II. JSP, JSTL y XML: Tecnologías en la capa de Presentación

Macario Polo,
Daniel Villafranca

CONTENIDO:

1. CSS: Hojas de estilo

CSS (*Cascading Style Sheets*, Hojas de estilo en Cascada), es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias a las CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como incluir márgenes, tipos de letra, fondos, colores...

Las Hojas de Estilo en Cascada se escriben dentro del código HTML de la página web, solo en casos avanzados se pueden escribir en un archivo a parte y enlazar la página con ese archivo. En un principio vamos a utilizar la manera más directa de aplicar estilos a los elementos de la página, mas adelante veremos la declaración en archivos externos. Para ello, y esto es la primera lección de este artículo, vamos a conocer un nuevo atributo que se puede utilizar en casi todas las etiquetas HTML: style.

Ejemplos:

```
<p style="color:green;font-weight:bold">El párrafo saldrá con color verde y en negrita</p>
```

Dentro del atributo style se deben indicar los atributos de estilos CSS separados por punto y coma (;). Durante este artículo vamos a conocer muchos atributos de CSS, los dos primeros que hemos visto aquí son:

Color: indica el color del contenido la etiqueta donde estemos utilizándolo, generalmente indica el color del texto.

Font-weight: indica el grosor del texto. Bold sirve para poner en negrita.
(*Ver más ejemplos en las referencias*)

Incluir estilos para todo un sitio web

Una de las características más potentes de la programación con hojas de estilo consiste en definir los estilos de todo un sitio web. Esto se consigue creando un archivo donde tan sólo colocamos las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas las páginas comparten una misma declaración de estilos y, por tanto, si la cambiamos, cambiarán todas las páginas. El proceso para incluir estilos con un

fichero externo, sería:

1- Creamos el fichero con la declaración de estilos: Es un fichero de texto normal, que puede tener cualquier extensión, aunque le podemos asignar la extensión .css para aclararnos qué tipo de archivo es. El texto que debemos incluir debe ser escrito exclusivamente en sintaxis CSS, es un poco distinta que la sintaxis que utilizamos dentro del atributo style. Sería erróneo incluir código HTML en este archivo: etiquetas y demás. Podemos ver un ejemplo a continuación:

```
P {
font-size : 12pt;
font-family : arial,helvetica;
font-weight : normal;
}
H1 {
font-size : 36pt;
font-family : verdana,arial;
text-decoration : underline;
text-align : center;
background-color : Teal;
}
BODY {
background-color : #006600;
font-family : arial;
color : White;
}
```

2- Enlazamos la página web con la hoja de estilos: Para ello vamos a colocar la etiqueta <LINK> con los atributos

- rel="STYLESHEET" indicando que el enlace es con una hoja de estilo.
- type="text/css" porque el archivo es de texto, en sintaxis CSS.
- href="estilos.css" indica el nombre del fichero fuente de los estilos.

Veamos una página web entera que enlaza con la declaración de estilos anterior:

```
<html>
      <head>
            <link rel="STYLESHEET" type="text/css" href="estilos.css">
            <title>Página que lee estilos</title>
      </head>
      <body>
            <h1>Página que lee estilos</h1>
            <p> Esta página tiene en la cabecera la etiqueta necesaria
            para enlazar con la hoja de estilos. Es muy fácil.
            </p>
      </body>
</html>
```

2. JSP y XML en la tecnología Web

El "Extensible Markup Language" (XML) se ha convertido en el formato estándar de facto para la representación de datos en Internet. Los datos XML se pueden procesar e interpretar en cualquier plataforma (ej. desde el dispositivo portátil a un servidor central).

