Universidad Andrés Bello®

Conectar · Innovar · Liderar

DESARROLLO DE APLICACIONES WEB DINÁMICAS JAVA

Java Server Pages

Qué es una vista JSP y para qué sirve

JSP (Java Server Page)

Una de las tecnologías más ampliamente utilizadas para el desarrollo de aplicaciones Webbajo ambiente Java es JSP (Java Server Pages). Esta herramienta permite crear una serie de plantillas HTML que incluyen código incrustado (llamados sniplets) mediante marcadores especiales.

El uso de JSP simplifica la programación de servlets pues no es necesario compilar códigoya que esto se realiza de forma automática. Además, debido a que la plantilla se escribe directamente en HTML es factible utilizar editores y diseñadores de páginas Web para programar la aplicación.

Una página JSP es una página (X)HTML que incorpora ciertos elementos dinámicos: etiquetas especiales y pequeños fragmentos de código.

- El código HTML aparece a la salida sin modificaciones.
- Los elementos dinámicos se evalúan o ejecutan en el servidor en el momento de construcción de la respuesta.
- Aunque no es estrictamente obligatorio, una página JSP se suele transformar enel código fuente de un servlet, que después se compila y ejecuta.

Ejemplo JSP

JSP ejemplo transformado en un servlet

```
(...)
out.write("<html>\n ");
out.write("  <head>\n ");
out.write("    <title>Hola Mundo</title>\n ");
out.write("  </head>\n ");
out.write("  <body>\n    <p>Hola, esto es una página JSP.</p>\n ");
out.write("    <p>La hora del servidor es ");
out.print( new Date() );
out.write("</p>\n ");
out.write("  </body>\n");
out.write("</html>\n");
(..)
```

JSP ejemplo: documento recibido por el cliente

```
<html>
  <head>
    <title>Hola Mundo</title>
  </head>
  <body>
    <p>Hola, esto es una página JSP.</p>
    <p>La hora del servidor es Wed Nov 06 13:25:34 CET 2002</p>
  </body>
</html>
```

Directivas JSP: page

Todas las páginas JSP deberían incluirla. Atributos habituales:

- language: lenguaje de programación (java por defecto).
- contenttype: tipo de contenido de la página (text/html por defecto).
- isErrorPage: indica si es una página de error (false por defecto).
- errorPage: página a la que dirigirse si ocurre una excepción procesando esta página.

```
<%@ page language='java' contentType='text/html'
         isErrorPage='false' errorPage='error.jsp' %>
```

Otras directivas JSP

- include: permite incluir directamente el código de otro fichero en el punto en que aparezca la directiva.
- import: permite importar clases Java utilizadas en la página JSP.

<%@ include file='footer.html' %>
<%@ page import='java.util.*' %>

Scriptlets

- Permiten incrustar código escrito en otro lenguaje de programación (normalmente Java):
    - <%= expresión %>: evalúa la expresión y muestra el resultado en la página.
    - <% sentencias %>: ejecuta las sentencias, sin mostrar nada en la página.
    - <%! declaraciones %>: declaraciones de variables.

- <%!-- los siguientes scriptlets son equivalentes --%>
- <%= user.getName() %>
- <% out.println(user.getName()); %>

Scriptlets (aviso)

- Con <%! declaraciones %> se declaran atributos y métodos de instancia:

    - Todas las peticiones a la página JSP comparten la misma instancia del servlet asociado, por lo que también comparten la misma copia de los atributos de instancia.

    - Por tanto, no se debe declarar variables locales de la página con esta directiva, sino con:

```
<% String text = new String(); %>
```

Scriptlets: ejemplo

```
<table>
    <tr>
        <th>Product</th>
        <th>Price</th>
    </tr>
    <%
    for (int i=0; i<catalog.length; i++ )
    {
        ProductInfoBean product = catalog[i];
    %>
    <tr>
        <td>
            <a href='<%= response.encodeURL("view?id="
                + product.getId()) %>'>
            <%= product.getShortName() %>
            </a>
```

