JSTL

SQL Tag Library

Esta cuarta librería JSTL permite interactuar con bases de datos relacionales.
Está recomendada para prototipos rápidos o aplicaciones pequeñas. El empleo de acceso directo a la base de datos desde la capa de presentación es altamente desaconsejado en sistemas en producción o en aplicaciones medianas o grandes. El beneficio de mantener separada la lógica de negocio y acceso a los datos de la presentación se ve reflejado en un mantenimiento más sencillo y una mayor adaptabilidad de la aplicación a cambios futuros.

Fuente de datos

Las fuentes de datos (datasources) son fábricas para la obtención de conexiones a bases de datos. Los servidores de aplicaciones con Java 2 Enterprise Edition (J2EE) suelen proporcionar compatibilidad integrada para fuentes de datos que son puestas a disposición de las aplicaciones J2EE a través de Java Naming y Directory Interface (JNDI).
Las etiquetas de la librería SQL confían en estas fuentes para obtener conexión. Varias etiquetas incluyen un atributo opcional "dataSource" para especificar explícitamente la conexión que puede ser una instancia de la interfaz javax.sql.DataSource o con un nombre JNDI.

Para obtener una instancia de la clase javax.sql.Datasource se utiliza la etiqueta <sql:setDataSource>, que tiene dos posibles sintaxis:

Sintaxis 1
<sql:setDataSource dataSource="expression" var="name" scope="scope"/>

Sintaxis 2
<sql:setDataSource url="expression" driver="expression" user="expression" password="expression" var="name" scope="scope"/>

En la sintaxis 1, el atributo "dataSource" es obligatorio. Para acceder a una fuente de datos se utiliza un nombre JNDI.

Mientras que en la sintaxis 2, el atributo "url" es obligatorio, donde debe especificarse una URL JDBC. A su vez, el atributo "driver" indica el nombre de la clase que implementa el controlador de base de datos, mientras que los atributos "user" y "password" proporcionan la credencial para el acceso a la base de datos.

Para cualquiera de las sintaxis de la etiqueta <sql:setDataSource>, son válidos los atributos "var" y "scope" cuyo significado es igual a todas las etiquetas hasta aquí estudiadas.

Consultas y actualizaciones

Una vez establecida la conexión, se puede utilizar la etiqueta <sql:query> para ejecutar consultas mientras que las actualizaciones se realizan mediante la etiqueta <sql:update>. Las consultas y actualizaciones se especifican como cualquier sentencia SQL, que puede ser parametrizada utilizando la interfaz java.sql.PreparedStatement JDBC. Para asignar los valores de los parámetros se utilizan las etiquetas <sql:param> y <sql:dateParam>.

Existen tres posibles sintaxis para la etiqueta <sql:query>:

Sintaxis 1
<sql:query sql="expression" dataSource="expression" var="name" scope="scope" maxRows="expression" startRow="expression"/>

Sintaxis 2
```
<sql:query sql="expression" dataSource="expression" var="name" scope="scope"
maxRows="expression" startRow="expression">
  <sql:param value="expression"/>
  ...
</sql:query>
```

Sintaxis 3
```
<sql:query dataSource="expression" var="name" scope="scope" maxRows="expression"
startRow="expression">
  SQL statement
  <sql:param value="expression"/>
  ...
</sql:query>
```

Tanto en la sintaxis 1 y 2, los atributos "sql" y "var" son obligatorios, mientras que en la sintaxis 3 solo el atributo "var" es obligatorio.

