Desarrollo de Aplicaciones JavaEE.

Módulo 4.

Java Server Pages.

(Parte 1)

Objetivos.

- Explicar la función de las JSPs, Java Server Pages.
- Describir los servicios que proporciona el Contenedor Web para las JSPs.
- Listar los beneficios de la tecnología de las JSPs.
- Componentes de las JSPs.
- Desarrollo de JSPs.
- Utilización del *Expression Language (EL)*.
- *Custom Tag Libraries.*
- *JSP Standard Tag Library*

Generalidades.

- Las JSPs son documentos tipo texto que describen como procesar un request de un cliente y crear una respuesta.

- Utilizando JSPs, un *diseñador de páginas* puede crear un documento que genere una respuesta compleja con contenido dinámico.

Generalidades (2).

- Una JSP puede contener:
  - tags de HTML.
  - tags de XML.
  - tags especiales de JSP.

Generalidades (3).

- La especificación de JSPs también define:
  - un lenguaje de script que se puede usar dentro de la página, cuyo default es Java.
  - una serie de acciones, directivas y objetos implícitos.

Nota. Hasta ahora, Java es el único lenguage de Script.

Generalidades (4).

- Se pueden hacer tareas de programación con mínimo esfuerzo de parte del desarrollador.
- Como son archivos tipo texto, se pueden crear con cualquier editor de texto, aunque existen editores especiales.
- Los archivos deben tener la extensión **.jsp.**

Ejemplo de una JSP.

```
<%@ page  import="java.util.*,ProductCatalog.*" %>
<%! Iterator it =null; Vector cpi = null; CatalogItem ci=null; %>
<html><head><title>Shopping Catalog </title></head><body>
<jsp:useBean id="catalogPageData" scope="session" class="MVCApp.CatalogPageData" />
<table>
<%
cpi = catalogPageData.getCatalogPageItems();
if (cpi != null) {
     it = itemList.iterator();
     while (it.hasNext()) {
          ci =(CatalogItem)it.next();
%>
  </tr>
  <td><%= ci.getTitle() %></td>
  <td><form method="get" action="/servlet/MVCApp.Controller">
      <input type="hidden" name="itemId" value="<%= ci.getId() %>">
      <input type="submit" name="addButton" value="Add Item To Cart">
      <input type="hidden" name="action" value="addItemToCart">  
      <input type="text" size ="3"name="quantity" value="1">
      </form></td>
  </tr>
<% } %>
</table></body></html>
```

Las JSPs corren en el Web Container.

- JSPs son componentes web que requieren un contenedor que les proporcione servicios.
- No existe un contenedor especial, usa el mismo que los servlets, el Web Container.
- El soporte para JSPs se implementa mediante un servlet que proporciona servicios a las JSPs.

Las JSPs corren como servlets.

- De hecho, los JSPs son traducidos a servlets y compilados automáticamente por el contenedor.
- Las JSPs corren como servlets dentro del contenedor.
- El contenedor reconoce cuando se modifica una JSP y la recompila automáticamente.

Ventajas de las JSPs.

- Rendimiento.
  - Igual que los servlets en momento de ejecución.
  - Traducción y recompilación automática
  - Proceso del lado del server.
- Programación.
  - Énfasis en componentes reusables.
  - Portabilidad de plataforma.
  - Separación de contenido estático y dinámico.
  - Se extienden fácilmente mediante tag libraries.
  - Lenguaje java con todas sus ventajas.

Partes de las JSPs.

