A-Z](-|\.)?\d{7}(-|\.)? [0-9A-Z]|\d{8}(-|\.)?[A-Z])$

Ejemplo
Pruebas con diferentes patrones de expresiones regulares

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Expresiones Regulares</title>
</head>

<script type="text/javascript">

 function expresion_regular(texto,regexp)
  {
     if(texto.match(regexp))
       alert("Tu patrón coincide en el texto " + texto);
     else
       alert("Tu patrón no coincide en el texto " + texto);
  }

 function principal(boton)
  {
   var re, frase;
   var frase=document.getElementById("texto").value;

    switch(boton)
     {
      case 1:
       re=/^ana/;//También se puede así: var re=/^[ana]/
       break;

      case 2:
       re=/ana$/;
       break;
```

```
      case 3:
       re=/^[ana]$/;
       break;

      case 4:
       re=/^a[.]a$/;
       break;

      case 5:
       re=/a[\d]a$/;
       break;

     }
   alert("Patrón " + boton + ": " + re + "\nTexto suministrado para evaluar el patrón: " + frase);
    expresion_regular(frase,re);
  }
</script>
 <body>
 <form name="form1" method="post" action="">
  <strong>Ingrese su frase:</strong>
  <input name="texto" type="text" id="texto" size="50" maxlength="100">
  <br/>
  <input name="boton" type="button" id="boton" value="Primer patrón" onClick="principal(1)">
   <input name="boton" type="button" id="boton" value="Segundo patrón" onClick="principal(2)">
   <input name="boton" type="button" id="boton" value="Tercer patrón" onClick="principal(3)">
   <input name="boton" type="button" id="boton" value="Cuarto patrón" onClick="principal(4)">
   <input name="boton" type="button" id="boton" value="Quinto patrón" onClick="principal(5)">
</form>
 </body>
</html>
```

Ejemplo

Palídromo con expresiones regulares. Este problema consiste en determinar si una frase ingresadas por teclado es o no un palíndromo

```
function palindromo()
```

```
  {
    var frase=document.getElementById("texto").value.toUpperCase();
    frase=frase.replace(/ /g, '');//Este patrón (expresión regular) permite reemplazar caracteres en una cadena; g indica que buscará todas las ocurrencias (global)

           if (comparar_caracteres(frase))
             alert(document.getElementById("texto").value + " es palíndromo");
           else
             alert(document.getElementById("texto").value + " no es palíndromo");
  }

//*****************************************************************************
  function comparar_caracteres(frase)
   {
          var i, sw;
          i=0;
          sw=true;
            while (i<=frase.length/2 && sw)
                  {
                    if (frase.substr(i,1)==frase.substr(frase.length-i-1,1))
                          i=i+1;
                        else
                          sw=false;
                  }
          return (sw);
        }
```

CLASE No. 5 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

COOKIES EN JAVASCRIPT

Son utilizadas por los servidores web para diferenciar usuarios y para actuar de distinta forma dependiendo de éstos. Su creación inicialmente se dio para “carros de compra” virtuales, donde los usuarios van almacenando los productos que se llevarán en el pedido, con opción de retirarlos de la compra.

Sus principales funciones son:
Llevar el control de usuarios, como por ejemplo al guardar el usuario y contraseña en un cliente, evitando así tener que estar ingresándolos permanentemente.
Conseguir datos sobre las costumbres del usuario al navegar, como por ejemplo para recuperar páginas más rápidamente.

En principio, solo podían ser utilizadas desde los lenguajes del servidor, pero Netscape le dio la capacidad a su lenguaje Javascript para hacer posible que se pudieran procesar desde el cliente, sin embargo esto ocasionó algunos inconvenientes con la seguridad.

Es posible crear, editar, borrar y mostrar una cookie, para ello existen funciones estándar que hacen dicha tarea que pueden implementarse con distintos lenguajes web. Los navegadores poseen opciones que permiten ver y leer el contenido de las cookies.

Ejemplo:
Crear una cookie y mostrar su contenido

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
		<title>Cookies 1</title>

		<script type="text/javascript">
			document.cookie="cookie1=Mi primera cookie";
			document.write(document.cookie);

		</script>
	</head>
	<body>
	</body>
</html>
```

Ejemplo:
Crear una cookie con fecha de expiración y mostrar su contenido

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
		<title>Cookies 1</title>

		<script type="text/javascript">
			document.cookie="'cookie2=Pedro Arias';expires='Wed, 27-Apr-2011
14:55:00 GTM'";
			document.write(document.cookie);
		</script>
	</head>
```

```
        <body>
        </body>
</html>
```

CLASE No. 6 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

QUÉ ES JQUERY

Para simplificar, puede decirse que jQuery es un framework Javascript. Un framework es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones y códigos para realizar tareas comunes; es un conjunto de librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

Por ejemplo, en el caso que nos ocupa, jQuery es un framework para el lenguaje Javascript, luego será un producto que nos simplificará la vida para programar en este lenguaje. Un desarrollador que utiliza Javascript, generalmente tiene que preocuparse por hacer scripts compatibles con varios navegadores y para ello tiene que incorporar mucho código que lo único que hace es detectar el cliente del usuario, para hacer una u otra cosa dependiendo de si es el cliente es Internet Explorer, Firefox, Opera, etc. jQuery nos ayuda en esta labor, puesto que implementa una serie de clases que nos permiten programar sin preocuparnos del navegador del usuario, ya que funciona de igual forma en todas las plataformas de navegación en la web.

Este framework Javascript nos ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery obtendremos ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Cuando se programa Javascript con jQuery se tiene a nuestra disposición una interfaz para programación que nos permitirá hacer tareas con el navegador y estar tranquilos de que funcionarán en todos los navegadores. Simplemente debemos conocer las librerías del framework y programar utilizando las clases, sus propiedades y métodos para la consecución de nuestros objetivos.

Además, jQuery se obtiene de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Para ello simplemente tendremos que incluir en nuestras páginas un script Javascript que contiene el código de jQuery, que podemos descargar de la propia página web del producto y comenzar a utilizar el framework.

http://jquery.com

El archivo del framework ocupa poco espacio (la versión 1.7.2 pesa 257KB, y en general las nuevas versiones siguen siendo muy livianas), lo que es bastante ventajoso y no retrasará la carga de las páginas.

Ventajas de jQuery con respecto a otras alternativas
Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, que básicamente nos sirven para hacer lo mismo. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación muy buena por parte de los programadores y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cuestión muy interesante, es la gran comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc. Uno de los competidores de jQuery, es Mootools, que también posee ventajas similares.

jQuery
Si estás interesado en enriquecer tu página web con componentes de la l Web 2.0, como efectos dinámicos, Ajax, interacción, interfaces de usuario avanzadas, etc., jQuery es una herramienta imprescindible para desarrollar todas estas cosas sin tener que complicarse con los niveles más bajos del desarrollo, ya que muchas funcionalidades ya están implementadas, o bien las librerías del framework permitirán realizar la programación mucho más rápida y libre de errores.

Ahora bien, todas estas mejoras de la web 2.0, que en un principio puede ser muy atractivas, también tienen un coste en tiempo de desarrollo de los proyectos. Sin un framework como jQuery, el tiempo de creación y depuración de todos esos componentes dinámicos sería mucho mayor, sin embargo, lo más complicado de jQuery es aprender a usarlo, igual que pasa con cualquier otro framework Javascript. Requerirá del desarrollador habilidades avanzadas de programación, así como el conocimiento, al menos básico, de la programación orientada a objetos. Una vez aprendido, las ventajas de utilizarlo compensarán más que de sobra el esfuerzo.

Primer script en jQuery

Ejemplo 1
Para empezar vamos a ver este ejemplo, donde tenemos dos botones y un texto. Al pulsar un botón, cambiaremos el texto y al pulsar el otro pondremos otro texto distinto.

En este ejemplo tenemos una capa que tiene este código

Haz clic en un botón

Luego tenemos dos botones con estos códigos:

<input type="button" value="Botón A" onclick="$('#capa').html('Has hecho clic en el botón A')">
<input type="button" value="Botón B" onclick="$('#capa').html('Has presionado un clic en el botón B')">

Como se puede ver, en los botones hay definidos un par de eventos onclick (uno en cada uno) que ejecutan una instrucción Javascript cuando se hace clic sobre ellos. La instrucción está en Javascript, pero hace uso de algunas herramientas disponibles en jQuery para trabajo con los elementos de la página. En este caso, se hace una selección del elemento div de la capa y luego se ejecuta un método sobre él para cambiar el contenido HTML del elemento.

Ejemplo

```
<html>
<head>
	<title>Primer script con jQuery</title>
<script src="../jquery-1.7.2.js" type="text/javascript"></script>
<script>

</script>
</head>
<body>
<div id="capa" style="padding: 10px; background-color: #ff8800">Haz clic en un botón</div>
<br/>
<form>
<input type="button" value="Botón A" onclick="$('#capa').html('Has hecho clic en el botón <b>A</b>')">
<input type="button" value="Botón B" onclick="$('#capa').html('Has presionado un clic en el botón <b>B</b>')">
</form>
</body>
</html>
```

Ejemplo 2

Ahora se tienen dos capas en la página. Una capa estará siempre visible y otra capa estará inicialmente oculta, y se mostrará u ocultará dependiendo de si el usuario pone el ratón encima de la capa que está siempre visible.

En este segundo ejemplo tenemos este código HTML, con las dos capas.

```
<div id="capa" style="padding: 10px; background-color: #ff8800;">Pon el ratón encima de esta capa</div>
<br>
<div id="mensaje" style="display: none;">Has puesto el ratón encima!! <br>(Ahora quítalo de la capa)</div>
```

Ahora vamos a tener un código Javascript con jQuery que defina los eventos del usuario, para cuando sitúa el ratón dentro o fuera de la primera capa.

```
$("#capa").mouseenter(function(evento){
   $("#mensaje").css("display", "block");
});
$("#capa").mouseleave(function(evento){
   $("#mensaje").css("display", "none");
});
```

De esta sencilla manera, hemos creado dos eventos en el div con su respectivo atributo id. Además, se definió el código de los eventos por medio de funciones, que se encargarán de mostrar u ocultar la segunda capa.

La instrucción $("#mensaje").css("display", "block");

Esto nos selecciona la capa con id "mensaje" y con el método css() indicamos que queremos cambiar el atributo "display" al valor "block" de ese elemento.

La instrucción $("#mensaje").css("display", "none");

Esta otra línea muy similar, simplemente cambia el "display" a "none" para ocultar el elemento.

Ejemplo

```
<html>
<head>
        <title>Segundo script con jQuery</title>
<script src="../jquery-1.7.2.js" type="text/javascript"></script>
<script>
$(document).ready(function()
 {
```

```
        $("#capa").mouseenter(function(evento)
        {
                $("#mensaje").css("display", "block");
        });
        $("#capa").mouseleave(function(evento)
        {
                $("#mensaje").css("display", "none");
        });
})
</script>
</head>
<body>
<div id="capa" style="padding: 10px; background-color: #ff8800;">Pasa el mouse sobre esta capa</div>
<br/>
<div id="mensaje" style="display: none; background-color: orange;">Has puesto el mouse sobre la capa <br/>(Ahora quítalo de la capa y mira lo que pasa)</div>
</body>
</html>
```

Como utilizar jQuery en las páginas web

Descargar la última versión del framework

Acceder a la página de jQuery (http://jquery.com) para descargar la última versión del framework. La versión a marzo 1 de 2013 es la 1.9.1; ésta se actualiza periódicamente, por lo que es bueno revisar el sitio si desea actualizar a la versión más reciente.

Dan dos posibilidades para descargar, una conocida como *production (comprimida)*, que es la adecuada para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida. La otra posibilidad es descargar la versión conocida como *develpoment (descomprimida)*, que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos escudriñar en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework.

La descarga proporciona un archivo de texto que guardamos en cualquier editor de texo con extensión js; éste contiene el código completo del framework. Este archivo se ubica en la carpeta deseada para utilizarlo en las distintas páginas.

El archivo lo incluimos en las páginas como cualquier archivo de Javascript, utilizando la etiqueta <script> y su atributo src. El nombre puede ser cualquiera, generalmente, se nombra indicando la versión, por ejemplo, podría ser algo así: *jquery1.7.2.js*.

Ejecutar código cuando la página ha sido cargada

Se trata de detectar el momento en que la página está lista para recibir comandos Javascript que hacen uso del DOM (Document Object Model).

Cuando hacemos ciertas acciones complejas con Javascript tenemos que estar seguros que la página haya terminado de cargar y esté lista para recibir comandos Javascript que utilicen la estructura del documento con el objetivo de cambiar algo, como crear elementos, quitarlos, cambiar sus propiedades, etc. Si no esperamos a que la página esté lista para recibir instrucciones, se corre el riesgo de obtener errores de Javascript en la ejecución.

Generalmente, cuando se desea ejecutar Javascript después de la carga de la página, si no utilizamos ningún framework, lo más normal será utilizar un código como este:

```
window.onload = function () {
   alert("Página cargada");
}
```

Pero esta sentencia, que carga una funcionalidad en el evento onload del objeto window, sólo se ejecutará cuando el navegador haya descargado completamente *todos* los elementos de la página, lo que incluye imágenes, iframes, banners, etc., lo que puede tardar bastante, dependiendo de los elementos que tenga esa página y el peso de éstos.

Pero en realidad no hace falta esperar todo ese tiempo de carga de los elementos de la página para poder ejecutar sentencias Javascript que alteren el DOM de la página. Sólo habría que hacerlo cuando el navegador ha recibido el código HTML completo y lo ha procesado al renderizar la página. Para ello, jQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados completamente. Esto se hace con la siguiente sentencia.