XML es un metalenguaje usado para definir documentos que contienen datos estructurados. Las características y beneficios del XML se pueden agrupar en estas áreas principales:

- **Extensibilidad: como metalenguaje**, XML puede usarse para crear sus propios lenguajes de marcas. Hoy en día existen muchos lenguajes de marcas basados en XML, incluyendo "Wireless Markup Language" (WML).
- **Estructura precisa**: HTML sufre de una pobre estructura que hace dificil procesar eficientemente documentos HTML. Por otro lado, los documentos XML están bien estructurados, cada documento tiene un elemento raíz y todos los demás elementos deben estar anidados dentro de otros elementos.
- Dos tipos de documentos: hay dos tipos principales de documentos XML.
    1. *Documentos Válidos*: un documento XML válido está definido por una "Document Type Definition" (DTD), que es la gramática del documento que define qué tipos de elementos, atributos y entidades podría haber en el documento. El DTD define el orden y también la ocurrencia de elementos.
    2. *Documentos Bien-Formateados*: un documento XML bien formateado no tiene que adherirse a una DTD. Pero debe seguir dos reglas: 1) todo elemento debe tener una etiqueta de apertura y otra de cierre. 2) debe haber un elemento raíz que contenga todos los otros elementos.
- **Extensión Poderosa**: Como se mencionó anteriormente, XML sólo se usa para definir la sintaxis o el contenido. Para definir la semántica, el estilo o la presentación, necesitamos usar "Extensible Stylesheet Language" (XSL). Observa que un documento podría tener múltiples hojas de estilo que podrían ser usadas por diferentes usuarios.

Presentar Documentos XML

Los documentos XML contienen datos portables. Esto significa que el ejemplo 1 puede procesarse como salida para diferentes navegadores (navegadores de escritorio para PC, micronavegadores para dispositivos de mano). En otras palabras, un documento XML puede ser transformado en HTML o WML o cualquier otro lenguaje de marcas.

Si cargamos un documento xml en un navegador que soporte XML (como IE de Microsoft), veríamos algo similar lo siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <portfolio>
  - <stock>
      <symbol>SUNW</symbol>
      <name>Sun Microsystems</name>
      <price>17.1</price>
    </stock>
  - <stock>
      <symbol>AOL</symbol>
      <name>America Online</name>
      <price>51.05</price>
    </stock>
  - <stock>
      <symbol>IBM</symbol>
      <name>International Business Machines</name>
      <price>116.10</price>
    </stock>
  - <stock>
      <symbol>MOT</symbol>
      <name>MOTOROLA</name>
      <price>15.20</price>
    </stock>
</portfolio>
```

A la hora de mostrar documentos XML se suele aplicar una transformación sobre el documento XML, para extraer los datos o para crear un nuevo formato (como transformar datos XML a HTML). Esta transformación se puede hacer usando un lenguaje de transformación como "*Extensible Stylesheet Language Transformation*" (XSLT), que forma parte de XSL. XSL permite que escribamos el vocabulario XML para especificar la semántica del formato. Es decir hay dos partes de XSL, que son parte de la actividad de estilo del World Wide Web Consortium (W3C).

- Lenguaje de Transformación (XSLT)
- Lenguaje de Formateo (objetos de formateo XSL)

Para generar las marcas HTML y extrae datos de los elementos del documento xml anterior haremos lo siguiente:

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl=
"http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
  <html>
  <head>
  <title>Stocks</title>
  </head>
  <body bgcolor="#ffffcc" text="#0000ff">
    <xsl:apply-templates/>
```

```
  </body>
  </html>
</xsl:template>

<xsl:template match="portfolio">

   <table border="2" width="50%">
     <tr>
      <th>Stock Symbol</th>
      <th>Company Name</th>
      <th>Price</th>
     </tr>
     <xsl:for-each select="stock">
       <tr>
         <td>
           <i><xsl:value-of select="symbol"/></i>
         </td>
       <td>
          <xsl:value-of select="name"/>
        </td>
       <td>
          <xsl:value-of select="price"/>
       </td>

        </tr>
      </xsl:for-each>
    </table>
  </xsl:template>

</xsl:stylesheet>
```

Sobre la sintaxis anterior se ha utilizado:

- *xsl:stylesheet*: elemento raíz
- *xsl:template*: cómo transformar los nodos seleccionados
- *match*: atributo para seleccionar un nodo
- *"/"*: nodo raíz del documento XML de entrada
- *xsl:apply-templates*: aplica las plantillas a los hijos del nodo seleccionado
- *xsl:value-of*: nodo seleccionados (extrae datos de los nodos seleccionados)

Ahora la cuestión es: ¿cómo usar esta hoja de estilos XSL con el documento stocks.xml? La respuesta es sencilla, necesitamos modificar la primera línea del documento stocks.xml del Ejemplo 1 para usar la hoja de estilos stocks.xsl para su representación. La primera línea del ejemplo 1 ahora debería ser:

```
<?xml:stylesheet type="text/xsl" href="stocks.xsl" version="1.0"
encoding="UTF-8"?>
```

Esto dice básicamente que cuando cargamos stocks.xml en un navegador (será analizado como un árbol de nodos), se debe utilizar la hoja de estilos correspondiente para extraer datos de los nodos del árbol. Cuando cargamos el fichero stocks.xml modificado en un navegador que soporta XML y XSL (como IE de Microsoft), veremos algo similar a la figura siguiente:

Stock Symbol	Company Name	Price
SUNW	Sun Microsystems	17.1
AOL	America Online	51.05
IBM	International Business Machines	116.10
MOT	MOTOROLA	15.20

Generar documentos XML desde JSP y JavaBeans

Se puede usar la tecnología JSP para generar documentos XML. Una página JSP podría generar fácilmente una respuesta conteniendo el documento stocks.xml del Ejemplo 1, como se ve en el ejemplo 3. El principal requerimiento para generar XML es que la página JSP seleccione el tipo de contenido de la página de forma apropiada. Como podemos ver a partir del ejemplo 3, el tipo de contenido se selecciona a text/xml. Se puede usar la misma técnica para generar otros lenguajes de marcas (como WML).

```
<%@ page contentType="text/xml" %>

<?xml version="1.0" encoding="UTF-8"?>
<portfolio>
  <stock>
    <symbol>SUNW</symbol>
    <name>Sun Microsystems</name>
    <price>17.1</price>
  </stock>
  <stock>
    <symbol>AOL</symbol>
    <name>America Online</name>
    <price>51.05</price>
  </stock>
  <stock>
    <symbol>IBM</symbol>
    <name>International Business
    Machines</name>
    <price>116.10</price>
  </stock>
  <stock>
    <symbol>MOT</symbol>
    <name>MOTOROLA</name>
    <price>15.20</price>
  </stock>
</portfolio>
```

Si solicitamos la página jsp anterior desde un servidor Web (como Tomcat), veríamos algo similar a la figura 1. Dejamos como ejercicio la generación de XML desde un componente (JavaBeen) y la modificación de la página JSP aplicando una hoja de estilo XSL para que se obtenga un resultado similar a la figura 2 (Solución en las referencias).

3. JSTL y desarrollar Etiquetas JSP Personalizadas

La librería JSTL (JSP Standard Tag Library) es un componente dentro de la especificación del Java 2 Enterprise Edition (J2EE) y es controlada por Sun MicroSystems. JSTL no es más que un conjunto de librerías de etiquetas simples y estándares que encapsulan la funcionalidad principal que es usada comúnmente para escribir páginas JSP. Las etiquetas JSTL están organizadas en 4 librerías:

- **core**: Comprende las funciones script básicas como loops, condicionales, y E/S.
- **xml**: Comprende el procesamiento de xml
- **fmt**: Comprende la internacionalización y formato de valores (moneda, fechas,..)
- **sql**: Comprende el acceso a base de datos.