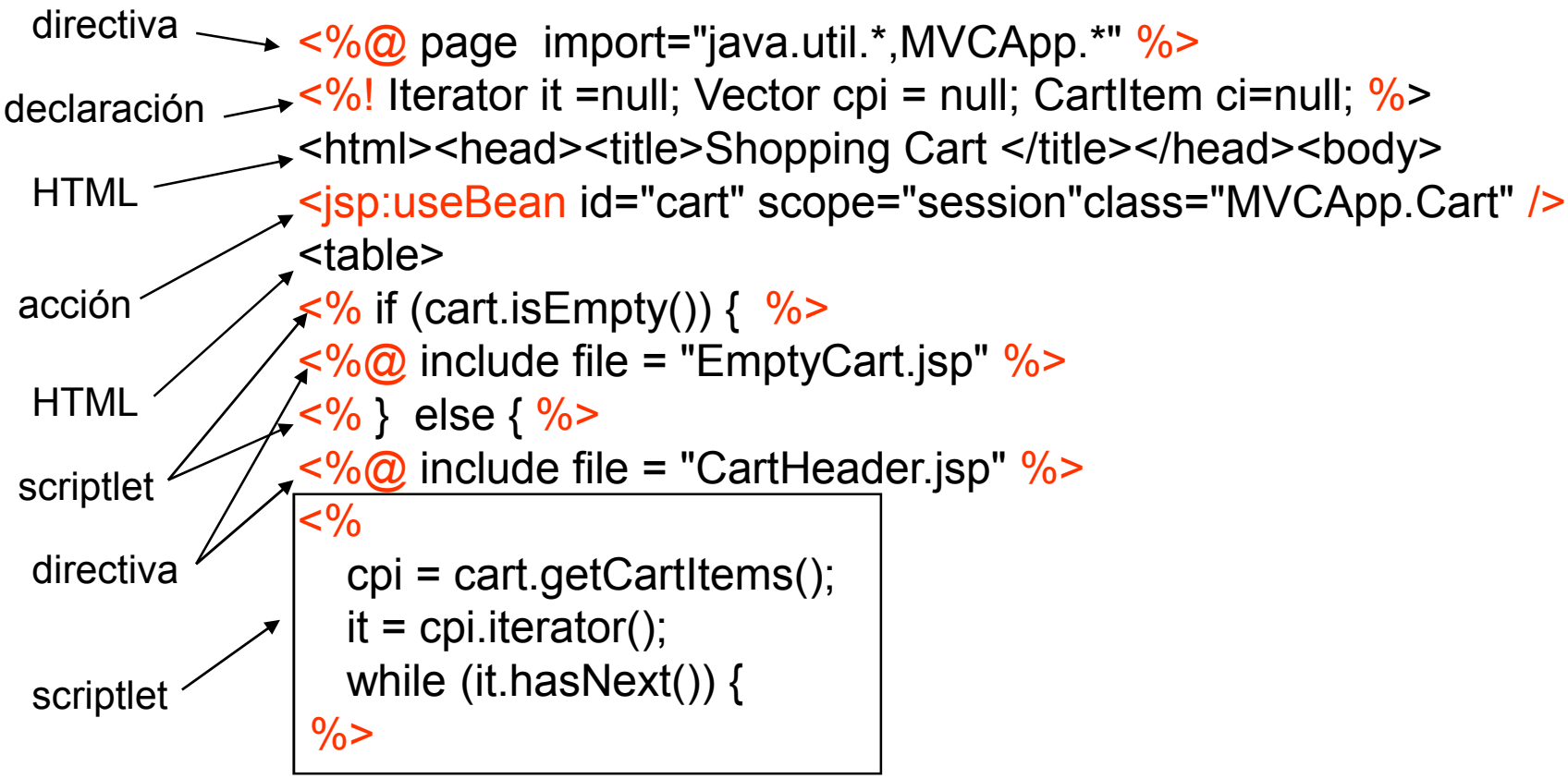
- Directivas: <%@ %>
  <jsp:directive.type> ... </jsp:directive.type>
- Scripting elements:
  - Declaraciones. <%! %>
    <jsp:declaration> ... </jsp:declaration>
  - Expresiones. <%= %>
    <jsp:expression> ... </jsp:expression>
  - Scriptlets. <% %>
    <jsp:scriptlet> ... </jsp:scriptlet>

Partes de las JSPs (2).