```
$(document).ready(function(){
   //código a ejecutar cuando el DOM está listo para recibir instrucciones.
});
```

Con $(document) se obtiene una referencia al documento (la página web) que se está cargando. Luego, con el método ready(), se define un evento, que se desata al quedar listo el documento para realizar acciones sobre el DOM de la página.

Ejemplo

Insertar un manejador de evento a las etiquetas a (enlaces) que hay en la página.

Para este primer ejemplo vamos a crear un evento clic en el enlace, para mostrar un mensaje cuando se haga clic sobre el link (enlace). Para crear un evento clicc sobre un

elemento tenemos que invocar al método click sobre ese elemento y pasarle como parámetro una función con el código que queremos que se ejecute cuando se hace clic.

```
$("a").click(function(evento){
  //aquí el código que se debe ejecutar al hacer clic
});
```

Como vemos en el código anterior, con $("a") obtenemos una referencia al enlace, lo que hace que se seleccionen todas las etiquetas a (enlaces) del documento. Luego, el método click() sobre el elemento $("a") está definiendo un evento, que se ejecutará cuando se haga clic sobre el enlace. Como se puede ver, al método click se le pasa una función donde se debe poner el código Javascript que queremos que se ejecute cuando se haga clic sobre el enlace.

Ahora veamos la definición del evento clic completa, colocada dentro del evento ready del document, para que se asigne cuando la página está lista.

```
$(document).ready(function(){
  $("a").click(function(evento){
    alert("Has pulsado el enlace. nAhora serás enviado a http://gides.innovasistemas.co");
  });
});
```

A continuación, se describen las instrucciones utilizadas.

```
$(document).ready(function(){
```

Esta línea sirve para hacer algo cuando la página está lista para recibir instrucciones jQuery que modifiquen el DOM.

```
$("a").click(function(evento){
```

Con esta línea estamos seleccionando todas las etiquetas a del documento y definiendo un evento click sobre esos elementos.

```
alert("Has pulsado el enlace. \nAhora serás enviado a http://gides.innovasistemas.co ");
```

Con esta línea simplemente mostramos un mensaje de alerta informando al usuario que se ha hecho clic sobre el enlace.

Una vez que tenemos creada la página con jQuery, se guarda y publica en un navegador. La publicación puede ser desde el servidor, con lo que puede utilizarse en proyectos que incluyan PHP .

Ejemplo

```
<html>
<head>
   <title>Primer script con jQuery</title>
<script src="jquery1.7.2.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
   $("a").click(function(evento){
      alert("Has pulsado el enlace. \nAhora serás enviado a http://gides.innovasistemas.co
");
   });
});
</script>
</head>
<body>
<a href=" http://gides.innovasistemas.co ">GIDES</a>
</body>
</html>
```

```
<html>
<head>
         <title>Script con jQuery</title>
<script src="../jquery1.7.2.js" type="text/javascript"></script>
<script>
$(document).ready(function()
  {
   //código a ejecutar cuando el DOM está listo para recibir instrucciones.
   alert("Hola, la página está lista");
  });

 $(document).ready(function(){
   $("a").click(function(evento){
      alert("Has pulsado el enlace..\nNo serás enviado a otra página");
      evento.preventDefault();
   });
});
</script>
</head>
<body>
<a href=" http://gides.innovasistemas.co ">GIDES</a>
<br/>
<a href="http://jquery.com">jQuery</a>
```

```
</body>
</html>
```

Bloquear el comportamiento normal de un enlace

Ahora veamos una pequeña modificación para alterar el comportamiento por defecto de los enlaces. Como sabemos, cuando se pulsa un enlace nos lleva a una URL. Al hacer clic, primero se ejecuta lo que hayamos colocado en el evento click del enlace y luego el enlace lleva al navegador a una nueva URL.

Este comportamiento se puede bloquear también desde jQuery, añadiendo una llamada al método preventDefault() sobre el evento. La función definida para marcar el comportamiento del evento click sobre el enlace recibe un parámetro. Ese parámetro es un manejador de evento, y tiene sus propios métodos y propiedades, como preventDefault() que utilizamos en el ejemplo anterior. Su uso es el siguiente:

```
$(document).ready(function(){
  $("a").click(function(evento){
    alert("Has pulsado el enlace, en envío a la URL se cancela");
    evento.preventDefault();
  });
});
```

Como hemos podido ver, invocando el evento.preventDefault(), lo que se consegue es que el link no lleve a ningún sitio, simplemente se ejecutará el código Javascript contenido para el evento click.

CLASE No. 7 SCRIPTS AVANZADOS DEL LADO DEL CLIENTE

Añadir y quitar clases CSS a elementos de la página

jQuery puede alterar elementos de una página web, añadiendo y quitando clases CSS. Esto se implementa fácilmente, ya que en jQuery los elementos tienen dos clases llamadas addClass() y removeClass(), que sirven justamente para que el elemento que recibe el método se le aplique una clase CSS o se le elimine.

En este ejemplo vamos a tener dos elementos. Primero una capa (div) con un texto. Después tendremos un enlace que estará fuera de la capa. Al situar el mouse sobre un enlace añadiremos una clase CSS a la capa y al retirarlo del enlace eliminaremos la clase CSS que se había añadido antes.

Ejemplo

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
		<title>Estilos CSS: A&ntilde;adir y quitar propiedades</title>

	<script type="text/javascript" language="JavaScript" src="../jquery1.7.2.js">
	</script>

	<script type="text/javascript" src="estilos_anadirquitarcss.js">
	</script>

	<style type="text/css">
		a{
		  text-decoration:none;
		  color:green;
		 }
		.clasecss{
	background-color: #ff8800;
	font-weight: bold;
		 }

		.estilo2{
			color:orange;
			text-decoration:underline;
		 }

	</style>
	</head>

	<body>
		<h1 style="text-align:center;">Estilos CSS: A&ntilde;adir y quitar
propiedades </h1>

		<div id="capa">
		  Esta capa es independiente y voy a añadir y eliminar clases css sobre ella
		</div>

		<br>
		<br>
		<form>
```

```
                    <input type="button" value="Ocultar enlace" id="boton1" name="boton1">

                </form>
                <a href="http://www.google.com">Anadir y quitar clase en la capa de arriba</a>
                <br>
                <div id="enlace1">
                <a href="http://www.yahoo.com">Yahoo</a>
                </div>
                <br>
                <a href="http://www.gmail.com" id="enlace2">Gmail</a>
        </body>
</html>


$(document).ready(function(){
  $("a").mouseover(function(event){
    $("#capa").addClass("clasecss");
         $("a").addClass("estilo2");
  });

  $("a").mouseout(function(event){
    $("#capa").removeClass("clasecss");
         $("a").removeClass("estilo2");

  });

  $("#boton1").click(function(){
   $("#enlace1").css("display","none");
        $("#enlace2").css("display","none");
  })
});
```

SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR CON PHP

CLASE No. 8 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

Entraremos ahora a explorar algunas características avanzadas del lenguaje PHP abriendo más las posibilidades para desarrollar con este potente lenguaje. Los temas que desarrollaremos están relacionados con el manejo de archivos, la POO (Programación Orientada a Objetos) en PHP, frameworks para PHP y aplicaciones entre otros temas.

MANEJO DE ARCHIVOS

PHP tiene la capacidad de manipular archivos, con lo que le da la capacidad a las aplicaciones de gestionarlos ya sea para crear, editar, leer o escribir; también hay otras operaciones que pueden realizarse, como elimnar el archivo, copiarlo, moverlo, etc. Para ello PHP cuenta con una serie de funciones que permiten su tratamiento.

Funciones para el tratamiento de archivos
Veamos ahora algunas funciones para tratar archivos mediante ejemplos:

Cambiar los permisos de un archivo
```
chmod ('archivo.ext', 0777);
```

Cambiar el nombre del propietario de un archivo
```
chown ('archivo.ext', 'Mi nombre');
```

Copiar un archivo
```
copy('archivo.ext', '/directorio_destino/');
```

Obtener el directorio inicial de un archivo o subdirectorio
```
dirname('directorio_raiz/subdirectorio');
```

Obtener el espacio disponible en un directorio en Bytes.
```
disk_free_space('directorio');
```

Verificar si un archivo existe
```
file_exists ('archivo.ext');
```

Obtener la hora del último acceso a un archivo
```
fileatime('archivo.ext');
```

Obtener la hora de la ultima modificación de un archivo
```
filemtime('archivo.ext');
```

Obtener el dueño de una archivo

```
fileowner ('archivo.ext');
```

Obtener los permisos de una archivo

```
fileperms ('archivo.ext');
```

Obtener el tamaño en bytes de un archivo

```
filesize('archivo.ext');
```

Obtener el tipo de archivo

```
filetype('archivo.ext');
```

Abrir un archivo o dirección URL

```
fopen ('archivo.ext', 'r');
```

Saber si un archivo es ejecutable

```
is_executable('archivo.ext');
```

Indicar si el archivo se puede leer

```
is_readable('archivo.ext');
```

Indica si un archivo fue subido mediante HTTP POST upload

```
is_uploaded_file ('archivo.ext');
```

Saber si un archivo se puede escribir

```
is_writable ('archivo.ext');
```

Crea un directorio con permisos.

```
mkdir('/nuevo_directorio', 0777);
```

Mover un archivo cargado a través de HTTP POST de un directorio a otro predefinido

```
move_uploaded_file ('directorio/archivo.ext', 'directorio2/archivo.ext');
```

Leer el contenido resultante de archivo

```
file_get_contents('archivo.ext');
```

Renombrar un archivo o directorio

```
rename("/tmp/archivo_temp.txt", "/home/usuario/login/docs/mi_archivo.txt");
```

Eliminar un directorio

```
rmdir('/directorio');
```

Abrir un directorio

```
opendir('/directorio');
```

Leer un directorio

```
readdir('/directorio');
```

Crear un archivo temporal

```
tmpfile();
```

Eliminar un archivo

```
unlink('archivo.ext');
```

Nota

Alguna de estas funciones requieren tener permisos 0777 chmod para poder trabajar correctamente.

La mayoría de ellas devuelven un booleano, por lo cual se puede imprimir un resultado en pantalla de la siguiente manera:

```
<?
if(unlink('archivo.txt')
  echo 'Se eliminó el archivo';

else
echo 'Imposible eliminar el archivo, compruebe que exista y que tenga
permisos.';

?>
```

Recuerde que para establecer permisos a directorios o archivos se deben emplear 4 cifras y no 3 como usualmente se hace y que produce error.

Ejemplos:
0777, 0755, 0666, etc.

Esto se debe a que el número para especificar el permiso debe incluirse en octal (base 8). En PHP y otros lenguajes de programación. Los números en base 8 se escriben con un 0 adelante para diferenciarlos de los números en base 10 o sistema decimal.

CLASE No. 9 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

Trabajando con archivos

Otra de gran ventaja de PHP es la manipulación de archivos internos y remotos. Para ello se utiliza una función esencial *fopen()* que se encarga de abrir un archivo o URL, basándose en una serie de parámetros que iremos describiendo.

La sintaxis de esta función con la que trabajaremos es la siguiente:

fopen (*'<archivo.ext>','<Modo>'*);

Donde *archivo.ext* es el nombre del archivo físico junto con su extensión; cabe destacar que si la llamada de este archivo se hace de la siguiente forma: *esquema://*, se asume que es una URL y PHP buscará un gestor de protocolo para tal esquema. Los protocolos pueden ser: http, https, ftp, ftps, pop, pop3, entre otros. Si PHP no encuentra una o más envolturas registradas para dicho protocolo, se emitirá un mensaje de error.

Modo, se refiere a la forma en que abrimos el archivo (bajo que recursos). Por ejemplo si queremos hacer una apertura para solo lectura, o lectura y escritura u otro. Estos modos se designan con letras predefinidas en PHP, que son las siguientes:

'r' Apertura para sólo lectura; ubica el apuntador de archivo al comienzo del mismo.

'r+' Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo del mismo.

'w' Apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud de cero. Si el archivo no existe, intenta crearlo.

'w+' Apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste y lo trunca a una longitud cero. Si el archivo no existe, intenta crearlo.

'a' Apertura para sólo escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo.

'a+' Apertura para lectura y escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo.

'**x'** Creación y apertura para sólo escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a *fopen()* fallará devolviendo *false* y generando un error de nivel *E_WARNING*. Si el archivo no existe, intentará crearlo. Esta opción es soportada desde PHP 4.3.2 y versiones posteriores, y sólo funciona con archivos locales.

'x+' Creación y apertura para lectura y escritura; ubica el apuntador de archivo al comienzo de éste. Si el archivo ya existe, la llamada a *fopen()* fallará devolviendo *false* y generando un error de nivel *E_WARNING*. Si el archivo no existe, intentará crearlo. Esta opción es soportada en PHP 4.3.2 y versiones posteriores, y sólo funciona con archivos locales.

Ejemplo

```
<?
$archivo= fopen("documento.txt" , "r");
if ($archivo)
 {
while (!feof($archivo))
{
echo fgets($archivo, 255) . '<br>';
}
}
fclose ($archivo);
?>
```

Observación:
Primero abrimos el archivo en modo de lectura y lo guardamos en la variable *$archivo*, si se puede realizar la apertura (o sea si *fopen()* devuelve *true*), ejecutamos un bucle *while* y en el argumento introducimos la función *feof()*, encargada de verificar si el apuntador a un archivo está al final del mismo. Luego la función *fgets()*, se encarga de imprimir la línea del archivo apuntado; y finalmente, cerramos el archivo con *fclose()*.

Ejemplo

```
<?
$archivo=fopen("documento.txt" , "w");
if ($archivo)
{
fputs ($archivo, "Probando el script PHP");
}
fclose ($archivo);
?>
```

Observación:
Abrimos el archivo nuevamente, pero esta vez en modo de solo Escritura. Si se puede ejecutar ese proceso *if($archivo)*, agregamos una línea con *fputs()*, que tendrá como contenido “Probando el script PHP”.

El último paso es cerrar el archivo. Cabe destacar que para que este último ejemplo funcione, obviamente el archivo documento.txt, requiere tener permiso de escritura, preferiblemente 0777 CHMOD.