La librería JSTL es distribuida como un conjunto de archivos JAR que simplemente tenemos que agregarlo en el classpath del contenedor de servlets

Ventajas e inconvenientes del uso de JSTL frente a scriptlets

La especificación JSP ahora se ha convertido en una tecnología estándar para la creación de sitios Web dinámicos en Java, y el problema es que han aparecido algunas debilidades:

- Aunque es más potente que las etiquetas JSTL, el código Java embebido en scriptlets es desordenado.
- Un programador que no conoce Java no puede modificar el código Java embebido, anulando uno de los mayores beneficios de los JSP: permitir a los programadores que escriben la lógica de presentación que actualicen el contenido de la página.
- El código de Java dentro de scriptlets JSP no pueden ser reutilizados por otros JSP, por lo tanto la lógica común termina siendo re-implementado en múltiples páginas.
- La recuperación de objetos fuera del HTTP Request y Session es complicada. Es necesario hacer el Casting de objetos y esto ocasiona que tengamos que importar más Clases en los JSP.

Los JSTL pueden agregar mayor sobrecarga en el servidor. Los scriptlets y las librerías de etiquetas son compilados a servlets, los cuales luego son ejecutados por el contenedor. El código Java embebido en los scriptlets es básicamente copiado en el servlet resultante. En cambio, las etiquetas JSTL, causan un poco más de código en el servlet. En la mayoría de casos esta cantidad no es mensurable pero debe ser considerado.

Un ejemplo de JSTL

Para demostrar el concepto de ordenamiento y mejor mantenimiento de los JSP, veamos 2 versiones de una página simple. Ambas páginas implementan la misma lógica: Graban una lista de objetos AddressVO del Request y luego iteran a través de la lista, imprimiendo el atributo apellido de cada objeto (si el apellido no es null y de longitud diferente a 0). En cualquier otro caso, imprimirá "N/A". Finalmente, la página imprime la fecha actual.

Con scriptlets JSP:

```
<%@ page import="com.ktaylor.model.AddressVO, java.util.*"%>

<p><h1>Customer Names</h1></p>
<%
  List addresses = (List)request.getAttribute("addresses");
  Iterator addressIter = addresses.iterator();
   while(addressIter.hasNext()) {
     AddressVO address = (AddressVO)addressIter.next();
     if((null != address) &&
     (null != address.getLastName()) &&
     (address.getLastName().length() > 0)) {
%>
        <%=address.getLastName()%><br/>
<%
     }
     else {
%>
        N/A<br/>
<%
     }
   }
%>
<p><h5>Last Updated on: <%=new Date()%></h5></p>
```

Con JSTL:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<p><h1>Customer Names</h1></p>

<c:forEach items="${addresses}" var="address">
   <c:choose>
      <c:when test="${not empty address.lastName}" >
        <c:out value="${address.lastName}"/><br/>
      </c:when>
      <c:otherwise>
        N/A<br/>
      </c:otherwise>
   </c:choose><br/>
 </c:forEach><br/>