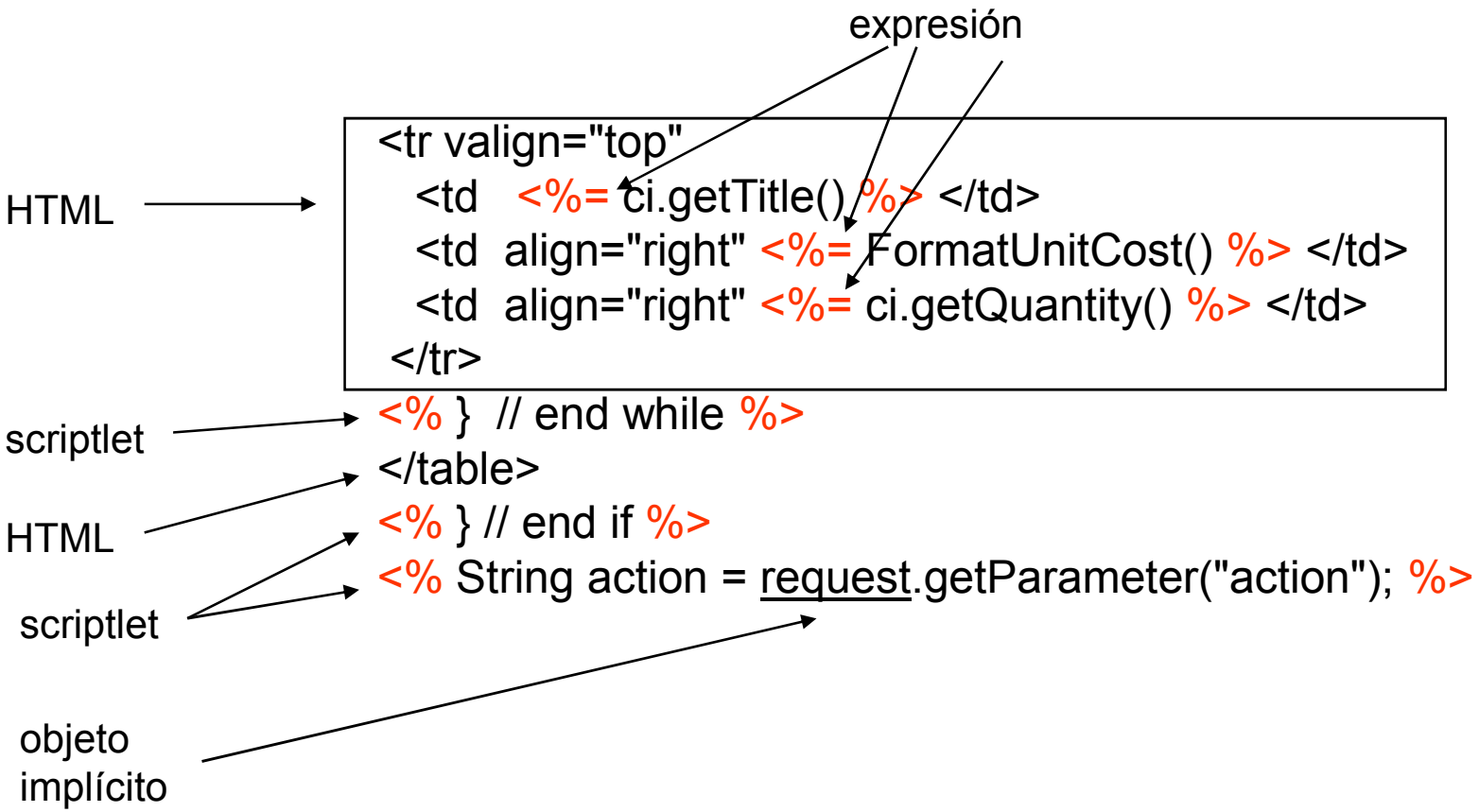
- *ActionTags*. <jsp:*action* />
- Objetos implícitos. request, response, pageContext, out, session, *exception (*)*.
- Elementos HTML o XML.
- Comentarios.