Veamos ahora cómo se puede comprobar mediante PHP si un archivo existe o no en el sistema de archivos del servidor, es decir, dentro del disco duro del servidor. Esta es una tarea que se debería comprobar antes de realizar acciones con cualquier fichero del servidor, porque, si no existe el fichero, las acciones a realizar sobre él no tendrían sentido.

En PHP existe una función que nos ayudará con la tarea de comprobar si existe o no un archivo en el servidor. Solamente tenemos que hacer uso de ella para saber si existe ese archivo y por tanto, operar con él.

```
file_exists(nombre_del_fichero)
```

La función devuelve un boleano, indicando con *true* que existe el archivo y con *false* que no existe. Recibe una cadena de caracteres que es el nombre del archivo que se desea comprobar.

En principio, si sólo se indica un nombre de un archivo, se supone que se debe buscar su existencia dentro del mismo directorio donde se aloja la página PHP.

Ejemplo
Determinar si existe un archivo llamado "miarchivo.txt" en el mismo directorio donde se encuentra alojada la página PHP:

```
<?
if (file_exists("mifichero.txt"))
{
  echo "El fichero existe";
}
Else
{
  echo "El fichero no existe";
}
?>
```

Si quisiéramos comprobar si existe ese archivo dentro de un directorio determinado, bastaría con que escribiésemos la ruta de directorios sobre la que deseamos buscar el fichero.

```
file_exists("micarpeta/miarchivo.txt")
```

Esta función buscaría el archivo "miarchivo.txt" dentro del subdirectorio "micarpeta", el cual está contenido dentro del directorio donde tenemos el script PHP.

Indicando rutas absolutas

También podríamos buscar un archivo indicando una ruta absoluta dentro del sistema de archivos del servidor. En este caso habría que comprobar si desde PHP tenemos permisos para acceder a la ruta en la que se pretende buscar el archivo, sobretodo si la ruta es externa al directorio de publicación del servidor.

```
file_exists("c:/miarchivo.txt");
```

Esta función buscaría el archivo *miarchivo.txt* en el directorio raíz del disco *c:* Habría que ver si tenemos permisos para acceder al directorio raíz del disco.

Podemos averiguar la ruta del directorio de publicación de la página con la variable del entorno del servidor:

$_SERVER["DOCUMENT_ROOT"]

Podemos utilizar esta variable para generar la ruta absoluta a un archivo que pretendamos comprobar su existencia.

```
file_exists($_SERVER["DOCUMENT_ROOT"] . "\miarchivo.txt")
```

Esta función buscará el archivo "*miarchivo.txt*" dentro de la raíz del directorio de publicación.

Nota
Como se puede ver, hemos utilizado el carácter "\" (backslash o contrabarra) para separar los directorios en la ruta de un archivo. Esto es posible sobre el sistema Windows, donde se utilizan backslash para escribir rutas, mientras que en Linux o en UNIX se utiliza el carácter "/" (slash o barra normal). Para evitar problemas, utilizamos el slash (/), dado que también funciona en sistemas Windows, mientras que el backslash (\) solo puede usarse en este sistema.

Ejemplo
Verificar si existe un archivo y abrirlo; si no existe, crearlo

Ahora vamos a mostrar un código típico que sirve para abrir un archivo con un nombre dado. Primero hay que comprobar si existe o no el archivo. Si existe, entonces lo abrimos tal cual está y si no existe, lo crearemos previamente a abrirlo.

```
<?
if (file_exists("miarchivo.txt"))
{
  echo "El archivo existe.  Abrimos el archivo";
  $archivo1 = fopen("miarchivo.txt", "a");
}
else
{
  echo "El archivo no existe.  Creamos y abrimos el archivo.";
  $archivo1 = fopen("miarchivo.txt", "w+");
}
```

```
//operar con el fichero
fwrite ($archivo1,"Nuevo texto para el archivo");
//cerrar el fichero
fclose ($reffichero);

?>
```

En este código, primero se comprueba la existencia del fichero. Si existe (caso afirmativo del if), se abre con el modo "*a*", que indica que se desea añadir información al fichero. Si no existe fichero (caso else del if), se abre con el modo "*w+*", que indica lectura y escritura y además indica que si no existe el fichero, se creará en el sistema de archivos en la ruta indicada. Después de que el archivo está abierto para escritura, podemos operar sobre él (escribiendo información), y luego al final cerrarlo.

CLASE No. 10 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

Ejemplo
Vamos a crear un archivo XML mediante una página PHP; aplicaremos varias de las funciones vistas para el tratamiento de archivo.

Ejemplo

```
<?php
$arreglo_nombres=array("Nombre"=>"Antonio","Apellido"=>"Zapata","Direccion"=>"Centro
","Telefono"=>"11111111","correo"=>"micorreo@nose.tal");
$xml='<?xml version="1.0" encoding="utf-8"?>
        <!--Con este script se crea un archivo XML en el disco-->
         <archivo_xml>';
  while (list ($etiqueta, $contenido) = each ($arreglo_nombres)):
    $xml.="<$etiqueta>$contenido</$etiqueta>";
  endwhile;
  $xml.="</archivo_xml>";
  $archivo=fopen("archivo.xml","w+");
  fwrite ($archivo,$xml);
  fclose($archivo);
echo "<br/>
     <div style='font-size:15px; background:green; width:300px;text-
align:center;padding:20px;border:2px solid black; top:1000px; float:left;'>
          <a href='archivo.xml' style='color:white; font-weight:bold;'>Ver archivo XML</a>
          </div>";

?>
```

CLASE No. 11 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO) EN PHP

La Programación Orientada a Objetos (POO) es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando *clases*. Una *clase* es un ente abstracto formado de un conjunto de *datos (atributos o propiedades*) y *funciones (métodos)* que operan sobre dichos datos. Las clases son definiciones, a partir de las que se crean *objetos*. Los *objetos* son ejemplares (tipos) de una clase determinada y como tal, disponen de los datos y funciones definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelizar los sistemas. Prueba de ello es la versión de PHP 5, que implanta la POO como metodología de desarrollo.

Ejemplo

Se ilustra a continuación la implementación de clases en PHP. Este ejemplo contiene varias páginas que se guardan en distintas carpetas.

Solución

En el *document root* (htdocs o la carpeta de su distribución), cree una carpeta llamada *ejemploclasesphp* y allí cree otras dos carpetas: *class* y *pages*. En la carpeta *pages*, guarde el archivo de nombre *pag1.php* y en la carpeta *class* guarde los archivos *colores.php*, *linea.php* y *misclases.php*. A continuación se presenta el código respectivo de cada página.

pag1.php

```
<html>
<head>
<title>Majeno de clases en PHP</title>
</head>
<body>
<?php
include('../class/misclases.php');
$linea1=new linea(0,'');
$color=new colores();
  for ($i = 0; $i <= 14; $i++)
   {
    $linea1->longitud=$i+1;
    $linea1->color=$color->color[$i];
    print("<div style='text-align:center;'>");
```

```
   $linea1->dibujar_linea();
   print("</div>");
  }
?>
</body>
</html>
```

misclases.php

```
<?php
$dir=opendir("../class");
  while ($archivo=readdir($dir))
    if ($archivo!="." && $archivo!=".." && $archivo!="misclases.php")
      include_once($archivo);

closedir($dir);
?>
```

colores.php

```
<?php
//Definición de la clase
 class colores
  {
   //Propiedades (variables) de la clase
   var $color;

    //Métodos de la clase

   //Constructor de la clase
   function colores()
    {
          $this->color=
                array(
                                    0=>'gold',
                                    1=>'red',
                                    2=>'yellow',
                                    3=>'orange',
                                    4=>'pink',
                                    5=>'black',
                                    6=>'violet',
                                    7=>'gray',
                                    8=>'green',
                                    9=>'brown',
                                    10=>'cyan',
                                    11=>'maroon',
                                    12=>'turquoise',
```

```
                                13=>'olive',
                                14=>'blue'
                        );
                }
        }
?>
```

linea.php

```
<?php
//Definición de la clase
 class linea
  {
   //Propiedades (variables) de la clase
   var $longitud;
   var $color;

   //Métodos de la clase

   //Constructor de la clase
   function linea($l,$c)
    {
     $this->longitud=$l;
     $this->color=$c;
    }

    //Método para dibujar la línea
    function dibujar_linea()
    {
      for ($i=1;$i<=$this->longitud;$i++)
       {
         print("<span style='color:$this->color;'>___</span>");
       }
    }
  }
?>
```

CLASE No. 12 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

Manejo de bases de datos mediante el uso de clases en PHP

Ejemplo

Vamos ahora a crear un pequeño sistema que maneje bases de datos. Lo novedoso en este ejemplo, será la conexión a la base de datos y las consultas a ella por medio de una

clase que se encargue de manejar estos aspectos. La aplicación utilizará la base de datos *bdusuarios* que contiene la tabla *usuario(usr, nombre, correo, clave)*, y estará compuesta de varios archivos, que los enunciaremos a continuación:

conexionbd.php

```
<?php
 class cConexion
 {
        var $servidor;
        var $usuario;
        var $contrasena;
        var $link;
        var $basedatos;
        var $sql;

        function cConexion()
        {
                $this->servidor="";
                $this->usuario="";
                $this->contrasena="";
                $this->link="";
                $this->basedatos="";
                $this->sql="";
        }

        function Asignar_Servidor($host)
        {
                $this->servidor=$host;
        }

        function Asignar_Usuario($user)
        {
                $this->usuario=$user;
        }

        function Asignar_Contrasena($pass)
        {
                $this->contrasena=$pass;
        }

        function Asignar_Basedatos($bd)
        {
                $this->basedatos=$bd;
        }
```

```
        function Devolver_Servidor()
        {
                return $this->servidor;
        }

        function Devolver_Usuario()
        {
                return $this->usuario;
        }


        function Devolver_Contrasena()
        {
                return $this->contrasena;
        }

        function Devolver_Basedatos()
        {
                return $this->basedatos;
        }

        function Conectar()
        {
                if(!($this->link=mysql_connect($this->servidor,$this->usuario,$this-
>contrasena)))
                {
                  echo "No se puede conectar con el servidor";
                  exit();
                }

                if(!mysql_select_db($this->basedatos, $this->link))
                {
                  echo "Base de datos no válida";
                  exit();
                }

                //return $this->link;
        }

        function Cerrar_Conexion()
        {
                mysql_close($this->link);
        }

        function Liberar_Consulta()
```

```
        {
                mysql_free_result($this->sql);
        }

   function Asignar_Consultatabla($consulta)
    {
                $this->sql=mysql_query($consulta,$this->link);
        }

        function Devolver_Consultatabla()
    {
                return $this->sql;
        }

   function Devolver_Totalregistros()
    {
          return mysql_num_rows($this->sql);
        }

        function Guardar_Registro($tabla, $campos)
        {
                $consulta="insert into " . $tabla;
                mysql_query($consulta,$this->link);
        }

        function Eliminar_Registro($tabla,$clausula)
        {
                $consulta="delete from " . $tabla . " where " . $clausula;
                mysql_query($consulta,$this->link);
        }
  }//fin de la clase

?>
```

establecerconexion.php

```
<?php
include("conexionbd.php");
$con=new cConexion();
$con->Asignar_Servidor("localhost");
$con->Asignar_Usuario("root");
$con->Asignar_Contrasena(""); /*Si tiene contraseña para el usuario MySQL debe especificarla*/
$con->Asignar_Basedatos("bdusuarios");
```

```
?>
```

mostrar_registros.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title>Registros</title>
</head>

<body>

<?php
//Datos de la conexión
include_once("establecerconexion.php");

$con->Conectar();

$con->Asignar_Consultatabla("select * from usuario");

$nreg=$con->Devolver_Totalregistros();

?>

<div id="capatabla" style="text-align:center;">
<table border="1" >
<caption align="bottom">
Total registros:
<span style="color:#090;">
<?php
  print $nreg;
?>
</span>
</caption>

<tr>
<th>Usuario</th>
<th>Nombre</th>
<th>Correo</th>
<th>Eliminar</th>
</tr>

<?php
$result=$con->Devolver_Consultatabla();
  while($fila=mysql_fetch_array($result))
```

```
  {
        printf("<tr>
                        <td>%s</td>
                            <td>%s</td>
                            <td>%s</td>
                            <td>
                              <a href='eliminar_registros.php?usr=%s'>
                                Eliminar
                              </a>
                            </td>
                </tr>",
                        $fila['usr'],$fila['nombre'],$fila['correo'],$fila['usr']);
  }

?>

</table>
</div>

<?php
$con->Liberar_Consulta();
$con->Cerrar_Conexion();
?>

</body>
</html>
```

eliminar_registros.php

```
<?php
include("establecerconexion.php");
$usr=$_GET['usr'];
$con->Conectar();
$con->Eliminar_Registro("usuario","usr='$usr'");

print "<script>
        window.location='mostrar_registros.php';
            </script>";
?>
```

CLASE No. 13 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

COMBOS DEPENDIENTES

Es común encontrar la necesidad de crear una lista dinámicamente y que a su vez se cree otra a partir de la primera. Esto se conoce como "*combos dependientes*", y requiere de adicionar un evento en la etiqueta de lista que ejecutará Javascript.

Ejemplo: combos dependientes
Vamos a utilizar la base de datos bdpaises que se describe en el siguiente script, junto con el código de la página que mostrará como trabajan las listas dependientes.

scriptbd_paises.sql

```
create table pais(
 nompais varchar(30) primary key not null
 )type=innodb;

create table departamento(
  coddepto varchar(5) primary key not null,
  nomdepto varchar(30) not null,
  nompais varchar(30) not null,
  index (nompais),
  foreign key (nompais) references pais(nompais)
    on delete cascade
          on update cascade