 <jsp:useBean id="now" class="java.util.Date" />

 <p><h5>Last Updated on: <c:out value="${now}"/></h5></p>
```

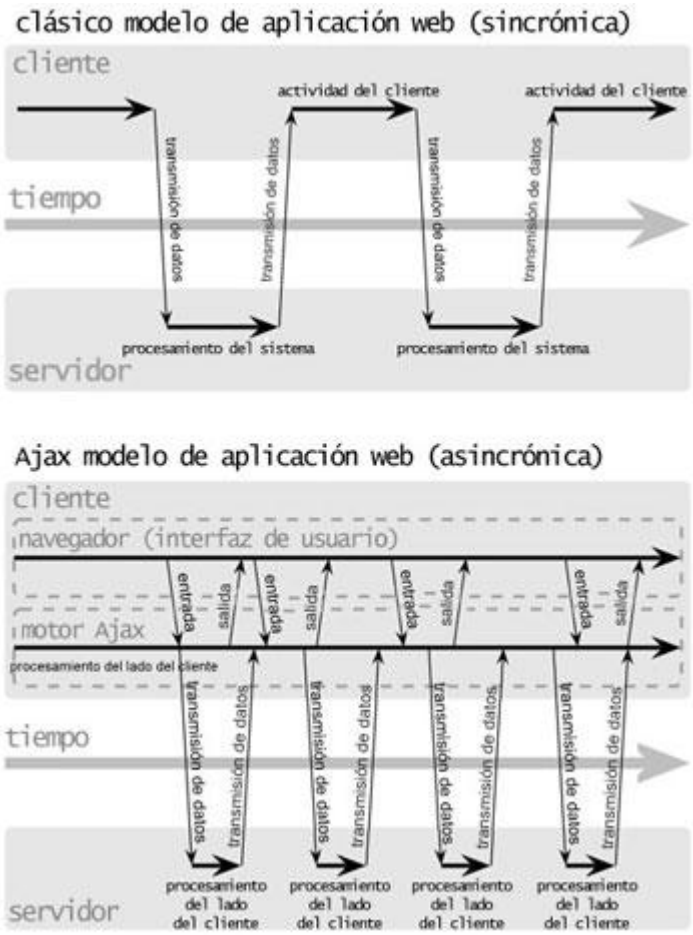
4. Breve introducción a AJAX

AJAX no es una tecnología, sino la unión de varias tecnologías que juntas permiten realizar nuevas funcionalidades en la capa de presentación. AJAX, es el acrónimo para Asynchronous JavaScript + XML y el concepto es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en *background*, los datos que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma.

El modelo clásico de aplicaciones Web funciona de esta forma: La mayoría de las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web. El servidor efectúa un proceso (recopila información, procesa números, hablando con varios sistemas propietarios), y le devuelve una pagina HTLM al cliente. En un modelo adaptado, las peticiones del cliente se aceleran mediante el motor de Ajax.

En vez de cargar un pagina Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript y usualmente "sacado" en un frame oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.



Cada acción de un usuario que normalmente generaría un requerimiento HTTP toma la forma de un llamado JavaScript al motor AJAX en vez de ese requerimiento. Cualquier respuesta a una acción del usuario que no requiera una viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejado por su cuenta.

5. Ejercicios prácticos

- Realizar la adaptación a JSTL del código Javascript entregado en la sesión 1.

Próximas sesiones

Implementación de MVC mediante Struts

EJB y acceso a datos (JDBC e Hibernate)

Depuración de aplicaciones Web

La capa servidor y uso de Servicios Web

Anexo I. Atributos de la Hoja de estilos (CSS)

Nombre del atributo	Posibles valores	Ejemplos
FUENTES - FONT		
color	valor RGB o nombre de color	color: #009900; color: red;
Sirve para indicar el color del texto. Lo admiten casi todas las etiqetas de HTML. No todos los nombres de colores son admitidos en el estandar, es aconsejable entonces utilizar el valor RGB.		
font-size	xx-small | x-small | small | medium | large | x-large | xx-large Unidades de CSS	font-size:12pt; font-size: x-large;
Sirve para indicar el tamaño de las fuentes de manera más rígida y con mayor exactitud.		
font-family	serif | sans-serif | cursive | fantasy | monospace Todas las fuentes habituales	font-family:arial,helvetica; font-size: fantasy;
Con este atributo indicamos la familia de tipografia del texto. Los primeros valores son genéricos, es decir, los exploradores las comprenden y utilizan las fuentes que el usuario tenga en su sistema. También se pueden definir con tipografías normales, como ocurría en html. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.		
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	font-weight:bold; font-weight: 200;
Sirve para definir la anchura de los caracteres, o dicho de otra manera, para poner negrillas con total libertad. Normal y 400 son el mismo valor, así como bold y 700.		
font-style	normal | italic | oblique	font-style:normal; font-style: italic;
Es el estilo de la fuente, que puede ser normal, itálica u oblícua. El estilo oblique es similar al italic.		
PÁRRAFOS - TEXT		
line-height	normal y unidades CSS	line-height: 12px; line-height: normal;
El alto de una línea,y por tanto, el espaciado entre líneas. Es una de esas características que no se podian modificar utilizando HTML.		
text-decoration	none | [underline || overline || line-through]	text-decoration: none; text-decoration: underline;
Para establecer la decoración de un texto, es decir, si está subrayado, sobrerayado o tachado.		
text-align	left | right | center | justify	text-align: right; text-align: center;
Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque recuerda que no tiene por que funcionar en todos los sistemas.		