Variables implícitas

- JSP proporciona automáticamente una serie de variables adecuadamente declaradas e inicializadas, utilizables por el programador:

```
javax.servlet.ServletContext application
javax.servlet.ServletConfig config
javax.servlet.jsp.JspWriter out
javax.servlet.jsp.PageContext pageContext
javax.servlet.http.HttpServletRequest request
javax.servlet.http.HttpServletResponse response
javax.servlet.http.HttpSession session

<%!-- Sólo en páginas marcadas con isErrorPage: --%>
java.lang.Throwable exception
```

Acciones: jsp:include

- La acción jsp:include invoca al servlet o JSP dado e incluye en resultado de su ejecución en el punto actual del documento JSP desde el cual se incluya.
- El control retorna finalmente a la página inicial.
- Opcionalmente, se pueden pasar parámetros.

```
<jsp:include page='header.jsp'>
    <jsp:param name='title' value='Welcome' />
</jsp:include>
```

Acciones: jsp:forward

- La acción jsp:forward invoca al servlet o JSP dado e incluye en resultado de su ejecución en el punto actual del documento JSP desde el cual se incluya, sin que retorne el control a la página inicial.
- Opcionalmente, se pueden pasar parámetros.

```
<jsp:forward page='list.jsp'>
    <jsp:param name='order' value='date' />
</jsp:forward>
```

Java Beans

- Un Java Bean es una clase Java que cumple el siguiente convenio:
  - Contiene propiedades (normalmente atributos de instancia privados).
  - El acceso a las propiedades se realiza mediante métodos de acceso get, sete is.
  - Tiene siempre un constructor sin argumentos (aunque podría tener más constructores).

Java Beans: ejemplo

```
public class UserInfoBean implements java.io.Serializable
{
  private String firstName;
  private boolean registered;

  public String getFirstName() {
    return firstName;
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public boolean isRegistered() {
```

```
    return registered;
  }

  public void setRegistered(boolean registered) {
    this.registered = registered;
```

Java Beans en JSP

- JSP proporciona instrucciones especiales para trabajar más cómodamente con Java Beans.
- La acción jsp:useBean:
    - Si el bean aún no existe en el contexto:
        - Declara, crea e inicializa el bean.
        - Crea una referencia al bean con el nombre dado por id.
    - Si el bean ya existe en el contexto:
        - Obtiene una referencia al mismo con el nombre dado por id.

```
<jsp:useBean id='user' class='foo.UserInfoBean' scope='session'>
  <jsp:setProperty name='user' property='name' value='Pepe' />
</jsp:useBean>
```

Java Beans en JSP: contextos (I)

- Un bean en JSP se puede almacenar en uno de los siguientes contextos:
  - page: asociado a la página JSP y a una petición HTTP concreta, desapareceal finalizar el procesado de la página.
  - request: asociado a una petición HTTP concreta, compartida entre todos los JSPs y servlets que atiendan dicha petición, desaparece al finalizar el procesado de la petición.
  - session: asociado a la sesión, compartida por todos los JSPs y servlets para todas las peticiones de una misma sesión, desaparece al finalizar la sesión.
  - application: asociado al contexto de la aplicación Web, compartido por todos los servlets y JSPs de la aplicación en todas las peticiones.

La acción jsp:getProperty

- Se evalúa al valor de una propiedad de un bean.

```
<jsp:getProperty name='user' property='fullName' />
```

La acción jsp:setProperty

- Establece el valor de una propiedad de un bean.
- Convierte, si es necesario, el valor de la propiedad desde una cadena de texto al tipo de datos correspondiente.
- Proporciona un atajo para establecer valores de propiedades a partir de los parámetros de la petición, si ambos tienen el mismo nombre.

```
<jsp:setProperty name='user' property='firstName' value='<%=
request.getParameter("fisrtName") %>' />
<%!-- si 'firstName' es un parámetro de la petición --%>
<jsp:setProperty name='user' property='firstName' />
<%!-- todos los parámetros de la petición cuyo nombre coincida con
propiedades --%>
<jsp:setProperty name='user' property='*' />
```

Java Server Pages Tag Libraries

- Permiten declarar nuevas marcas JSP (tags) cuyo código Java se programa en clases de implementación de cada tag:
    - Permite reducir el código Java incrustado en páginas.
    - Útil para acciones utilizadas a menudo (p.e. acceso a bases de datos).
- No las veremos en esta asignatura.