 )type=innodb;

create table ciudad(
  codciudad varchar(5) primary key not null,
  nomciudad varchar(30) not null,
  coddepto varchar(30) not null,
  index (coddepto),
  foreign key (coddepto) references departamento(coddepto)
    on delete cascade
          on update cascade

 )type=innodb;

 insert into pais(nompais)
   values("Colombia"),
        ("Argentina"),
                  ("Brasil");

 insert into departamento(coddepto, nomdepto, nompais)
   values("01","Antioquia","Colombia"),
        ("02","Atlántico","Colombia"),
```

```
                ("03","Cundinamarca","Colombia"),
        ("04","Buenos Aires","Argentina");


 insert into ciudad(codciudad, nomciudad, coddepto)
   values("001","Medellín","01"),
        ("002","Envigado","01"),
                ("003","Bello","01"),
        ("004","Buenos Aires","04");
```

cargar_combos.php

```
<html>
<head>
<title>Combos dependientes</title>
</head>
<body>
<?
  mysql_connect("localhost","root","");
  $sql="SELECT * FROM pais";
  $result=mysql_db_query("dbpaises",$sql);
  //print "Total países: " . mysql_num_rows($result);
?>
<form name="form1" method="post">
 <input type= "text" name="t1" id="t1" size="100"/>
 <input type= "text" name="t2" id="t2" size="100"/>
 <input type= "text" name="t3" id="t3" size="100"/>
 <br/>
País
 <select name='pais' size='1' id='pais' onChange='javascript:document.form1.submit();'>
   <option value='Seleccione pais'>Seleccione pais</option>

   <?
      while ($row=mysql_fetch_array($result))
       {
        if ($row['nompais'] == $_POST['pais'])
                {
          echo "<option value='".$row['nompais']."'
selected='selected'>".$row['nompais']."</option>";
         }
        else
                {
          echo "<option value='".$row['nompais']."'>".$row['nompais']."</option>";
         }
```

```
      }

      $pais=$_POST['pais'];
      $tp=mysql_num_rows($result);

   ?>
 </select>

<?
     $sql="select * from departamento where nompais='$pais'";
     $result1=mysql_db_query("paises",$sql);
?>
Departamento
 <select name="depto" size="1" id="depto"
onChange='javascript:document.form1.submit();' >
   <option value="Seleccione departamento" selected="selected">Seleccione
departamento </option>

   <?
      while ($row1=mysql_fetch_array($result1))
            {
             if ($row1['coddepto'] == $_POST['depto'])
              {
   ?>
         <option value="<? echo $row1['coddepto']; ?>" selected="selected"><? echo
$row1['nomdepto']; ?></option>;

   <?
          }
         else
          {
   ?>
         <option value="<? echo $row1['coddepto']; ?>"><? echo $row1['nomdepto'];
?></option>;

   <?
        }
      }

    $td=  mysql_num_rows($result1);
     $depto=$_POST['depto'];

   ?>
</select>
```

```
<?
    $sql="select * from ciudad where coddepto='$depto'";
   $result2=mysql_db_query("paises",$sql);
?>

Ciudad
<select name="ciudad" size="1" id="ciudad"
onChange='javascript:document.form1.submit();' >
  <option value="Seleccione ciudad" selected="selected">Seleccione ciudad </option>

  <?
     while ($row2=mysql_fetch_array($result2))
      {
       if ($row2['codciudad'] == $_POST['ciudad'])
          {
  ?>
  <option value="<? echo $row2['codciudad']; ?>" selected="selected"><? echo
$row2['nomciudad']; ?></option>;

  <?
       }
     else
       {
  ?>

  <option value="<? echo $row2['codciudad']; ?>"><? echo $row2['nomciudad'];
?></option>;

  <?
       }

      }

   $tc=  mysql_num_rows($result2);
   $ciudad=$_POST['ciudad'];

  echo "<script>
              document.getElementById('t1').value='$pais Total: $tp';
              document.getElementById('t2').value='$depto Total: $td';
              document.getElementById('t3').value='$ciudad Total: $tc';
         </script>";
  ?>
</select>
```

