Sesiones

Administración de Sesiones

Que es una sesión

La sesión representa una forma de identificar a un cliente en particular del lado del servidor, y poder mantener un conjunto de valores determinados únicamente para ese cliente a lo largo de un periodo de tiempo. Técnicamente, es una espacio de memoria que reserva el servlet container para un cliente, identificando este espacio con un identificador de sesión o sessionID.

Las sesiones se utilizan típicamente en logins de usuarios, sitios de e-commerce donde resulta necesario construir un carrito de compras, y también en bancos que proveen el servicio de Home Banking.

El sessionID

De acuerdo a lo comentado anteriormente, este identificador sirve para identificar unívocamente a un cliente. Tanto su creación como su destrucción está a cargo del servlet container.

El sessionID es una cadena de caracteres sumamente larga, que combina números y letras, e identifica unívocamente a un cliente dentro del “universo” de clientes.

Por su parte, el protocolo HTTP es “stateless” o sin estado, con lo cual para que el servidor pueda determinar quien es el cliente, necesita obtener algún identificador. El identificador es precisamente el sessionID, que viaja permanentemente entre el cliente y servidor de forma transparente para usuarios finales y también para desarrolladores. Gracias al sessionID, el servidor podrá brindarle un “trato personalizado” al cliente.

El objeto session

El objeto session es uno de los objetos implícitos instanciados por el Web container en tiempo de ejecución, y modela la sesión de un usuario particular. En este objeto es posible almacenar información para luego ser recuperada. Técnicamente el objeto session es una instancia de una clase que implementa la interfaz HttpSession.

Es posible almacenar los valores que resulten necesarios utilizando el método setAttribute()

La forma de almacenar un atributo es la siguiente:

session.setAttribute(

“nombre_del_atributo”, objeto_con_el_valor);

Por ejemplo, se podría guardar un objeto del tipo Usuario como sesión:

session.setAttribute(“user”, elUsuario);

Para obtener un valor de sesión, es necesario utiliza el método getAttribute()

Usuario unUsuario =

(Usuario) session.getAttribute(“user”);

Session timeout

El session timeout determina el tiempo que dura la sesión sin que el usuario interactúe con el servidor

El Web container controla el último pedido de un cliente con el servidor

Luego de cumplido este tiempo sin interacción por parte del usuario, la sesión es destruida por el Web container

Para establecer el session-timeout es necesario agregar las siguientes líneas dentro del descriptor de despliegue:

```
<session-config>
    <session-timeout>10</session-timeout>
</session-config>
```

El tiempo del timeout es expresado en minutos

JavaBean como session

Es posible utilizar JavaBeans como sesión

Simplemente hay que establecerle al bean el ámbito (scope)

La forma de establecerlo es la siguiente:

```
<jsp:useBean id=”miBean” class=”paquete.Clase”
scope=”session” />
```

Destrucción de una sesión

Una vez terminado el uso de los valores de sesión, la sesión debe ser destruida

Un mecanismo de destrucción es por tiempo, presentado anteriormente con el session-timeout

Para eliminar una sesión en forma manual, se deberá utilizar el método invalidate()

El modo de utilización es el siguiente:

session.invalidate();

URL rewriting

A veces resulta necesario reescribir la url con información adicional cuando el browser trabaja sin las cookies habilitadas. En caso de que el browser acepte cookies, al crearse una sesión, el servlet container le envía al browser una cookie con el identificador sesion, denominado jsessionid. En caso de que el browser NO acepte cookies, o estén deshabilitadas, es necesario modificar todas las urls para que transporten el identificador de la sesión.

Un ejemplo de envio del identificador de sesión es el siguiente:

../destino.jsp;jsessionid=aa363bcaa3af23bac8f4999

Cuando el servidor recibe un pedido HTTP obtiene el valor del jsessionid para asociar el pedido con la sesión correspondiente al usuario.

Para trabajar de forma automática es necesario modificar manualmente todos las urls que formen parte de la aplicación, tanto las pertenecientes a redirecciones como las de links. Afortunadamente, esto se realiza de manera automática con la utilización de ciertos métodos que forman parte del objeto response, estos son el método response.encodeUrl() y el método response.encodeRedirectUrl()

El método response.encodeUrl() se utiliza en todos los links del portal. La forma de utilizarlo es la siguiente:

<a target="_blank" href=<%=response.encodeUrl(“destino.jsp”)%> >

Destino del link

El método response.encodeRedirectUrl() se utiliza en todas las redirecciones realizadas en los archivos jsp. La forma de utilizarlo es la siguiente:

<%

response.sendRedirect(

response.encodeRedirectUrl(“destino.jsp”));

%>

JSTL – JavaServerPages Standard Tag Library

Introducción

Que son las tag libraries

Es una tecnología que representa un componente dentro de la especificación de J2EE

Encapsulan funcionalidad que es necesaria para escribir archivos JSP

En su concepción, son librerías que contienen tags JSP agrupados por funcionalidad

Tienen el objetivo de reemplazar el scripting, aumentando reutilización y simplificando los JSPs

JSTL es una única tag library representada por un archivo .jar

Pueden ser creadas por el usuario

El detalle de las librerías puede encontrarse en el siguiente link:

http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/index.html

Utilización

Los tags están basados en la sintaxis de xml

Existen 2 tipos de tags, los que tienen cuerpo y los que no

La forma de uso general es:

<prefijo:nombreTag>

“prefijo” es el valor utilizado en prefix en la importación

“nombreTag” es un tag perteneciente a la tag-lib importada

La forma de utilizarlos tags sin cuerpo es:

<prefijo:nombreTag />

La forma de utilizarlos tags con cuerpo es:

<prefijo:nombreTag>

    aqui el cuerpo

</prefijo:nombreTag>

Organización

Dentro de JSTL existen distintas acciones (tags) organizadas según áreas de funcionalidad

Las áreas de funcionalidad pre-existentes dentro de JSTL son Core, XML, Internationalization, SQL y Functions

Core

Definición

Es el área de funcionalidad que contiene los tags necesarios para las funciones básicas de scripting, como ser: loops, condicionales, impresiones en pantalla, etc.