text-indent	Unidades CSS	text-indent: 10px; text-indent: 2in;
Un atributo que sirve para hacer sangrado o márgenes en las páginas. Muy útil y novedosa.		
text-transform	capitalize | uppercase | lowercase | none	text-transform: none;

		text-transform: capitalize;

Nos permite transformar el texto, haciendo que tenga la primera letra en mayúsculas de todas las palabrs, todo en mayúsculas o minúsculas.

FONDO - BACKGROUND		
Background-color	Un color, con su nombre o su valor RGB	background-color: green; background-color: #000055;

Sirve para indicar el color de fondo de un elemento de la página.

Background-image	El nombre de la imagen con su camino relativo o absoluto	background-image: url(mármol.gif) ; background-image: url(http://www.x.com/fondo.gif)

Colocamos con este atributo una imagen de fondo en cualquier elemento de la página,

BOX - CAJA		
Margin-left	Unidades CSS	margin-left: 1cm; margin-left: 0,5in;

Indicamos con este atributo el tamaño del margen a la izquierda

Margin-right	Unidades CSS	margin-right: 5%; margin-right: 1in;

Se utiliza para definir el tamaño del margen a la derecha

Margin-top	Unidades CSS	margin-top: 0px; margin-top: 10px;

Indicamos con este atributo el tamaño del margen arriba de la página

Margin-bottom	Unidades CSS	margin-bottom: 0pt; margin-top: 1px;

Con el se indica el tamaño del margen en la parte de abajo de la página

Padding-left	Unidades CSS	padding-left: 0.5in; padding-left: 1px;

Indica el espacio insertado, por la izquierda, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas.

El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

Padding-right	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;

Indica el espacio insertado, en este caso por la derecha, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas.

El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

Padding-top	Unidades CSS	padding-top: 10pt; padding-top: 5px;

Indica el espacio insertado, por arriba, entre el borde del elemento-continente y el contenido de este.

Padding-bottom	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;

Indica el espacio insertado, en este caso por abajo, entre el borde del elemento-continente y el contenido de este.

Border-color	color RGB y nombre de color	border-color: red; border-color: #ffccff;

Para indicar el color del borde del elemento de la página al que se lo aplicamos. Se puede poner colores por separado con los atributos border-top-color, border-right-color, border-bottom-color, border-left-color.

Border-style	none | dotted | solid | double | groove | ridge | inset | outset	border-style: solid; border-style: double;

El estilo del borde, los valores significan: none=ningun borde, dotted=punteado (no parece funcionar), solid=solido, double=doble borde, y desde groove hasta outset son bordes con varios efectos 3D.

border-width	Unidades CSS	border-width: 10px; border-width: 0.5in;

El tamaño del borde del elemento al que lo aplicamos.

float	none | left | right	float: right;

Sirve para alinear un elemento a la izquierda o la derecha haciendo que el texto se agrupe alrededor de dicho elemento. Igual que el atributo align en imagenes en sus valores right y left.

clear	none | right | left	clear: right;

Si este elemento tiene a su altura imagenes u otros elementos alineados a la derecha o la izquierda, con el atributo clear hacemos que se coloque en un lugar donde ya no tenga esos elementos a el lado que indiquemos.

Referencias

- http://www.programacion.com/java/tutorial/jspyxml/2/. *Desarrollo de aplicaciones Web con JSP y XML*. SUN (Trad. Juan Antonio Palacios)
- **http://www.1x4x9.info/files/jstl/html/online-chunked/index.html**. *JSTL*. Alejandro Ramírez
- http://www.proginternet.com.ar/css.php. *CSS*
- www.desarrolloweb.com – Varias referencias
- www.sicaman-nt.com – *Dpto. Desarrollo* – SICAMAN Nuevas Tecnologías, SL