```
</form>
</body>
</html>
```

CLASE No. 14 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

FRAMEWORKS PARA PHP

Existe gran cantidad de frameworks para PHP, disponibles en la red para su descarga y uso. La mayoría implementa el MVM (Modelo Vista Controlador), lo cual permite desarrollar aplicaciones acordes a las exigencias actuales, así como aprovechar las ventajas que trae utilizar este tipo de herramientas, entre ellas, el orden que le añade a las aplicaciones, las bibliotecas de funciones disponibles, la inclusión de aspectos de seguridad, la optimización de la ejecución de las páginas, URL amigables, entre otros aspectos.

Veamos algunos frameworks PHP[6]

Nombre del Framework	Versión PHP	MVC
CodeIgniter	4, 5	Si
CakePHP	4, 5	Si
Fusebox	4, 5	Si
Symfony	5	Si
eZ Components	5	No
Prado	5	Si
Zend	4, 5	Si
Yii	5	Si
Kumbia	5	Si
Akelos	4, 5	Si

Esta es solo una pequeña lista de toda la que existe en cuanto a frameworks para PHP; sin embargo, lo principal siempre es un buen manejo del código PHP y comprender cual es su filosofía y forma de trabajo. Decidir con que framework trabajar lo decide el gusto (después de ensayar con varios) y/o la necesidad (algunas empresas utilizan uno u otro, que quizá sea necesario aprender).

Lo más recomendable en la actualidad, es seleccionar un framework MVC, dado que el desarrollo de aplicaciones está orientado hacia esta metodología; al conocer el código PHP y manejar un framework MVC, la transición hacia otro no será complicada.

[6] Datos tomados de http://www.phpframeworks.com

FRAMEWORK CODEIGNITER

Un framework es un programa para desarrollar otros programas, CodeIgniter, por tanto, es un programa o aplicación web desarrollada en PHP para la creación de cualquier tipo de aplicación web bajo PHP. Es un producto de código libre, libre de uso para cualquier aplicación.

Como cualquier otro framework, CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que debemos seguir para obtener provecho de la aplicación. Esto es, marca una manera específica de codificar las páginas web y clasificar sus diferentes scripts, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el proceso de desarrollo llamado Model View Controller (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales. Este sistema tiene sus características, que veremos más adelante.

CodeIgniter no lo hace todo por uno, pero contiene muchas ayudas para la creación de aplicaciones PHP avanzadas, que hacen que el proceso de desarrollo sea más rápido. A la vez, define una arquitectura de desarrollo que hará que programemos de una manera más ordenada ya que contiene diversas herramientas que ayudan a hacer aplicaciones más versátiles y seguras.

CodeIgniter y otros frameworks PHP pueden ayudar a dar el salto definitivo como desarrollador PHP, creando aplicaciones web más profesionales y con código más reutilizable, con la diferencia que CodeIgniter está creado para que sea fácil de instalar en cualquier servidor y de empezar a usar, más que cualquier otro framework. Además, muchas de sus utilidades y modos de funcionamiento son opcionales, lo que hace que se goce de mayor libertad a la hora de desarrollar sitios web.

Características generales de CodeIgniter

Algunos de los puntos más interesantes sobre este framework, sobre todo en comparación con otros productos similares, son los siguientes:

Versatilidad: Quizás la característica principal de CodeIgniter, en comparación con otros frameworks PHP. CodeIgniter es capaz de trabajar en la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo se tiene acceso por FTP para enviar los archivos al servidor y donde no se tiene acceso a su configuración.

Compatibilidad: CodeIgniter es compatible con la versión PHP 4 y PHP 5. Sin embargo, desde la versión 2 de CodeIgniter ya solo es compatible con la versión 5 de PHP. La

versión antigua del framework (como la 1.7 u otra) se encuentra disponible para los que aun utilicen PHP 4.

Facilidad de instalación: No es necesario más que la cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con la edición de un archivo, donde debemos escribir aspetos tales como el acceso a la base de datos. Si la instalación es local, es similar, solo descomprimimos el paquete en la carpeta donde vamos a trabajar con el framework.

Flexibilidad: CodeIgniter es menos rígido que otros frameworks. Define una manera de trabajar específica, aunque podría no seguirse ya que sus reglas de codificación muchas veces las podemos saltar para trabajar como más a gusto nos sintamos. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda también a que el aprendizaje sea más sencillo al principio.

Ligereza: El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

Documentación: La documentación de CodeIgniter es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando ya sabemos acerca del framework y queremos consultar sobre una función o un método en concreto, pero para iniciar sin duda es muy importante.

Instalación y configuración
Lo primero es descargar el framework del sitio de CodeIgniter, del cual se obtiene un archivo empaquetado .zip. La versión disponible (marzo de 2013) es la 2.1.3. El sitio oficial de CodeIgniter, donde puede descargarse el framework, es:

http://ellislab.com/codeigniter

Requisitos para trabajar con CodeIgniter
Necesitamos disponer de PHP 5 para trabajar con CodeIgniter 2.x.x.

En lo que respecta al trabajo con bases de datos, CodeIgniter es compatible con varios motores, entre ellos MySQL (4.1 o posterior), MySQLi, MS SQL, PostgreSQL, Oracle, SQLite, y acceso a cualquier base de datos en entornos Windows por ODBC.

Instalación de CodeIgniter
- Descomprimir el paquete en la carpeta del sitio o del servidor (con lo que estará disponible para las aplicaciones que allí creemos)

- Configurar la URL base de la aplicación web

Es necesario indicarle a CodeIgniter la URL base de la aplicación, es decir, la URL para acceder a la raíz de CodeIgniter, según en el servidor y directorio donde se ha puesto el código del framework. Para ello abrimos el archivo de configuración, que se encuentra en ***application/config/config.php***, con cualquier editor de texto y cambiar la variable de configuración llamada que se guarda en ***$config['base_url']***.

Si la instalación es local, puede ponerse algo así: http://localhost/ y si está en un directorio específico, podría ser algo como http://localhost/directorio_codeigniter

Si hemos instalado el framework en un dominio de Internet podremos indicar algo como http://midominio.co/ y si creamos una carpeta para subir CodeIgniter en ella pondremos el nombre del dominio y luego el nombre de la carpeta o carpetas, separadas por barras y acabando siempre en una barra: http://midominio.co/carpeta/otracarpeta/

Según nos indican en el manual de instalación, aparte de este dato podemos opcionalmente escribir una llave de encriptación en la variable $config['encryption_key'], que servirá si deseamos usar la clase de encriptado que proporciona CodeIgniter o queremos que nuestras variables de sesión estén encriptadas, algo que hace el framework de manera transparente para nosotros.

- Configurar la base de datos
En este último paso hay que indicar los datos de acceso a la base de datos que se utilizará con CodeIgniter, ya que prácticamente todas las aplicaciones web que pueden crearse con el framework van a tener que utilizar la base de datos para algo. Para ello tenemos que editar el archivo ***application/config/database.php*** e indicar los parámetros de conexión al servidor de base de datos, como el nombre del servidor, el nombre de la base de datos, el motor (driver), el usuario y la contraseña entre otros.

Con esto ya se tiene listo el entorno de trabajo para comenzar a crear aplicaciones web PHP con el framework. Para verificar que CodeIgniter está funcionando, basta con acceder a la URL donde se instaló. Se muestra el mensaje de bienvenida de CodeIgniter, lo cual indica que está funcionando.

CodeIgniter define una serie de reglas de trabajo que son necesarias conocer antes de escribir líneas de código. Entendiendo la metodología de trabajo del framework, el desarrollo de aplicaciones se facilitará bastante y se comprenderá mejor las ventajas que ofrece.

Flujo de aplicación de CodeIgniter
En CodeIgniter existe un procedimiento para atender una solicitud de página del cliente. Este proceso se realiza internamente por el propio CodeIgniter y de manera transparente para el desarrollador. Durante el proceso participan varios módulos como el enrutamiento de la solicitud, la caché interna, etc., que ha medida que se avanza en su estudio se comprende mejor cual es su función.

La siguiente imagen es tomada de la documentación de CodeIgniter, donde se han traducido algunos nombres de los módulos que interactúan en el framework.

En resumen, para que se pueda entender el flujo de aplicación que implementa CodeIgniter, se pueden seguir los siguientes puntos:

1. Toda solicitud de una página a partir de CodeIgniter comienza en un index.php que hay en la raíz del framework.
2. Luego se realiza un filtrado de la URL para saber cuál es elemento que tiene que procesar esta página.
3. Si la página se había generado antes y está en la caché de CodeIgniter, se devuelve el archivo de la caché ya generado, con lo que se ahorra procesamientos repetidos. La caché se puede configurar y si lo deseamos, incluso deshabilitar.
4. Antes de continuar con el proceso se realiza un tratamiento de seguridad sobre la entrada que tengamos, tanto de la información que haya en la URL como de la información que haya en un posible POST, si lo hemos configurado así.
5. El controlador adecuado realiza el procesamiento de la solicitud. CodeIgniter decide el controlador que debe procesar la solicitud en función de la URL solicitada.
6. El controlador comunica con una serie de módulos, los que necesite, para producir la página.
7. A través de las vistas adecuadas, el controlador genera la página, tal cual se tiene que enviar al navegador.
8. Si la página no estaba en la caché, se introduce, para que las futuras solicitudes de esta página sean más rápidas.

Algunos de estos módulos, como la caché o el enrutamiento, funcionan de manera transparente para nosotros. Algunos otros, como los controladores, modelos y vistas, los tenemos que programar por nuestra cuenta y localizan cada una de las partes de nuestro programa que, al estar sepadadas nos ayudan a organizar también nuestro código. También tenemos a nuestra disposición diversas librerías, ayudantes (helpers) y plugins ya escritos en CodeIgniter con numerosas clases y funciones muy útiles para el desarrollo de aplicaciones web.

El múdulo de enrutamiento (routing) permite que cualquier URL que se solicite al servidor se ejecute en el controlador adecuado. La URL se analiza y los datos se procesan y aseguran antes de enviarlos al controlador adecuado, en el que simplemente tendremos que codificar sus diversos métodos.

MVM Modelo - Vista - Controlador (Model - View - Controller)
El Modelo, Vista, Controlador es típicamente utilizado para la creación de aplicaciones web y no sólo CodeIgniter lo implementa, sino también otra serie de frameworks de desarrollo web, en PHP u otros lenguajes. Es interesante porque separa en varios grupos las complejidades de las distintas partes que componen una página web, como la vista y la lógica, así como el acceso a la base de datos.

El Modelo - Vista - Controlador es un patrón de desarrollo o un estilo de arquitectura de software que separa el código fuente de las aplicaciones en tres grupos:

Modelo
Todo el código que tiene que ver con el acceso a la base de datos. En el modelo mantendremos encapsulada la complejidad de nuestra base de datos y simplemente crearemos funciones para recibir, insertar, actualizar o borrar información de nuestras tablas. Al mantenerse todas las llamadas a la base de datos en un mismo código, desde otras partes del programa podremos invocar las funciones que necesitemos del modelo y éste se encargará de procesarlas. En el modelo nos podrán preocupar cosas como el tipo de base de datos con la que trabajamos, o las tablas y sus relaciones, pero desde las otras partes del programa simplemente llamaremos a las funciones del modelo sin importarnos qué tiene que hacer éste para conseguir realizar las acciones invocadas.

Vista
La vista codifica y mantiene la presentación final de nuestra aplicación de cara al usuario. Es decir, en la vista colocaremos todo el código HTML, CSS, Javascript, etc. que se tiene que generar para producir la página tal cual queremos que la vea el usuario. En la práctica la vista no sólo sirve para producir páginas web, sino también cualquier otra salida que queramos enviar al usuario, en formatos o lenguajes distintos, como pueden ser XML, feeds RSS, archivos JSON, etc.

Controlador
El controlador podríamos decir que es la parte más importante, porque hace de enlace entre el modelo, la vista y cualquier otro recurso que se tenga que procesar en el servidor para generar la página web. En resumen, en el controlador guardamos la lógica de nuestras páginas y realizamos todas las acciones que sean necesarias para generarlas, ayudados del modelo o la vista.

Nota
Esto quiere decir, en la práctica para el caso de CodeIgniter, que según sea el ámbito donde estemos codificando, tendremos que escribir las líneas de código de cualquier página web en tres grupos de archivos distintos. En una aplicación estándar podremos

tener varios modelos (por ejemplo, para cada una de las entidades distintas de la base de datos), varias vistas (una o varias para cada página o sección) y varios controladores (para cada página o sección de la web). Luego veremos dónde tenemos que guardar los archivos de cada uno de estos tres grupos.

Durante el desarrollo con CodeIgniter será muy recomendable seguir las normas del diseño Modelo - Vista - Controlador (MVC), pero realmente el framework es bastante flexible y permite que, si lo deseamos, no sigamos el desarrollo atendiendo a dicha arquitectura. En este caso, podríamos tener simplemente controladores y dentro de ellos realizar todas las acciones de acceso a la base de datos directamente, sin hacer llamadas al modelo, o escribir texto en la salida sin utilizar las vistas. Obviamente, esta opción no aprovechará las ventajas de separación de código entre presentación, lógica y modelo de base de datos.

En el caso que no utilizemos los modelos, no ocurrirá ningún efecto negativo en el desempeño del framework, pero en el caso de las vistas, si no las utilizamos y escribimos salida directamente desde el controlador, como por ejemplo con sentencias echo de PHP en el código de los controladores, perderemos algunas de las ventajas que CodeIgniter realiza por nosotros para procesar la salida antes de enviarla al usuario.

Nota
Separar la vista o presentación de la lógica de los programas es una ventaja importante, ya que ayuda a mantener un código más sencillo, reducido y con mayor facilidad de mantenimiento. En principio puede parecer un tanto complejo el hecho de manejar varios archivos con códigos distintos, pero a medio plazo nos vendrá muy bien separar el código de cada parte de nuestra aplicación. Nos evitará muchos de los problemas que a veces tenemos en desarrollos PHP, donde en un mismo archivo tenemos mezclado código en varios lenguajes como HTML, CSS, SQL, con el propio código PHP para generar la lógica de nuestro negocio.

URLs amigables a buscadores
Uno de los puntos que debemos conocer antes de empezar a trabajar con CodeIgniter es sobre cómo son las direcciones URL que utiliza este popular framework PHP para cada una de las páginas de las aplicaciones PHP que creemos utilizándolo. La verdad es que es un punto que realmente resulta transparente para nosotros, puesto que las URL se generan automáticamente a medida que vamos programando el sitio web, por lo que vamos a comentar algunas aspectos sobre esto.

Uno de los puntos fuertes de este framework PHP es que las URL se presentan siempre en un formato amigable a los buscadores. Esto significa que cualquier motor de búsqueda puntuará positivamente, a priori, las direcciones de las páginas. Del mismo modo, las direcciones tendrán una forma fácil de entender y recordar por los seres humanos.

Por ejemplo, las siguientes URL's no suelen ser puntuadas bien por los buscadores:

http://www.midominio.com/ejemplo.php?ced=37
http://www.midominio.com/ejemplo.php?nombre=Evaristo

Si lo vemos, tenemos una página, ejemplo.php, que se le pasan distintos parámetros, pero los buscadores muchas veces interpretan que es la misma página. Con CodeIgniter las URL tienen mejor arquitectura, con formas como estas:

http://www.midominio.com/ejemplos/ejemplo/37

www.midominio.com/controlador/funcion/parametro

Las diferencias saltan a la vista, tanto para nosotros humanos como para motores de búsqueda como Google u otro. Y lo bueno es que nosotros no tenemos que hacer nada para conseguir este tipo de direcciones.

Nota
Las URL en CodeIgniter tienen el nombre de una página llamada index.php, pero esto es algo que podemos hacer desaparecer si sabemos configurar el framework.

Query String desactivado
En CodeIgniter en principio está desactivada la posibilidad de envío de variables a través de la URL, lo que se conoce en inglés como Query String. Es decir, que direcciones en las que se envían variables a través de las URL, que decíamos que eran poco amigables a buscadores, no funcionarán.

Si se desea, se puede hacer que CodeIgniter reconozca las variables enviadas por la URL, pero como en principio el sistema de URLs amigables a buscadores que implementa el framework está pensado para poder evitar el problemático Query String, su uso está desactivado.

Segmentos de la URL y el Modelo - Vista - Controlador
Cada una de las partes de la URL de las aplicaciones creadas con el framework sirve para identificar qué controlador, del ya explicado Modelo - Vista - Controlador de CodeIgniter, se va a hacer cargo del procesamiento de la página, así como de la función que se invocará y los parámetros que se le enviarán a la misma. Por ejemplo:

aplicacioncodeiginter.com/facturacion/editarempresa/5610

aplicacioncodeiginter.com es el nombre del supuesto dominio donde tenemos CodeIgniter instalado.
facturacion es el nombre del controlador que se encargará de procesar la solicitud.

editarempresa es el nombre de la función que habrá dentro del controlador y donde estará el código que ejecute y genere la página. (estrictamente hablando, en la terminología de la programación orientada a objetos POO, en vez de función, deberíamos llamarle método o función miembro).

Por último, 5610, es el parámetro que se le pasa a la función editarempresa, que servirá en este caso para que editarempresa sepa cuál es la empresa que se desea editar. Si queremos o necesitamos enviar varios parámetros a esta función, y no sólo el identificador de la empresa a editar, podremos colocarlos a continuación, separados por barras.

Nota
CodeIgniter pone a nuestra disposición una clase para trabajar con URLs llamada URI Class y una librería llamada URL Helper que contienen funciones para trabajar fácilmente con URLs y datos enviados en las mismas. En estas librerías hay funciones tan interesantes como site_url() que sirve para que el propio CodeIgniter cree una URL dentro del sitio a partir de un parámetro que le pasemos. Otro ejemplo es base_url(), que simplemente devuelve la URL raíz donde está nuestra aplicación CodeIgniter.

Todo pasa por index.php
En CodeIgniter existe un index.php que está en la raíz del framework que se encarga de las funciones de enrutamiento hacia el controlador que se debe encargar de procesar la solicitud. Por ello, de manera predeterminada en CodeIgniter veremos que las URLs incluyen el nombre del archivo index.php. Este comportamiento se puede configurar.

Más adelante veremos cómo eliminar este index.php en las URLs de CodeIgniter, algo que simplificará las direcciones.

Añadir un sufijo a las URL
Otro de los detalles que podemos hacer con CodeIgniter, que pueden personalizar aun más nuestras direcciones URL, es añadir un sufijo, que nosotros deseemos, al final de todas las URL que formen parte del framework. Por ejemplo, podríamos desear que todas las URL acaben en .html o en .php, o como queramos. Esto se puede hacer a través de los archivos de configuración del framework.

La idea es que una URL como esta:

http://dom.com/index.php/blog/post/cualquier-articulo

Pase a ser una dirección como esta otra:

http://dom.com/index.php/blog/post/cualquier-articulo.html

Para esto editamos el archivo de configuraciones generales:

application/config/config.php y editamos la variable url_suffix y colocar el valor que deseemos, por ejemplo:

$config['url_suffix'] = ".html";

CLASE No. 15 GESTIÓN AVANZADADE INFORMACIÓN CON EL MOTOR MYSQL

EL MOTOR MYSQL
En los últimos años ha tenido grandes avances en su desarrollo y actualmente es uno de los DBMS más usados en sitios web y que en sus últimas versiones ha incorporado nuevas características como la implementación de las tablas InnoDB, disparadores, replicación, funciones y procedimientos almacenados.
Uso de los procedimientos almacenados

Procedimientos almacenados
Los procedimientos almacenados son de gran utilidad para realizar tareas frecuentes en una base de datos, es sorprendente la cantidad de tiempo que se puede llegar a ahorrar al hacer uso de este mecanismo. Aunque es de anotar que en ciertas ocasiones pueden volver algunos procesos lentos, por lo que es necesario revisar cuando aplicarlos. Veamos algunos ejemplos de implementación de procedimientos almacenados en MySQL y como llamarlos desde una página PHP.

La sintaxis general para crear un procedimiento almacenado es:
CREATE PROCEDURE <*nombre_del_procedimiento* ([*parámetros*[,...]])>
[características ...] cuerpo

la sintaxis para especificar los parámetros es:
[**IN | OUT | INOUT**] *nombre_del_parametro* **tipo_de_dato**

IN, indica que el parámetro es sólo de entrada, **OUT** que es una variable de salida y **INOUT** que es de ambas formas, si no se especifica el modo del parámetro por defecto es de tipo IN.

Los procedimientos almacenados se crean como consultas SQL y pueden ejecutarse directamente

Ejemplos de aplicación de procedimientos almacenados
Veremos varias formas de crear procedimientos almacenados en MySQL y llamarlos desde PHP

ejemplo1
Escribimos las siguientes sentencias SQL

```
CREATE PROCEDURE ejemplo1()
```

```
  INSERT INTO comentarios(comentario,idpersona) VALUES("Hola a todos", 1), ("Hoy es miércoles", 1);
call ejemplo1();
```

ejemplo2

```
delimiter //
CREATE PROCEDURE ejemplo2(in id int, out idc int)
 begin
   INSERT INTO comentarios(comentario,idpersona) VALUES("Hola a todos 2", id), ("Hoy es miércoles 2", id);
   select count(idcoment) into idc from comentarios where idpersona=id;
 end

//
delimiter ;
call ejemplo2(1,@id);
select @id;
```

ejemplo3

```
delimiter //
CREATE PROCEDURE ejemplo3(in coment varchar(50), in id int, out idc int)
 begin
   INSERT INTO comentarios(comentario,idpersona) VALUES(coment, id);
   select count(idcoment) into idc from comentarios where idpersona=id;
 end
//
delimiter ;
call ejemplo3("Esto va por buen camino",1,@id);
select @id;
```

ejemplo4

```
delimiter //
CREATE PROCEDURE ejemplo4(in id int, out idc int, out idc2 int)
 begin
   select count(idcoment) into idc from comentarios;
   delete from comentarios where idcoment=id;
   select count(idcoment) into idc2 from comentarios;
 end
//
delimiter ;
call ejemplo14(9,@id, @id2);
select @id  as antiguo, @id2 as nuevo;
```

ejemplo5

```
delimiter //
CREATE PROCEDURE ejemplo5 (IN per int)
    BEGIN
         DECLARE variable1 CHAR(10);
         IF per = 1 THEN
               SET variable1 = 'F';
         ELSE
               SET variable1 = 'M';
         END IF;
         INSERT INTO comentarios(comentario,idpersona) VALUES (variable1, per);
         select * from comentarios;
    END
//
delimiter ;
call ejemplo5(1);
```

ejemplo6

```
delimiter //
CREATE PROCEDURE ejemplo6 (in nom varchar(50), in ed int, in coment varchar(50), in
n int)
    BEGIN
        DECLARE t int default 0;
        INSERT INTO persona(nombre,edad) VALUES (nom, ed);
        set t=0;

          WHILE t < n do
             INSERT INTO comentarios(comentario,idpersona) VALUES (coment, 1);
             set t=t+1;
          END WHILE;
    END

//
delimiter ;

call ejemplo6("Maria", 35, "A", 5);
```

ejemplo7

```
delimiter //
CREATE PROCEDURE ejemplo7 (in nom varchar(50), in ed int)

    BEGIN
      INSERT INTO persona(nombre,edad) VALUES (nom, ed);
    END
```

```
//
delimiter ;

call ejemplo18("Jairo", 30);
```

Ejemplo
Código PHP para guardar un registro utilizando procedimientos almacenados