JSTL (java Servlet Tag Libs)

¿Qué es JSTL? (JSP Standard Tag Library)

La librería JSTL 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 libreríasde 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 entrada/salida.
- **xml:** Comprende el procesamiento de xml
- **fmt:** Comprende la internacionalización y formato de valores como de moneda y fechas.
- **sql:** Comprende el acceso a base de datos.

Ventajas

- Debido a que las etiquetas JSTL son XML, estas etiquetas se integran limpia y uniformemente a las etiquetas HTML.
- Las 4 librerías de etiquetas JSTL incluyen la mayoría de funcionalidad que será necesaria en una página JSP. Las etiquetas JSTL son muy sencillas de usarlas para personas que no conocen de programación, a lo mucho necesitará conocimientos de etiquetas del estilo HTML.
- Las etiquetas JSTL encapsulan la lógica como el formato de fechas y números. Usando los scriptlets JSP, esta misma lógica necesitaría ser repetida en todos los sitios donde es usada, o necesitaría ser movida a Clases de ayuda.
- Las etiquetas JSTL pueden referenciar objetos que se encuentren en los ambientes Request y Session sin conocer el tipo del objeto y sin necesidad de hacer el Casting.

- Los JSP EL (Expression Language) facilitan las llamadas a los métodos Get y Set enlos objetos Java. Esto no es posible en la versión JSP 1.2, pero ahora está disponibleen JSP 2.0. EL es usado extensamente en la librería JSTL.

Desventajas

- 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 mesurable pero debe ser considerado.
- Los scriptlets son más potentes que las etiquetas JSTL. Si desea hacer todo en un script JSP pues es muy probable que insertará todo el código en Java en él. A pesarque las etiquetas JSTL proporciona un potente conjunto de librerías reutilizables, no puede hacer todo lo que el código Java puede hacer. La librería JSTL está diseñada para facilitar la codificación en el lado de presentación que es típicamente encontrado en la capa de Vista si hablamos de la arquitectura Modelo-Vista-Controlador.

Utilizando c:out para el despliegue de datos

<c: out> es una etiqueta central JSTL, que se utiliza para mostrar variables del lado del servidor y valores codificados en el navegador (cliente).

Atributos

- **value:** expresión que se tiene que evaluar.

- **escapeXML:** a true (valor por defecto) indica que los caracteres <, >, &, ', " que haya en la cadena resultado se deben convertir a sus códigos correspondientes (<, >, &, ', ", respectivamente).
- **default:** valor por defecto si el resultado es null. Se puede indicar por el atributo o por el cuerpo del tag.

```
<% @ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
 <html> <head> <title> c: out Ejemplo de etiqueta </title> </ head> <body>
<c: out value = "$ {' <b> Este es un <c: out> ejemplo </b> '}" />
 </body> </html>
```

Utilizando c:if y c:choose

C:IF

El tag if permite ejecutar su código si se cumple la condición que contiene su atributo test.

Sintaxis:

Sin cuerpo:

```
<c:if test="condicion" var="variable"
 [scope="page|request|session|application"]/>
```

Con cuerpo:

```
<c:if test="condicion" [var="variable"]
 [scope="page|request|session|application"]>
        Cuerpo
</c:if>
```

Atributos:

- test: condición que debe cumplirse para ejecutar el if.
- var: variable donde se guarda el resultado de evaluar la expresión. El tipo de esta variable debe ser Boolean.
- scope: ámbito de la variable a la que se asigna el valor de la condición.

```
<c:if test="${visitas > 1000}">
<h1>¡Mas de 1000 visitas!</h1>
</c:if>
```

Sacaría el mensaje "¡Mas de 1000 visitas!" si el contador visitas fuese mayor que 1000.