(*) *exception* sólo es válido en páginas de error.

Ejemplo.


directiva
declaración
HTML
acción
HTML
scriptlet
directiva
scriptlet
<%@ page import="java.util.*,MVCApp.*" %>
<%! Iterator it =null; Vector cpi = null; CartItem ci=null; %>
<html><head><title>Shopping Cart </title></head><body>
<jsp:useBean id="cart" scope="session"class="MVCApp.Cart" />
<table>
<% if (cart.isEmpty()) { %>
<%@ include file = "EmptyCart.jsp" %>
<% } else { %>
<%@ include file = "CartHeader.jsp" %>
<%
cpi = cart.getCartItems();
it = cpi.iterator();
while (it.hasNext()) {
%>

Ejemplo (2).


expresión
HTML
<tr valign="top"
<td <%= ci.getTitle() %> </td>
<td align="right" <%= FormatUnitCost() %> </td>
<td align="right" <%= ci.getQuantity() %> </td>
</tr>
scriptlet
<% } // end while %>
</table>
HTML
<% } // end if %>
scriptlet
<% String action = request.getParameter("action"); %>
objeto
implícito

Directivas.

- Contienen información de configuración de la JSP.
- No generan salida.
- Normalmente al principio de la página.
- Sintaxis general:

```
<%@ directiva atributo="valor", ...  %>
<jsp:directive.directiva> ... </jsp:directive.directiva>
```

Tipos de Directivas.

- Tres tipos:
  - page.
  - include.
  - taglib.

La Directiva page.

- Define atributos que afectan a toda la página.
- Atributos:
  - language — lenguaje de script (Java.)
  - session — especifica si la página participa en una sesión (true o false)
  - buffer — stream para objeto out. (8kb)
  - errorPage — define una URI para página de error. (/error.jsp)

Nota. Subrayado indica default.

La Directiva page (2).

• import	lista los paquetes accesibles. (Genera postulados import de Java.)
• isErrorPage	especifica si la página es página de error de otra. (true o false)
• isThreadSafe	especifica el modelo de Thread. (true o false)
• contentType	formato de los datos. (text/html)

Notas. Subrayado indica default, isThreadSafe está descontinuada.

La Directiva page (3).

- info — String de información para identificar la JSP.
- autoFlush — determina que hace cuando se llena el buffer. (true o false)

- Ejemplo

```
<%@ page  import="java.util.*","java.text.*" %>
```

Nota. Subrayado indica default.

La Directiva include.

- Incluye el recurso mencionado en la JSP.
- Puede ser HTML u otra JSP.
- Resuelto en el momento de la traducción a servlet (estático).
- El texto se incluye en el lugar de la directiva.
- Por ejemplo:

```
<%@ include file="header.jsp" %>
```

La Directiva taglib.

- Extiende el conjunto de tags que interpreta el contenedor.
- Asocia un prefijo con una biblioteca de tags.
- Una biblioteca de tags es un conjunto de programas Java que contienen métodos asociados con los tags de la JSP.
- Por ejemplo:

```
<%@ taglib uri="TagLibrary" prefix="pref" %>
```

Nota. Las taglibs se estudian en detalle más adelante.

Scripting elements.

- Tres tipos:
  - Declaraciones.
  - Expresiones.
  - Scriptlets.

Declaraciones.

- Variables y métodos.
- Sintaxis del lenguaje Java.
- Postulados declarativos de Java.
- No generan salida, se incluyen a nivel clase en el servlet generado.
- Sintaxis general:

```
<%! declaración ... %>
<jsp:declaration>…</jsp:declaration>
```

Declaraciones (2).