```
<?php
include("conexion.php");
$link=conectar();
$nom="Andrea";
$ed=16;
 if(mysql_query("call ejemplo7('$nom',$ed)",$link))
   echo "Registro guardado";
 else
   echo "No se pudo guardar el registro";
?>
```

CLASE No. 16 AJAX Y XML EN SITIOS WEB

AJAX
Es el acrónimo de *Asynchronous JavaScript And XML*, una tecnología que utiliza Javascript y XML para trabajar de forma asincrónica con el servidor.

AJAX és una técnica de desarrollo web para crear aplicaciones interactivas usando diferentes tecnologías web que colaboran entre ellas.

Nacido en 2005 cómo una técnica para la sustitución del iframe como método de comunicación sin refresco se ha convertido en pocos años en una tecnología imprescindible en la mayoría de aplicaciones web.

AJaX (Asynchronous JavaScript and XML) és una técnica de desarrollo web para crear aplicaciones interactivas mediante la combinación de cinco tecnologías.

JavaScript: Para manejar el objeto ***XMLHttpRequest*** y el ***DOM*** para tratar para los datos recibidos.
HTML: Distribuye en la ventana del navegador los elementos de la aplicación y la información recibida por el servidor.
CSS: Define el aspecto de cada elemento y dato de la aplicación.
XML: Es el formato de los datos transmitidos del servidor al cliente (navegador) y que posteriormente serán mostrados.
Lenguaje de servidor: Genera la información útil en XML y la envía al navegador.

Funcionamiento de AJaX
El usuario accede a la aplicación que es enviada por el servidor en formato HTML, JavaScript y CSS. Luego el código JavaScript de la aplicación pide al servidor los datos que quiere mostrar y éste ejecuta un código de lado de servidor que envía al navegador los datos en formato XML.

Problemas
El principal problema de la gran mayoría de aplicaciones AJaX es la baja compatibilidad entre navegadores, puesto que la capa JavaScript es de una gran complejidad y a menudo por falta de experiencia en el lenguaje, o por falta de tiempo, se opta por programar solo para Internet Explorer.

Objeto XMLHttpRequest
Un objeto ***XMLHttpRequest*** es una instancia de una API que nos permite la transferencia de datos en formato XML desde los scripts del navegador (JavaScript, JScript, VBScript, etc.) a los del servidor (PHP, Perl, ASP, Java, etc.) y viceversa.

Ejemplos

1. Aquí vamos a mostrar el contenido de un archivo .txt mediante Ajax

```
<html>
<head>
<title>Ejemplo 1 Ajax</title>
<script>
function leerDatos()
 {
  if (oXML.readyState == 4)
   {
     alert ("Ajax en acción \n"+oXML.responseText);

   }
 }

function AJAXCrearObjeto()
 {
  var obj;
   if(window.XMLHttpRequest)
    { // no es IE
     obj = new XMLHttpRequest();
    }
   else
    { // Es IE o no tiene el objeto
     try
           {
      obj = new ActiveXObject("Microsoft.XMLHTTP");
     }
     catch (e)
           {
      alert('El navegador utilizado no está soportado');
     }
    }
  return obj;
 }

oXML = AJAXCrearObjeto();
oXML.open('GET', 'archivo.txt');
oXML.onreadystatechange = leerDatos;
oXML.send('');
```

```
</script>
</head>
</html>
```

2. Este ejemplo muestra datos de un archivo xml

```
<html>
<head>
<title>Ejemplo 2 Ajax</title>
<script>
function leerDatos()
{
  if (oXML.readyState == 4)
  {
        var xml = oXML.responseXML.documentElement;
        //alert(xml.getElementsByTagName('texto').length);
        //alert(xml.getElementsByTagName('mensaje').length);
        for (i = 0; i < xml.getElementsByTagName('mensaje').length; i++)
        {
          var item = xml.getElementsByTagName('mensaje')[i];
                //for (j = 0; j < xml.getElementsByTagName('texto').length; j++)
                //{
                        var txt = item.getElementsByTagName('texto')[0].firstChild.data;
                        alert(txt);
                //}
        }
  }
}
function AJAXCrearObjeto(){
var obj;
if(window.XMLHttpRequest) { // no es IE
obj = new XMLHttpRequest();
} else { // Es IE o no tiene el objeto
try {
obj = new ActiveXObject("Microsoft.XMLHTTP");
}
catch (e) {
alert('El navegador utilizado no está soportado');
}
}
return obj;
}

oXML = AJAXCrearObjeto();
oXML.open('get', 'archivo.xml');
```

```
oXML.onreadystatechange = leerDatos;
oXML.send('');
</script>
</head>
</html>
```

El archivo xml es el siguiente

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
 <mensaje>
 <texto>Ejemplo 1.1</texto>
 </mensaje>
 <mensaje>
 <texto>Ejemplo 2.1</texto>
 </mensaje>
</xml>
```

3. Este ejemplo muestra datos de un archivo xml

```
<html>
<head>
  <title>Ejemplo 3 Ajax</title>
  <script>
  // Recibe y muestra los datos
  function leerDatos(){
    // Comprobamos que se han recibido los datos
    if (oXML.readyState == 4) {
      // Accedemos al XML recibido
      var xml  = oXML.responseXML.documentElement;
      // Accedemos al DIV
      var miDiv = document.getElementById('miDiv1');
      // Vaciamos el DIV
      miDiv.innerHTML = '';
      // Iteramos cada usuario
      for (i = 0; i < xml.getElementsByTagName('usuario').length; i++){
        // Accedemos al objeto XML usuario
        var item = xml.getElementsByTagName('usuario')[i];
        // Recojemos el id
        var id = item.getElementsByTagName('id')[0].firstChild.data;
        // Recojemos el nombre
        var nombre = item.getElementsByTagName('nombre')[0].firstChild.data;
        // Mostramos el enlace
        miDiv.innerHTML += '<a href="/perfil/'+id+'/">'+nombre+'</a><br>';
```

```
    }
   }
  }

  // Crea el objeto AJAX
  function AJAXCrearObjeto(){
    var obj;
    if(window.XMLHttpRequest) { // no es IE
      obj = new XMLHttpRequest();
    } else { // Es IE o no tiene el objeto
      try {
        obj = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e) {
        alert('El navegador utilizado no está soportado');
      }
    }
    return obj;
  }

  oXML = AJAXCrearObjeto();
  oXML.open('get', 'usuarios.xml');
  oXML.onreadystatechange = leerDatos;
  oXML.send('');
  </script>
</head>
<body>
  <div id="miDiv1">Aquí aparecerán los datos</div>
   <br>

</body>
</html>
```

El respectivo archivo xml es:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
 <usuario>
  <id>1</id>
  <nombre>Ana</nombre>
 </usuario>
 <usuario>
  <id>2</id>
  <nombre>Pedro</nombre>
```

```
</usuario>
<usuario>
 <id>3</id>
 <nombre>Juan</nombre>
</usuario>
</xml>
```

Ejemplo. Aplicación AJaX PHP

Este ejemplo utiliza Ajax en una aplicación que gestiona bases de datos con PHP

consulta_persona.html

```
<html>
   <head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   <title>Registros con AJAX</title>
<script language="JavaScript" type="text/javascript"src="../jquery-1.7.2.js" ></script>
   <script language="JavaScript" type="text/javascript" src="crearobjetoajax.js"></script>
   <script language="JavaScript" type="text/javascript" src="consultaajax.js"></script>
   <script language="JavaScript" type="text/javascript" src="insertaajax.js"></script>
   <script language="JavaScript" type="text/javascript" src="insertaajaxjquery.js"></script>
   <script language="JavaScript" type="text/javascript" src="eliminarajax.js"></script>
   <script language="JavaScript" type="text/javascript" src="limpiarelementos.js"></script>
      </head>
   <body>
   <p>Consultar registros con ajax</p>
     <form name="form1" id="form1">
  Nombre <input type="text" name="txtnombre" id="txtnombre"/>
  <br/>
  Edad <input type="text" name="txtedad" id="txtedad"/>
  <br/>
  <input type="button" value="Consultar" onclick="MostrarConsulta('consulta.php');"/>
  <input type="button" value="Guardar"
onclick="GuardarDatos('insertar2.php');MostrarConsulta('consulta.php');"/>
  <input type="button" value="Limpiar" onclick="limpiar();"/>
  <br/>
  Con enlaces
  <br/>
  <a href="#" onclick="MostrarConsulta('consulta.php');">Consultar</a>
  <a href="#" id="enlaceajax">Guardar</a>
  <a href="#" onclick="limpiar();">Limpiar</a>
 </form>
```

```
  <div id="resultado" style="text-align:center;background-color:orange;
width:800px;"></div>
  <div id="destino"></div>
  </body>
</html>
```

consulta.php

```
<?php
//Configuracion de la conexion a base de datos
include("conexion.php");
$link=conectar();
$sql=mysql_query("SELECT * FROM persona",$link);
$nreg=mysql_num_rows($sql);
//muestra los datos consultados
echo "<table>
      <tr>
         <th>Id</th>
         <th>Nombre</th>
         <th>Edad</th>
         <th>Fecha y hora de registro</th>
         <th>Eliminar</th>
      </tr>";

 while($row = mysql_fetch_array($sql))
  {
   $id=$row['id'];
   echo "<tr>
         <td>" . $row['id'] . "</td>
         <td>" . $row['nombre'] . "</td>
         <td>" . $row['edad'] . "</td>
         <td>" . $row['fechahora_registro'] . "</td>
         <td>
          <a href='#'
onclick='EliminarRegistro(\"eliminar.php?id=$id\");MostrarConsulta(\"consulta.php\");'>
       <img src='imagenes/icono_eliminar.gif' border='0' title='Eliminar'/>
          </a>
         </td>
        </tr>";
  }
  echo "<caption align='bottom'>
         Total registros
         <span style='color:white;'>$nreg </span>
      </caption>
      </table>";
 mysql_free_result($sql);
```

```
 mysql_close($link);

?>
```

insertar.php

```
<?php
//Configuracion de la conexion a base de datos
include("conexion.php");
$link=conectar();
  if(isset($_POST["nombre"]) && isset($_POST["edad"]))
  {
   $nom=$_POST['nombre'];
   $ed=$_POST['edad'];
    if($nom!="" && $ed>=0)
       mysql_query("INSERT INTO persona(nombre, edad) VALUES('$nom', '$ed')",$link);
    else
      echo "Suministre datos";
  }
 else
   echo "No hay datos";

mysql_close($link);
?>
```

eliminar.php

```
<?php
//Configuracion de la conexion a base de datos
include("conexion.php");
$link=conectar();
$id=$_GET['id'];
 mysql_query("DELETE FROM persona WHERE id='$id'",$link);
mysql_close($link);
?>
```

conexion.php

```
<?php
function conectar()
  {
    if(!($link=mysql_connect("localhost","root","")))
      {
       echo "Error no pude conectrse al servidor";
            exit();
           }
    if(!mysql_select_db("bdpersona",$link))
      {
```

```
            echo "Error no pude conectarse a la base de datos";
            exit();
           }
      return $link;
  }
?>
```

insertaajax.js

```
function GuardarDatos(pagina)
  {
    nombre = document.getElementById('txtnombre').value;
    edad = document.getElementById('txtedad').value;
    ajax=AJAXCrearObjeto();
    ajax.open("GET", pagina+"?nombre="+nombre+"&edad="+edad,true);
    //alert("Registro guardado");
    ajax.send(null);
  }
```

eliminarajax.js

```
function EliminarRegistro(datos)
  {
   ajax=AJAXCrearObjeto();
   ajax.open("GET", datos);
   ajax.send(null);
  }
```

consultaajax.js

```
function MostrarConsulta(pagina){
       divResultado = document.getElementById('resultado');
       ajax=AJAXCrearObjeto();
       ajax.open("GET", pagina);
       ajax.onreadystatechange=function() {
              if (ajax.readyState==4)
               {
                 divResultado.innerHTML = ajax.responseText;
               }
         }
       ajax.send(null);
}
```

crearobjetoajax.js

```
function AJAXCrearObjeto()
 {
  var obj;
   if(window.XMLHttpRequest)
```

```
   { // no es IE
    obj = new XMLHttpRequest();
   }
  else
   { // Es IE o no tiene el objeto
    try
     {
      obj = new ActiveXObject("Microsoft.XMLHTTP");
     }
    catch (e)
     {
      alert('El navegador utilizado no está soportado');
     }
   }
 return obj;
}
```

CLASE No. 17 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

XML (EXTENSIBLE MARKUP LANGUAGE)

Muchas empresas han empezado a utilizar XML como una forma de intercambiar datos. Los gobiernos de algunos países han elegido XML como el formato estándar de intercambio de datos, y ya se está exigiendo a los desarrolladores Web que aprendan a manejar XML. Por tanto, puede resultar interesante aprender sobre XML y ampliar su rango de conocimientos.

¿Qué es XML?

XML es un acrónimo cuyo significado en inglés es *E**X**tensible **M**arkup **L**anguage* (Lenguaje de Marcado Extensible). Es decir, es un lenguaje de marcado, como el lenguaje HTML, lo cual significa que utiliza etiquetas. Pero, ¿qué es lo que hace realmente? ¿Y por qué sus clientes piden XML? Sorprendentemente, XML no hace nada. Simplemente describe información y la distribuye en un formato independiente de la plataforma.

XML es un sistema independiente de la plataforma porque no usa un lenguaje específico. Las etiquetas de XML no están predefinidas, lo cual significa que cada uno escribe sus propias etiquetas. La ventaja de esto es que XML no precisa ninguna explicación adicional.

Aunque no lo crea, usted ha estado usando XML desde hace tiempo. Cuando lee titulares de noticias directamente en su programa de correo electrónico, o cuando visita páginas Web desde su teléfono móvil, está utilizando tecnología basada en XML.

Diferencias entre XML y HTML

En primer lugar, XML no es un sustituto de HTML y su objetivo es totalmente distinto. XML fue diseñado para describir, almacenar e intercambiar datos, mientras que HTML fue diseñado para presentar datos en un formato legible para las personas. HTML utiliza un conjunto predefinido de elementos (llamados etiquetas y atributos) para definir aspectos visuales de un documento, como el diseño de la página o el formato del texto, y para incluir vínculos a documentos o imágenes. En HTML, está limitado a usar el conjunto de etiquetas de HTML. Por tanto, el tipo de información que puede mostrar es limitado; por ejemplo, mostrar una fórmula matemática con HTML puede ser muy complicado. XML resuelve este tipo de problemas mediante la extensibilidad: puede "inventar" sus propias etiquetas y su propia estructura del documento; se pueden añadir o eliminar elementos sin que esto afecte a la estructura global del documento.

El siguiente texto es un ejemplo de una descripción XML de un departamento de una empresa:

Ejemplo