C:CHOOSE

El tag choose permite definir varios bloques de código y ejecutar uno de ellos en funciónde una condición. Dentro del choose puede haber espacios en blanco, una o varias etiquetas when y cero o una etiquetas otherwise.

El funcionamiento es el siguiente: se ejecutará el código de la primera etiqueta when que cumpla la condición de su atributo test. Si ninguna etiqueta when cumple su condición, se ejecutará el código de la etiqueta otherwise (esta etiqueta, si aparece, debe ser la última hija de choose).

SINTAXIS:

```
<c:choose>
        <c:when test="condicion1">
                codigo1
        </c:when>
        <c:when test="condicion2">
                codigo2
        </c:when>
        ...
        <c:when test="condicionN">
                codigoN
        </c:when>
        <c:otherwise>
                codigo
        </c:otherwhise>
</c:choose>
```

EJEMPLO:

```
<c:choose>
        <c:when test="${a < 0}">
        <h1>a menor que 0</h1>
        </c:when>
        <c:when test="${a > 10}">
        <h1>a mayor que 10</h1>
        </c:when>
        <c:otherwise>
        <h1>a entre 1 y 10</h1>
        </c:otherwhise>
</c:choose>
```

Mostrará mensaje "a es menor que 0" si la variable a es menor que 0, el mensaje "a es mayor que 10" si es mayor que 10, y el mensaje "a esta entre 1 y 10" si no se cumple ninguna de las dos anteriores.

Iterando con c:foreach

El tag forEach permite repetir su código recorriendo un conjunto de objetos, o duranteun número determinado de iteraciones.

Sintaxis:

Para iterar sobre un conjunto de objetos:

```
<c:forEach [var="variable"] items="conjunto"
[varStatus="variableEstado"] [begin="comienzo"]
[end="final"] [step="incremento"]>
        codigo
</c:forEach>
```

Para iterar un determinado número de veces:

```
<c:forEach [var="variable"]
 [varStatus="variableEstado"] begin="comienzo"
 end="final" [step="incremento"]>
        codigo
</c:forEach>
```

Atributos:

- **var:** variable donde guardar el elemento actual que se está explorando en la iteración. El tipo de este objeto depende del tipo de conjunto que se esté recorriendo.
- **items:** conjunto de elementos que recorre la iteración. Pueden recorrerse varios tipos:
    - **Array:** tanto de tipos primitivos como de tipos complejos. Para los tipos primitivos, cada dato se convierte en su correspondiente wrapper (Integerpara int, Float para float, etc)
    - **java.util.Collection:** mediante el método iterator() se obtiene el conjunto,que se procesa en el orden que devuelve dicho método.
    - **java.util.Iterator**
    - **java.util.Enumeration**
    - **java.util.Map:** el objeto del atributo var es entonces de tipo Map.Entry, y se obtiene un Set con los mapeos. Llamando al método iterator() del mismo se obtiene el conjunto a recorrer.
    - **String:** la cadena representa un conjunto de valores separados por comas,que se van recorriendo en el orden en que están.
- **varStatus:** variable donde guardar el estado actual de la iteración. Es del tipo javax.servlet.jsp.jstl.core.LoopTagStatus.
- **begin:** indica el valor a partir del cual comenzar la iteración. Si se está recorriendoun conjunto de objetos, indica el índice del primer objeto a explorar (el primero es el 0), y si no, indica el valor inicial del contador. Si se indica este atributo, debeser mayor o igual que 0.
- **end:** indica el valor donde terminar la iteración. Si se está recorriendo un conjuntode objetos, indica el índice del último objeto a explorar (inclusive), y si no, indica el valor final del contador. Si se indica este atributo, debe ser mayor o igual que begin.
- **step:** indica cuántas unidades incrementar el contador cada iteración, para ir de begin a end. Por defecto es 1 unidad. Si se indica este atributo, debe ser mayor o igual que

EJEMPLO:

```
<c:forEach var="item"
 items="${cart.items}">
        <tr>
                <td>
                <c:out value="${item.valor}"/>
                </td>
        </tr>
</c:forEach>
```

Muestra el valor de todos los items.