Utilización

El prefijo utilizado es “c”

La forma de importar la librería es:

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

El tag out

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

<html>

<body>

<c:out value="Hola mundo, soy el tag out!" />

</body>

</html>

El tag set

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


<html>
<body>
<%-- Setea una variable --%>
<c:set var="miVariable" scope="request"
value="soy el valor" />

<%-- Imprime el valor miVariable --%>
<c:out value="miVariable" />

<%-- Imprime el contenido de miVariable --%>
<c:out value="${miVariable}" />
</body>
</html>
```

El tag remove

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


<html>
<body>
<%-- Setea una variable --%>
<c:set var="miVariable" scope="request"
value="soy el valor" />

<%-- Imprime el contenido de miVariable --%>
<c:out value="${miVariable}" />
```

```
<%-- Remueve la variable --%>
<c:remove var="miVariable" scope="request" />
<%-- Imprime el contenido de miVariable --%>
<c:out value="${miVariable}" />
</body>
</html>
```

El tag if

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

<html>
<body>
<%-- Define variables --%>
<c:set var="vacio" scope="request" value="" />
<c:set var="miVariable" scope="request"
value="100" />
<c:if test="${empty vacio}">
<c:out value="La variable <B>vacio</B>
esta vacía!" escapeXml="" />
</c:if>
<c:if test="${!empty vacio}">
<c:out value="La variable <B>vacio</B>
NO esta vacía!" escapeXml="" />
</c:if>

<c:if test="${ miVariable == '100'}">
<c:out value="miVariable es igual a 100" />
```

```
</c:if>
<c:if test="${!(miVariable == '100')}">
<c:out value="miVariable NO es igual a 100" />
</c:if>
</body>
</html>
```

El tag choose

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

<html>
<body>
<%-- Define variables --%>
<c:set var="categoria" value="C" scope="request" />
<%-- Utiliza el choose --%>
<c:choose>
<c:when test="${categoria == 'A'}">
<c:out value="Es la categoría A" />
</c:when>
<c:when test="${categoria == 'B'}">
<c:out value="Es la categoría B" />
</c:when>
<c:when test="${categoria == 'C'}">
<c:out value="Es la categoría C" />
</c:when>
<c:otherwise>
<c:out value="No es una categoria" />
```

```
</c:otherwise>
</c:choose>
</body>
</html>
```

El tag forEach

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

<html>
<body>
<%-- Itera desde 10 al 20 saltando de a dos --%>
<c:forEach var="i" begin="10" end="20" step="2">
<c:out value="${i}"/><BR>
</c:forEach>
<%-- Itera del 10 al 20 saltando index y count--%>
<c:forEach var="i" begin="10" end="20"
varStatus="status">
indice: <c:out value="${status.index}"/>
iteracion #:
<c:out value="${status.count}"/> <BR>
</c:forEach>
<%-- Itera por los items de una coleccion --%>
<jsp:useBean id="empresaBean" scope="request"
class="nombreBean" />
<c:forEach var="e"
items="${nombreBean.atributoColeccion}">
<c:out value="${e.nombre}" /> |
```

```
<c:out value="${e.apellido}" /><BR>
</c:forEach>
</body>
</html>
```

El valor param

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

<html>
<body>
Para ejecutar este ejemplo, ingrese lo siguiente como
url:<br>
<b><i>param.jsp?saludar=true&nombre=Juan</i></b>

<c:if test="${param.saludar}">
<h1>Hola ${param.nombre}!</H1>
</c:if>
</body>
</html>
```

XML

Definición

Es el área de funcionalidad que contiene los tags necesarios para realizar procesamiento de xml

Utilización

El prefijo utilizado es “x”

La forma de importar la librería es:

<%@taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

Internationalization & Formatting

Definición

Es el área de funcionalidad que contiene los tags necesarios para formateo de valores (como decimales, dinero, fechas, etc) según regiones

Utilización

El prefijo utilizado es “fmt”

La forma de importar la librería es:

<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

El tag formatDate

```
<html>
<body>
<jsp:useBean id="now" class="java.util.Date" />
Fecha: <fmt:formatDate value="${now}"
dateStyle="full" />
<br>
Fecha: <fmt:formatDate value="${now}"
pattern="EEE, MMM d, yy" />
    </body>
</html>
```

Los valores posibles a utilizar en “pattern” son los descriptos dentro de la clase SimpleDateFormat

SQL

Definición

Es el área de funcionalidad que contiene los tags necesarios para realizar el acceso a datos

Utilización

El prefijo utilizado es “sql”

La forma de importar la librería es:

<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

El tag setDataSource

<sql:setDataSource var="fuente"

url="jdbc:mysql://localhost:3306/j2ee-web" driver="com.mysql.jdbc.Driver"

user="root" password=""/>

El tag query

<sql:query var="consulta" dataSource="${fuente}">

SELECT * FROM alumnos

</sql:query>

El tag param

<sql:update var="updateCount" dataSource="${fuente}">

UPDATE alumnos SET nombre=?

<sql:param value="Pepe"/>

WHERE nameid=1

</sql:update>

Functions

Definición

Contiene los tags necesarios para manipular cadenas de caracteres y obtener longitud de colecciones

Utilización

El prefijo utilizado es “fn”

La forma de importar la librería es:

<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/fn" %>