```
<?xml version="1.0" encoding="iso-8859-1"?>
<departmento>
  <empleado>
          <nombre>María Duque</nombre>
          < trabajo >Desarrollador de Software </trabajo>
          <salario>800000</salario>
  </empleado>
  <empleado>
          <nombre>Juan Medina</nombre>
          < trabajo >Diseñador </trabajo>
          <salario>900000</salario>
  </empleado>
</departmento>
```

Observe la primera instrucción de este archivo XML que contiene una línea similar a la siguiente:

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

La cual especifica la declaración de un documento XML con una codificación de caracteres específica.

Guarde este primer documento XML y obtenga una vista previa en el navegador. Podrá observar que el navegador, al igual que el editor, también colorea el código y muestra el documento XML como un árbol que puede contraerse (si el navegador admite XML).

Nota

La mayoría de los navegadores admiten XML de forma predeterminada.

Si hace clic en el signo menos (**-**) al lado de cada etiqueta podrá contraer el elemento. Para expandir un elemento, haga clic en el signo más (**+**) situado junto al elemento.

El mismo ejemplo se podría escribir en HTML de la siguiente forma:

```
<table>
        <tr>
                <td> María Duque </td>
                <td> Desarrollador de Software </td>
                <td>800000</td>
        </tr>
        <tr>
                <td> Juan Medina </td>
                <td> Diseñador </td>
                <td>900000</td>
        </tr>
</table>
```

Si carga la página en el navegador, ésta tendrá el aspecto de una tabla HTML clásica

Las etiquetas en el ejemplo anterior fueron diseñadas específicamente para describir la información relacionada con los empleados de una empresa. Si compara los dos ejemplos, apreciará que XML se basa en el contenido, mientras que HTML se basa en el formato: los nombres de las etiquetas XML describen los datos en las propias etiquetas, mientras que las etiquetas HTML describen la presentación de los datos en las etiquetas.
El objetivo del ejemplo anterior es ilustrar algunas diferencias importantes entre XML y HTML. Sin embargo, debería tener en cuenta que XML no ha sido diseñado como un sustituto de HTML, y que no todos los documentos XML pueden convertirse en un documento HTML.

CLASE No. 18 SCRIPTS AVANZADOS DEL LADO DEL SERVIDOR

¿Qué es XHTML?
Llegados a este punto, las cosas pueden parecer un poco confusas con tanto acrónimo. Aunque XHTML está fuera del propósito de este texto, explicaremos brevemente en qué consiste.

XHTML son las siglas de Extensible HyperText Markup Language (Lenguaje de Marcado de Hipertexto Extensible). XHTML es una versión más limpia de HTML basada en XML. Es una especificación más estricta de HTML, diseñada y pensada para sustituir en un

futuro a HTML[7]. XHTML contiene prácticamente las mismas etiquetas que HTML, pero éstas se reescribieron para cumplir las reglas de sintaxis XML.

XML almacena datos

Como XML almacena y describe datos, se asemeja a una base de datos. Puede que haya oído hablar de las palabras "esquema" o "lenguaje de consultas" citadas en relación con XML. Entonces, ¿en qué se diferencia XML de un sistema de administración de bases de datos (DBMS) típico?

En primer lugar, XML es compatible con múltiples plataformas. A diferencia de las bases de datos, que se basan en un lenguaje específico, XML incluye el significado en sus propias etiquetas. XML no precisa ninguna explicación adicional, lo cual significa que contiene tanto la estructura como la semántica de los datos, mientras que las bases de datos sólo pueden definir la estructura de los datos.

Además, XML puede representar datos mediante árboles jerárquicos, como ya hemos visto anteriormente. Por ejemplo, el elemento <empleado> es un elemento hijo del elemento <departmento> y se incluye como tal en el árbol XML.

Por supuesto, XML también tiene ciertos inconvenientes comparado con las bases de datos, lo cual es normal, puesto que éstas fueron diseñadas con otras finalidades. El inconveniente más obvio de XML es que carece de funciones específicas de bases de datos, como desencadenadores, acceso multiusuario, almacenamiento eficiente, índices, seguridad, transacciones, comprobaciones de integridad de datos o consultas en múltiples documentos. Esto es normal, teniendo en cuenta que un DBMS está diseñado para manipular, almacenar y recuperar datos de forma rápida y segura, mientras que XML está diseñado para intercambiar datos entre plataformas.

Como consecuencia de lo anterior, la búsqueda en documentos XML también es más lenta, debido a la falta de índices y de funciones de optimización de búsquedas, las cuales sí están disponibles en las bases de datos.

Además, XML es más detallado, ya que necesita un par de etiquetas y/o atributos para cada elemento de datos.

A pesar de que la división en bases de datos centradas en los datos y bases de datos centradas en el documento está un poco obsoleta, esta división resulta útil para entender la filosofía de XML y los conceptos más importantes de la tecnología XML.

A continuación se muestran algunas de las ventajas e inconvenientes de XML y se ilustran algunas situaciones de la vida real donde le resultaría más provechoso utilizar XML.

[7] Aunque actualmente los esfuerzos del W3C están enfocados en el nuevo estándar, aun en versión de prueba, de HTML 5

¿Por qué utilizar XML?
Para averiguar por qué debería usar XML, es preciso que conozca sus principales ventajas. Ya se han mencionado algunas anteriormente. Veamos las ventajas que ofrece XML frente a otros lenguajes de formato o tecnologías similares.

La X significa "ampliable" (en inglés, "eXtensible")
Cuando se trabaja con XML, con frecuencia se olvida que la X significa "ampliable". Ser ampliable significa que puede definir cualquier conjunto de etiquetas adicionales sin que se bloquee la aplicación. Por ejemplo, puede añadir hijos a este elemento:

```
<empleado>
        <nombre>María Duque</nombre>
        <trabajo>Desarrollador de software</ trabajo >
        <salario>2000</salario>
</ empleado >
```

Para que tenga este nuevo aspecto:

```
<empleado>
        <nombre> María Duque </nombre>
        <trabajo> Desarrollador de software
                <responsabilidad>Escribir especificaciones técnicas</responsibilidad>
                <responsibility>Testear software</responsibility>
        </trabajo>
        <fecha_ingreso>Jun 25, 2005</fecha_ingreso>
        <salario>800000</salario>
</empleado>
```

La aplicación que lea este documento XML todavía podrá saber de qué empleado se trata.

Plataformas distintas, un único lenguaje
La portabilidad de XML es consecuencia de que es el propio desarrollador el que define las etiquetas y los atributos. No se necesitan bibliotecas ni servidores de aplicaciones especiales para leer un documento XML (aunque su entorno de desarrollo puede necesitar una configuración especial para ser sensible a XML). Los documentos XML son archivos de texto normal, por lo que no requieren un software propietario para interpretarlos, como ocurre con la mayoría de los archivos binarios. Esto significa que puede usar el Bloc de notas para abrir y editar un archivo XML.

Por tanto, XML es la elección correcta cuando se necesita intercambiar información a través de varias plataformas (incompatibles) de hardware o de software, o de varias aplicaciones. Tal es el caso de aplicaciones de XML en tecnologías de la comunicación como, por ejemplo, el popular WML (lenguaje de formato inalámbrico) y WAP (protocolo

de aplicaciones inalámbricas). WML es un lenguaje basado en XML que se utiliza para dar formato a aplicaciones de Internet para dispositivos de mano, como teléfonos o PDA.

La portabilidad de XML también resulta útil en aplicaciones de comercio electrónico entre empresas (B2B), donde necesitan intercambiar una gran cantidad de información financiera de forma independiente de la plataforma. Diversas aplicaciones utilizan SOAP (Protocolo sencillo de acceso a objetos), un protocolo muy popular basado en XML, para intercambiar este tipo de información a través de Internet. Estas aplicaciones basadas en XML que se utilizan para compartir información se denominan servicios Web.

El contenido es independiente de la presentación

Con XML, se puede reducir el riesgo de incluir contenido redundante. Sus clientes se concentrarán en usar HTML y CSS para definir el diseño y la presentación, lo cual no se verá afectado por los cambios en la información subyacente, que se almacena por separado en un archivo XML.

Por ejemplo, un sistema de administración de contenido podría suministrar documentos a usuarios finales en varios formatos: HTML, PDF, etc. Sin embargo, no tiene sentido almacenar múltiples versiones (una en cada formato) de un mismo documento. El contenido se duplicaría y ocuparía un valioso espacio en el disco, lo cual llenaría el CMS de información redundante y ralentizaría su uso. Si se utiliza un motor basado en XML, el contenido puede almacenarse una sola vez para extraerlo y mostrarlo posteriormente en el formato deseado.

Si XML separa el contenido de la presentación

Todos los requisitos de procesamiento o formato deben ser manejados por un documento XSL independiente (Lenguaje ampliable de hojas de estilos). Una hoja de estilos XSL especifica la presentación de los datos contenidos en un archivo XML. A la hora de mostrar los datos, XML y XSL se combinan y aplican el formato adecuado a los datos de la misma forma que las CSS (hojas de estilos en cascada) permiten aplicar estilos a HTML.

Si tiene en cuenta las tres ventajas más destacadas de XML, entenderá inmediatamente en qué contextos debería usar XML:

- En aplicaciones que manejan gran cantidad de datos y, a la vez, necesitan ser flexibles y ampliables. Por ejemplo, sitios Web, listas de ofertas de empleo o aplicaciones financieras.
- En aplicaciones donde la presencia de contenidos redundantes sea un peligro, como en sistemas de administración de contenido, bibliotecas de documentos o sistemas de seguimiento de sitios Web.
- En aplicaciones donde es necesario intercambiar gran cantidad de datos a través de distintas plataformas, como las aplicaciones B2B, clientes de correo electrónico compatibles con servidores de noticias o dispositivos móviles.
- En aquellas situaciones donde la información debe estar disponible para un gran número de clientes. Por ejemplo, titulares de noticias, comunicados de prensa de

empresas, avisos y anuncios importantes, marcadores, listas de reproducción, calendarios de eventos o listas de correo.

En la actualidad, XML se usa habitualmente para transferir datos entre diferentes aplicaciones de bases de datos. La mayoría de los DBMS (sistemas de administración de bases de datos), incluidos MySQL o Microsoft Access, permiten exportar tablas de bases de datos como archivos XML.

Una de las aplicaciones más populares de XML es RSS (Distribución realmente sencilla). RSS es otro tipo de formato de documento XML diseñado para distribuir noticias y contenido similar. Algunos ejemplos de sitios Web que utilizan RSS son comunidades como Wired y Slashdot, o blogs personales, entre otros. La ventaja principal de RSS es que las personas leen el contenido en el formato elegido por ellos y lo pueden importar en sus propios sitios Web.

Si nos fijamos en algunas de las desventajas de XML, es fácil imaginarse dónde no se debería usar.

- No es aconsejable usar XML en sitios Web personales pequeños, o en sitios Web de presentación de empresas, ya que estos sitios contienen poca información o tienen pocas páginas. En este caso, es más recomendable utilizar HTML estático o diseños CSS/Flash.
- XML no debería emplearse en sitios Web con enormes cantidades de datos, en los cuales la velocidad de recuperación de datos y la seguridad resultan cruciales. En este caso, las bases de datos tradicionales proporcionan una solución mucho más eficiente. Algunos ejemplos de estas aplicaciones son los sitios Web corporativos, almacenes de datos, etc.
- XML no resulta aconsejable como sustituto de HTML ni de bases de datos. Si su sitio Web ya se basa en una base de datos, no hay ningún motivo para cambiarse a XML. No obstante, puede añadir un agregador RSS para ofrecer noticias o anuncios de su empresa y permitir a los clientes que importen esta información directamente en sus sitios Web.

Sintaxis XML

La sintaxis es bastante simple y sus reglas son claras y sencillas. Un documento XML está formado por una declaración XML y un elemento raíz o una etiqueta que contiene varios elementos anidados. Para empezar, enumeraremos las reglas de sintaxis más importantes:

- Todos los documentos XML deben tener un elemento raíz.
- Todos los elementos XML deben tener una etiqueta de cierre.
- Las etiquetas distinguen entre mayúsculas y minúsculas.
- Todos los elementos XML deben estar anidados correctamente.
- Los atributos deben estar incluidos en la etiqueta de apertura y deben ser escritos entre comillas.

Todos los documentos XML deben empezar con la declaración XML. Las aplicaciones que llaman al documento XML utilizan la declaración XML con el fin de leer e interpretar correctamente la información. La declaración XML no es un elemento y no se trata como parte de un documento XML.

El siguiente aspecto a considerar es que el documento debería contener un único elemento raíz. En el ejemplo anterior, el elemento raíz es <departmento>. Imagínese, sin embargo, que la empresa tiene más de un departamento. ¿Se podría añadir un segundo elemento <departmento> al documento, como el que se muestra a continuación?