Utilizando funciones útiles en JSTL

Librería SQL

Los tags de la librería SQL permiten acceder y manipular información de bases de datos relacionales. Vienen definidos con el prefijo "sql".

Con esta librería podremos:

- Establecer la base de datos a la que acceder
- Realizar consultas a bases de datos (select)
- Acceder a los resultados de las consultas realizadas
- Realizar actualizaciones sobre la base de datos (insert, update, delete)
- Agrupar operaciones en una sola transacción

Estas acciones se realizan sobre objetos de tipo javax.sql.DataSource, que proporciona conexiones a la fuente de datos que representa. Así, se obtiene un objeto Connection de dicho DataSource, y con él podremos ejecutar sentencias y obtener resultados. Podemos definir el DataSource mediante la etiqueta setDataSource y luego acceder a ese DataSource con los atributos dataSource de las etiquetas de la librería.

Etiquetas de la librería SQL - Librería SQLetDataSource

El tag setDataSource permite definir el objeto DataSource con el que trabajar, y dejarlo asignado en una variable.

Sintaxis:

```
<sql:setDataSource {dataSource="DataSource" |
 url="url" [driver="driver"] [user="usuario"]
 [password="password"]} [var="variable"]
 [scope="page|request|session|application"]/>
```

Atributos:

- dataSource: objeto DataSource al que queremos enlazar (en caso de que esté creado de antemano).
- url: URL de la base de datos a acceder
- driver: driver con que conectar con la base de datos
- user: nombre de usuario con que conectar a la base de datos
- password: password con que conectar a la base de datos
- var: variable donde guardar el DataSource que se obtenga
- scope: ámbito de la variable var

Notar que se puede obtener el DataSource tanto indicándolo directamente en el atributo dataSource como indicando url, driver, user y password (los tres últimos opcionales).

Query

El tag query permite definir una consulta (select) en una base de datos.Sintaxis:

Sin cuerpo:

```
<sql:query sql="consulta" var="variable"
 [scope="page|request|session|application"]
 [dataSource="DataSource"] [maxRows="max"]
 [startRow="inicio"]/>
```

Con cuerpo donde indicar parámetros de la consulta:

```
<sql:query sql="consulta" var="variable"
 [scope="page|request|session|application"]
 [dataSource="DataSource"] [maxRows="max"]
 [startRow="inicio"]>
       Campos <sql:param>
</sql:query>
```

Atributos:

- **sql:** consulta a realizar (en formato SQL). Puede indicarse la consulta tanto en este atributo como en el cuerpo de la etiqueta.
- **dataSource:** objeto DataSource asociado a la base de datos a la que se accede. Si especificamos este campo, no podemos incluir esta etiqueta dentro de una transacción (etiqueta transaction).
- **maxRows:** máximo número de filas que se devuelven como resultado
- **startRow:** fila a partir de la cual devolver resultados. Por defecto es la 0.
- **var:** variable donde guardar el resultado. Es de tipo javax.servlet.jsp.jstl.sql.Result.
- **scope:** ámbito de la variable var.

Update

El tag update permite definir una actualización (insert, update, delete) en una base de datos.

Sintaxis:

Sin cuerpo:

```
<sql:update sql="actualizacion"
 [dataSource="DataSource"] [var="variable"]
 [scope="page|request|session|application"]/>
```

Con cuerpo donde indicar parámetros de la actualización:

```
<sql:update sql="actualizacion"
 [dataSource="DataSource"] [var="variable"]
 [scope="page|request|session|application"]>
        Campos <sql:param>
</sql:update>
```

Con cuerpo donde indicar la propia actualización y los parámetros de la misma:

```
<sql:update [dataSource="DataSource"] [var="variable"]
 [scope="page|request|session|application"]>
        Actualización
        Campos <sql:param>
</sql:update>
```

Atributos:

- **sql:** actualización a realizar (en formato SQL). Puede indicarse tanto en este atributo como en el cuerpo de la etiqueta.
- **dataSource:** objeto DataSource asociado a la base de datos a la que se accede. Si especificamos este campo, no podemos incluir esta etiqueta dentro de una transacción (etiqueta transaction).
- **var:** variable donde guardar el resultado. Es de tipo Integer (se devuelve el númerode filas afectadas por la actualización).
- **scope:** ámbito de la variable var.