- Sintaxis para declarar variables:

  <%! atributos Clase nombre = postuladoJava; %>

- Por ejemplo:

  <%! private String pattern = "dd/mm/yyyy"; %>

Declaraciones (3).

- Sintaxis para declarar métodos:

```
<%! firmaMétodo { cuerpo del método }  %>
```

- Por ejemplo:

```
<%!
  private boolean isEmptyCart() {
      return cart.isEmpty();
  }
%>
```

Expresiones.

- Inserta el valor de una expresión de Java en el resultado de la JSP (como parte de un postulado out.print del método “processRequest”).
- Evaluada en el momento de ejecución.
- Convierte automáticamente el resultado a String y lo inserta en el stream de salida (out.println).
- Generalmente obtiene valores dinámicos para ser incluídos en el HTML generado.

Expresiones (2).

- Sintaxis.

  <%= expresión %>

- Por ejemplo:

```
<tr valign="top">
 <td> <%= ci.getTitle() %>  </td>
 <td> <%= ci.getFormattedUnitCost() %>  </td>
 <td> <%= ci.getQuantity() %>  </td>
</tr>
```

Scriptlets.

- Generan contenido dinámico dentro de la JSP; el código java se incluye en el método “processRequest” del servlet generado.
- Cualquier fragmento de código válido en Java.
- Acceso completo a la API de Java.
- Para incluir otros paquetes, se usa la directiva import.

Scriptlets (2).

- Sintaxis.

<% fragmento de código %>

- Por ejemplo:

```
<%
  if (request.getParameter("username") == null) {
       url = ScreenMgr.LOGIN_ERRORPAGE;
  } else {
      account = getAcctData(
                  request.getParameter("username"));
  }
%>
```

Action Tags.

- Proporcionan funcionalidad adicional para el desarrollador de la JSP.
- Usan tags estilo XML.
- Parejas atributo/valor.
- Se pueden crear otros usando tag libraries.
- *Pueden* modificar el output stream.
- Usan, modifican o crean objetos.
- También se conocen como *standard tags*.

Action Tags (2).

Principales action *tags*:

- jsp:useBean
- jsp:getProperty
- jsp:setProperty
- jsp:include
- jsp:forward
- jsp:param

<jsp:useBean>

- Para crear o utilizar una instancia de un Java Bean.
- Accesible dentro de un scope definido.
- Si no existe un objeto con la misma variable de referencia, se instancia.
- Si ya existe, se vuelve a usar la variable.

<jsp:useBean> (2)

- Sintaxis 1:

```
<jsp:useBean id="nomVar" scope="scope" class = "clase" />
```

- Sintaxis 2:

```
<jsp:useBean id="nomVar" scope="scope" class = "clase" >
<% código para inicialización  %>
</jsp:useBean>
```

<jsp:useBean> (3)

- scope puede ser:
  - page
  - request
  - session
  - application

<jsp:useBean> (4)

- Por ejemplo

```
<jsp:useBean id="account" scope="session" class="Account" >
    <% account.init(request.getSession()); %>
</jsp:useBean>
```

<jsp:getProperty>

- Obtiene el valor de una propiedad de un Bean definido previamente con <jsp:useBean >
- Sintaxis

```
<jsp:getProperty name="nomBean"  property="pName" />
```

- Ejemplo:

```
<jsp:getProperty name="account"  property="balance" />
```

<jsp:setProperty>

- Establece el valor de una propiedad de un Bean definido previamente con <jsp:useBean >
- Sintaxis

<jsp:setProperty name="nomBean" expresión />

donde expresión puede ser:

property="propName" [param="paramName" | value="propValue"]

property="*"

Notas. Si se omiten ambos param y value, se asume que el nombre de la propiedad es el mismo que el nombre del parámetro.
Si se especifica property = "*" todas las propiedades del bean adquieren los valores de los parámetros del request.