```
<?xml version="1.0" encoding="iso-8859-1"?>
<departmento> </departmento>
<departmento> </departmento>
```

No. En este caso, tendría que definir un nuevo elemento raíz: <compania>. El nuevo elemento raíz puede tener ahora todos los elementos hijo (departamentos) que se deseen (observe que no se utiliza la ñ):

```
<compania>
 <departmento>
       <empleado>
          <nombre>María Duque</nombre>
          <trabajo>Desarrollador de Software </trabajo>
          <salario>800000</salario>
      </empleado>
       <empleado>
          <nombre>Juan Medina</nombre>
          <job>Diseñador</job>
          <salario>900000</salario>
      </empleado>
 </departmento>
 <departmento>
   <empleado>
   </empleado>
 </departmento>
</compania>
```

El resto de los elementos hijo debe incluirse en el ámbito de la etiqueta raíz.

En HTML se pueden utilizar elementos con una única etiqueta como <hr> o
. Sin embargo, en XML, todos los elementos deben tener una etiqueta de cierre. Si se omite la etiqueta de cierre, el navegador devolverá un error similar a éste:

The following tags were not closed: department. Error processing resource 'http://www.domain.org/company.xml'.

Además, los nombres de etiquetas distinguen entre mayúsculas y minúsculas. Por lo tanto, <Departmento> es un elemento totalmente distinto a <departmento> o a <DEPARTMENTO>. Obviamente, las etiquetas de apertura y cierre de un mismo elemento deben escribirse con el mismo tipo de letras (mayúsculas o minúsculas). El siguiente ejemplo muestra un par de etiquetas no válidas en XML:

<JOB>Software Analyst </job>

Como hemos visto antes, los elementos XML se relacionan mediante relaciones padre-hijo. En el ejemplo anterior, <empleado> es un hijo de <departmento>, el cual, a su vez, es un hijo del elemento raíz único, <compania>. Para preservar estas relaciones, los elementos deben estar correctamente anidados. En HTML, las etiquetas pueden entrelazarse, como se muestra en el siguiente ejemplo. Sin embargo, esto no es posible en XML, ya que los elementos deben estar siempre anidados de forma jerárquica.

Este texto aparece con negrita y cursiva.
Esto es perfectamente válido en HTML y se muestra en un navegador. Este solapamiento, que no debería hacerse en HTML pero que sin embargo es admitido, en XML no es posible hacerlo.

En XML, el contenido o la información real se almacena en los elementos y/o en sus atributos. Un elemento puede contener texto sencillo, otros elementos o ambos. Por ejemplo, el siguiente elemento:

```
<employee>
        <name>John Doe</name>
        <job>Software Analyst</job>
        <salary>2000</salary>
</employee>
```

se puede escribir también del siguiente modo:

```
<employee>
        John Doe
        <job>Software Analyst</job>
        <salary>2000</salary>
</employee>
```

Esto significa que el elemento employee tiene un contenido mixto: texto sencillo y otros elementos.

También se permite el uso de elementos vacíos. El siguiente elemento se podría interpretar como "tenemos una oferta de empleo, pero todavía estamos buscando a la persona adecuada".

```
<employee></employee>
El mismo elemento se podría escribir de otra forma utilizando atributos:
<employee job="Software Analyst">
        John Doe
        <salary>2000</salary>
</employee>
```

En XML, se llama atributos a las propiedades de un elemento. Los atributos describen sus características. Se pueden utilizar comillas sencillas (' ') o dobles (" ") para marcar los valores de los atributos. Como se puede ver en los ejemplos anteriores, los mismos datos se pueden almacenar como elementos hijo o como atributos. Entonces, ¿qué método es mejor? En teoría, los atributos se deberían utilizar sólo para proporcionar información adicional sobre los datos; es decir, cuando se necesitan metadatos. Por ejemplo:

```
<employee id="31">
        <name>John Doe</name>
        <job>Software Analyst</job>
        <salary>2000</salary>
</employee>
```

El ID del empleado (employee id) no es relevante en este caso para los datos. Sin embargo, este ID puede ser usado por un software que procese XML para identificar al empleado con mayor rapidez. Este tipo de información se denomina metadatos, es decir, datos sobre los datos.

Usar atributos en lugar de elementos tiene también ciertas desventajas. La estructura global del documento XML es menos clara y menos ampliable. Además, los atributos no pueden tener múltiples valores y resulta más complicado trabajar con ellos. I magine por ejemplo que la información de un empleado se almacenara de la siguiente manera:

```
<employee name="John Doe" job="Software Analyst" salary="2000"></employee>
```

Esto iría totalmente en contra del propósito de un documento XML: mostrar la información bien estructurada y fácil de intercambiar.

Tipos de nombres se pueden utilizar

Llegados a este punto, es posible que se haga la siguiente pregunta: "De acuerdo, si yo puedo definir mis propias etiquetas, ¿puedo usar cualquier nombre para un elemento?" La respuesta es sí y no. Se puede utilizar cualquier cosa como nombre de un elemento, puesto que no existen palabras reservadas en XML, pero se deben seguir unas sencillas reglas de asignación de nombres:

- Los nombres pueden contener cualquier carácter alfanumérico, pero no pueden comenzar con un número o un carácter de puntuación.
- Los nombres no pueden contener espacios.

- Los nombres no pueden comenzar con las letras xml, ya que se podría confundir con una definición de un documento XML.
- No se deben usar caracteres ":" en los nombres de elementos.

Aunque está permitido utilizar los caracteres "." y "-" en los nombres de elementos, no es recomendable. La aplicación que procesa el archivo XML podría interpretar estos signos como operadores. Si necesita usar un nombre largo, puede sustituir los símbolos anteriores por el carácter "_", como se muestra en el siguiente ejemplo:

```
<employee>
        <first_name>John</first_name>
        <last_name>Doe</last_name>
        <job>Software Analyst</job>
        <salary>2000</salary>
</employee>
```

El contenido de un elemento

Se puede utilizar prácticamente cualquier cosa como contenido. También se pueden utilizar caracteres que no sean en inglés, pero asegúrese de que selecciona el juego de caracteres correcto y de que la aplicación cliente que procesa el documento XML admite contenido que no esté en inglés. Además, a diferencia de lo que ocurre en HTML, se conservan los espacios en blanco que aparecen dentro del contenido. Esto significa que se pueden escribir varios espacios seguidos sin que éstos sean eliminados.

Todos los lenguajes de programación o de formato permiten el uso de comentarios, así que XML también lo permite. La sintaxis es igual que en HTML:

```
<!-- This employee deserves a raise. -->
```

Un RSS es una aplicación de XML. En el siguiente ejemplo se muestra una versión simplificada de un agregador RSS del Centro de desarrolladores de Macromedia con el objetivo de ilustrar la estructura de los documentos XML y las reglas de sintaxis que se aplican.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="1.0">
        <channel>
                <title>Macromedia Developer Center RSS Feed June 27, 2005</title>
                <link>http://www.macromedia.com/devnet/</link>
                <description>Macromedia Developer Center is your center for the
tutorials, articles, and sample applications you need to master Macromedia
products.</description>
        <item>
                <title>Creating a Dynamic Playlist for Progressive Flash Video</title>
```

```
        <link>http://www.macromedia.com/devnet/flash/articles/prog_download.html</link
>
                <description>Learn how to create an XML-driven playlist for viewing
progressive FLV files.</description>
        </item>
        <item>
                <title>Chatting Through IM Gateways in ColdFusion MX 7</title>

        <link>http://www.macromedia.com/devnet/coldfusion/articles/imgateway.html</lin
k>
                <description>Learn the fundamentals of gateway apps as you build a
sample chat application that monitors your ColdFusion server status</description>
        </item>
</channel>
</rss>
```

Observe cómo RSS almacena información orientada a listas (elementos o artículos). Por supuesto, RSS ha sido diseñado para intercambiar grandes cantidades de elementos similares pero, para simplificarlo, sólo se han incluido dos elementos en el ejemplo anterior. Los elementos componen un "canal" de información, el cual consta de un título, un vínculo y una descripción. Cada elemento representa un artículo publicado en el Centro de desarrolladores y tiene como elementos hijo el título del artículo, un vínculo al mismo y una breve descripción. Además, se podrían añadir elementos para identificar el autor, la fecha de publicación y el tema de cada artículo, como se podrá comprobar en mis próximos artículos. Aquí se explica cómo escribir código XML correctamente desde el punto de vista de la sintaxis y las reglas. Cuando se familiarice más con XML, aprenderá a manejar las definiciones de tipo de documento (DTD) y los esquemas XML, que son las reglas que se definen para los lenguajes basados en XML que usted crea y utiliza. Una DTD sirve para especificar los elementos válidos que se pueden usar en un documento XML. Un esquema XML es simplemente una versión de DTD basada en XML.

BIBLIOGRAFÍA Y CIBERGRAFÍA

TEXTOS

MONTOYA MONTOYA, Jaime Eduardo. Guía Académica Programación para la Web. Medellín: Cesde, Semestre 1-2013. 198p.

GUTÍERREZ, Abraham; BRAVO, Ginés. PHP 5 a través de ejemplos. México D.F.: Alfaomega RA-MA, 2005.

LÓPEZ QUIJADO, José. Domine PHP MySQL: programación dinámica en el lado del servidor. México, Alfaomega RA-MA, 2007

GIL RUBIO, Francisco Javier; TEJEDOR CERBEL; Jorge A. y YAGUE PANADERO, Agustín. Creación de sitios web con PHP 4. Medelin: McGraw-Hill, 2001. 547 p. 1 Cd-rom.

ZAKAS, Nicholas C. JavaScript para desarrolladores web. Madrid: Anaya Multimedia, 2005.

DANESH, Arman; DÍAZ MENA, Jorge Iván y GARCÍA, María Margarita. Aprendiendo Javascript en una semana. España: Prentice Hall Hispanoamericana, s.a., 1996. 543 p.

FRENTZEN, Jeff. Superutilidades para Javascript. Santafé de Bogotá: McGraw-Hill, c1999. 1 Cd-rom.

KORTH, Henry F., SILBERSCHATZ, Abraham. Fundamentos de Bases de Datos. Madrid: McGraw-Hill, Segunda edición, 1992.

ENLACES EN INTERNET

Manual de PHP. En internet: http://www.php.net/manual/es/index.php

MySQL 5.0 Reference Manual. En internet: http://dev.mysql.com/doc/refman/5.0/es/index.html

Desarrollo Web.com. Tú mejor ayuda para aprender a hacer webs. En internet: http://www.desarrolloweb.com

Wikipedia, la enciclopedia libre. PHP. En internet: http://es.wikipedia.org/wiki/PHP

Wikipedia, la enciclopedia libre. MySQL. En internet: http://es.wikipedia.org/wiki/MySQL

Wikipedia, la enciclopedia libre. HTML. En internet: http://es.wikipedia.org/wiki/HTML

Wikipedia, la enciclopedia libre. DHTML. En internet: http://es.wikipedia.org/wiki/DHTML

Wikipedia, la enciclopedia libre. Tim Barners-Lee. En internet: http://es.wikipedia.org/wiki/Tim_Barners-Lee

Wikipedia, la enciclopedia libre. En internet: http://es.wikipedia.org/wiki/ICANN

Wikipedia, la enciclopedia libre. Internet. En internet: http://es.wikipedia.org/wiki/Internet

Wikipedia, la enciclopedia libre. World Wide Web. En internet: http://es.wikipedia.org/wiki/World_Wide_Web

WebEstilo. Usabilidad, programación y mucho más. En internet: http://www.webestilo.com

WebTaller. El portal del Webmaster. En internet: http://www.webtaller.com

Programación Web. Programación y diseño de páginas web. En internet: http://www.programacionweb.net

Programación Fácil. En internet: http://www.programacionfacil.com

Tejedores del Web. En internet: http://www.tejedoresdelweb.com

Programatium. Comunidad de programación por y para programadores. En internet: http://www.programatium.com

Programación y diseño web. Todo sobre programación web & más. En internet: http://www.todo-programacion.com.ar/

La Web del Programador: Comunidad de Programadores. En internet: http://www.lawebdelprogramador.com

Manual de PHP. En internet: http://www.manualdephp.es

Manual de PHP. En internet: http://www.manualdephp.com

El Guru Programador: Actualidad, Programación y Desarrollo Web. En internet: http://www.elguruprogramador.com.ar

Tutorial de PHP. En internet: http://flanagan.ugr.es/php/

Manual de PHP 4. En internet: http://www.laigu.net/ManualPHP.html

Turorial de PHP. En internet: http://www.tutorialesfacil.com.ar/php/

Online Language Dictionaries. En internet: http://www.wordreference.com/es/index.htm

Estadísticas históricas y de uso diario. En internet: http://news.netcraft.com

En internet: http://www.htmlquick.com/es/tutorials/forms.html

En internet: http://es.kioskea.net/contents/html/htmlform.php3

En internet: http://webusable.com/CharactersTable.htm

En internet: http://webusable.com/CharsExtendedTable.htm

En internet: http://html.conclase.net/w3c/html401-es/struct/lists.html

En internet: http://goliatenterrado.es/2009/03/03/configurar-el-mercury32-del-xampp-para-enviar-correos-externos/

En internet: http://www.hackingballz.com/herramientas/manual-oficial-de-php/function.date.html

En internet: http://www.tuxi.com.ar/2007/06/18/javascript-validar-e-mail-con-expresiones-regulares/

Wikipedia, la enciclopedia libre. Correo Yahoo! En internet: http://es.wikipedia.org/wiki/Correo_Yahoo!

Wikipedia, la enciclopedia libre. GNU. En internet: http://es.wikipedia.org/wiki/GNU

Wikipedia, la enciclopedia libre. GNU - Licencia Pública General. En internet: http://es.wikipedia.org/wiki/GNU_General_Public_License

Wikipedia, la enciclopedia libre. W3C. En internet: http://www.w3.org/

W3C. En internet: http://www.w3c.org/ (W3 España)

W3C. En internet: http://validator.w3.org/

Wikipedia, la enciclopedia libre. WYSIWYG. En internet: http://es.wikipedia.org/wiki/World_Wide_Web_Consortium

Wikipedia, la enciclopedia libre. WYSIWYG. En internet: http://es.wikipedia.org/wiki/WYSIWYG

Wikipedia, la enciclopedia libre. Cookie (Informática). En internet: http://es.wikipedia.org/wiki/Cookie_(informática)