Creando Formularios para capturar información

Creamos una página jsp en la cual crearemos nuestro formulario.

```
<head>
<title>HOLA FORMULARIOS</title>
<style type="text/css" media="screen">
      /*la directiva include copia el contenido de un archivo y lo incrusta en la pa
     <%@ include file="estilo.css" %>
</style>
</head>
<body>

<form action="proceso.jsp" method="post">
    Nombre:
    <input type="text" name="nombre">
    <br/>
    Apellido:
    <input type="text" name="apellido">
    <br/>
    Edad:
    <input type="text" name="edad">

    <br/>
    Lenguaje preferido:
    <select name="lenguaje">
      <option value="java">java
      <option value="jsp" selected>jsp
      <option value="php">php
      <option value="C/C++">C/C++
      <option value="C#">C#
      <option value="Asp">Asp
      <option value="AS3">AS3
    </select>
    <br/>
    Me gusta el:
    <br/>
    <input type="Radio" name="preferencia" value= "Diseño"checked>Diseño
    <br/>
    <input type="Radio" name= "preferencia"value="Programacion">Programacion
    <br/>
    <input type="Radio" name= "preferencia"value="Modelado">Modelado
    <br/>
    <input type="Radio" name= "preferencia"value="Gerencia">Gerencia de proyectos
    <br/>

    <p><input type="submit" value="Enviar"></p>
</form>

</body>
</html>
```

y por ultimo tenemos la pagina que procesa el formulario proceso.jsp

```
<html>
<:head;,
<tí tl e>HOLA I=ORMIJLAR] OS</ ti tl e>
<:style type"" text/css" media'"screél">
      /'la directiva include copia el contendo de un archivo y lo incrusta en la pa
     <%@ illclude file="estilo.css" %>
<:/s tyle>
<:/ head>
<:body;,

<%
   !'"podemos leelí los datos del request a una variable./
   String edad=(Strilg)request.getParameter("edad");
   Stri ng prefieres"(Strilg)request.getPa rameter("preferencia");
   out.prilit("tll nombre es "+request.getParameter("nombre")+" "+request.getParameter
   out.prilit("<br/>");
   out.prilit("tienes "+edad+" afios");
   out.prilit("<br/>");
   out.prilit("tu lenguaje favorito es "+request.getParameter("leélguaje"));
   out.prilit("<br/>");
   out.prilit(•y prefieres el(a) "+prefieres+" de llll proyecto");
   out.prilit("<br/>");
   /..podernos usalí los datos directamente desde el request•/
   out.prilit("Bienvenido a jsp "+ request.getParameter("nombre").toString().toUpperC

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

UNIVERSIDAD ACREDITADA
Gestión Institucional
Docencia de Pregrado
Investigación
Vinculación con el Medio
Por 5 años hasta diciembre 2022
CHE MSA Middle States Commission on Higher Education 3624 Market Street Philadelphia, PA 19104-2680
UNIVERSIDAD ACREDITADA 2015 - 2020
CHEA International Quality Group
UNAB forma parte de la Red LAUREATE INTERNATIONAL UNIVERSITIES
www.unab.cl

Anexo: Referencias

[1] Qué es Java Enterprise (J2ee, JEE)
Referencia: https://www.fundesem.es/bt/publicacion-java-ee-y-el-desarrollo-web-un-enfoque-de-aprendizaje

[2] Qué son los Servlets
Referencia: https://users.dcc.uchile.cl/~jbarrios/servlets/general.html

[3] Utilizando métodos post y get
Referencia: https://www.ecodeup.com/enviar-parametros-servlet-desde-una-vist a-utilizando-post-get/

[4] Sesiones y cookies
Referencia: https://ricardogeek.com/manejo-de-sesiones-y-cookies-en-servlets-java/

[5] Etiquetas JSTL
Referencia: http://www.jtech.ua.es/ayto/ayto2008/jsp/sesion07-apuntes.html