<jsp:setProperty> (2).

- Ejemplos.

```
<jsp:setProperty name="account"  property = "balance" />
<jsp:setProperty name="account"  property = "balance" value="1000" />
<jsp:setProperty name="account"  property = "*" />
<jsp:setProperty name="account"  property = "balance" param ="saldo" />
```

<jsp:include> y <jsp:param>

- Incluye recursos en el momento de ejecución de la JSP (dinámicamente).

- Sintaxis 1:

```
<jsp:include page="url" />
```

- Sintaxis 2:

```
<jsp:include page="url" >
    <jsp:param name="nombre" value="valor" />
</jsp:include>
```

<jsp:forward> y <jsp:param>

- Pasa el control a otro recurso en forma incondicional.
- Sintaxis 1:

```
<jsp:forward page="url" />
```

- Sintaxis 2:

```
<jsp:forward page="url" />
    <jsp:param name="nombre" value="valor" />
</jsp:forward>
```

Ejemplo.

```
<jsp:include page="/banner.jsp">
    <jsp:param name="subTitle" value="Thank You!"/>
</jsp:include>
```

```
<%@ page import="domain.League" %>
<%
String bannerTitle = "Duke's Soccer League";
String subTitle = request.getParameter("subTitle");
%>
```

Otros Action Tags.

- jsp:plugin — genera información para Applets.
- jsp:fallback — genera texto alterno si no hay soporte Java en el browser (usado con jsp:plugin).
- jsp:text — transmite datos de patrones.
- jsp:output — modifica las propiedades del objeto output.
- jsp:root — especifica el elemento raíz de una JSP.

Valores dinámicos en atributos.

- Normalmente los valores de los atributos de los *action tags* son constantes y se resuelven en el momento de la traducción a servlet.
- Ciertos valores pueden ser especificados mediante expresiones.
- Se llaman *Request Time Attribute Values*
- Son los siguientes:
  - name y value de <jsp:setProperty>
  - page de <jsp:include> y <jsp:forward>
  - value de <jsp:param>

Ejemplo.

```
<jsp:forward page="<%=ScreenMgr.getPage(action)%>" >
```

Objetos Implícitos.

- Variables predefinidas que pueden ser usadas en la JSP.
- Siempre disponibles.
- Los principales son:
  - request — objeto de clase HttpServletRequest que apunta al request recibido.
  - response — objeto de clase HttpServletResponse que apunta al response recibido.

Objetos Implícitos (2).

- pageContext — El contexto de la página, contiene métodos para obtener información de la página.
- session — El objeto clase session, instancia de HttpSession.
- out — La variable de referencia al objeto clase PrintWriter,usado para escribir la respuesta.
- exception — El objeto excepción producido en otra página. (Sólo para páginas de manejo de errores).

Comentarios.

<!-- Comentario estilo HTML/XML, se transmite a la salida de la JSP -->

<%-- Comentario estilo JSP, NO se transmite a la salida de la JSP --%>

Nota. también se pueden usar comentarios de java en scriptlets y declaraciones.

Manejo de errores.

```
<html>
<head><title>Today.jsp</title></head>
<body>
<%@ page import="java.util.*", "java.text.*" %>
<%@ page errorPage="TodayErrorPage.jsp"  %>
<%! public String format(String pattern) {
    SimpleDateFormat sdf = new SimpleDateFormat(pattern);
     return sdf.format(new Date());
}  %>
<% String pattern = request.getParameter("pattern"); %>
<p>Hoy es <%= format(pattern); %></p>
</body>
</html>
```

Manejo de errores (2).

```
<html>
<head><title>TodayErrorPage.jsp</title></head>
<body>
<%@ page isErrorPage="true" %>
<h1>Error</h1>
<p>Exception: <%= exception %>
</body>
</html>
```

Laboratorio 4.

Desarrollar la JSP allCustomers.jsp de la aplicación BrokerTool.