Los atributos "var" y "scope" permiten almacenar el resultado de la consulta bajo un determinado alcance. El atributo "maxRows" se utiliza para limitar la cantidad de filas devueltas por la consulta, mientras que el atributo "startRow" permite definir desde qué fila se comenzará a construir el resultado ignorando las filas anteriores.
La variable generada es una instancia de la interfaz javax.servlet.jsp.jstl.sql.Result; este objeto proporciona las propiedades para el acceso a las filas, los nombres de las columnas, etc. A continuación se presenta un resumen de cada propiedad:

Propiedad	Definición
rows	Un vector de objetos SortedMap donde cada uno de ellos es un map de nombres de columna de una fila del resultado
rowsByIndex	Una matriz donde cada vector se corresponde con una fila del resultado
columnNames	Un vector con los nombres de columna del resultado basándose en el mismo orden de la propiedad rowsByIndex
rowCount	El número total de filas del resultado de la consulta
limitedByMaxRows	Su valor es verdadero si la consulta fue limitada por el atributo "maxRows"

Dentro de una etiqueta <sql:query> se puede especificar la sentencia SQL en el cuerpo de la misma o en el atributo "sql". La instrucción SQL puede ser parametrizada con el caracter "?". Para cada parámetro, debe haber una etiqueta <sql:param> o <sql:dateParam> que deben estar incluidas en el cuerpo de la etiqueta <sql:query>.
La etiqueta <sql:param> contiene un único atributo "value" que sirve para especificar el valor del parámetro. Cuando el valor del parámetro sea una cadena de caracteres, se puede omitir el atributo "value" y proporcionar dicho valor en el cuerpo de la etiqueta.
Para especificar parámetros que sean fechas, horas o ambas, se utiliza la etiqueta <sql:dateParam> con la siguiente sintaxis:

```
<sql:dateParam value="expression" type="type"/>
```

Siendo el atributo "value" una instancia de la clase java.util.Date, mientras que el atributo "type" debe ser "date", "time" o "timestamp".

Al igual que la etiqueta <sql:query>, la etiqueta <sql:update> presenta tres posibles sintaxis:

Sintaxis 1
```
<sql:update sql="expression" dataSource="expression" var="name" scope="scope"/>
```

Sintaxis 2
```
<sql:update sql="expression" dataSource="expression" var="name" scope="scope">
  <sql:param value="expression"/>
  ...
</sql:update>
```

Sintaxis 3
```
<sql:update dataSource="expression" var="name" scope="scope">
  SQL statement
  <sql:param value="expression"/>
  ...
</sql:update>
```

Tanto el atributo "sql" como "dataSource" tienen la misma semántica que la etiqueta <sql:query>. Pero en este caso, la variable que se genera bajo un determinado alcance, es una instancia de la clase java.lang.Integer que contiene la cantidad de filas afectadas por la ejecución de la actualización.

Transacciones

Se utiliza la etiqueta <sql:transaction> que presenta la siguiente sintaxis:

```
<sql:transaction dataSource="expression" isolation="isolationLevel">
  <sql:query .../> or <sql:update .../>
  ...
</sql:transaction>
```

Ningún atributo es obligatorio. El atributo "isolation" se utiliza para especificar el nivel de aislamiento de la transacción que puede ser: "read_committed", "read_uncommitted", "repeatable_read" o "serializable".

Como era de esperarse, todas las consultas y actualizaciones deben utilizar el mismo "dataSource", por lo que este atributo no puede ser especificado en las etiquetas <sql:query> y <sql:update> si están dentro de la etiqueta <sql:transaction>.

Veamos un ejemplo completo,

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSTL SQL</title>
    <link rel="stylesheet" type="text/css" href="./css/style.css"/>
</head>
<body>
    <h1>Herramienta SQL</h1>
    <form name="sql" action="index.jsp" method="post" target="_top">
       <textarea name="queries" rows="10" cols="109"></textarea>
       <br/><br/>
       <input type="submit" value="Ejecutar"/>
    </form>
    <br/>
    <c:set var="queries" value="${param.queries}" scope="request"/>
    <c:catch var="exception">
       <c:if test="${!empty queries}">
          <hr/>
          <sql:setDataSource var="dataSource" driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
url="jdbc:sqlserver://localhost;databaseName=das;" user="sa" password="pyxis" />
          <c:forTokens delims=";" items="${queries}" var="query">
             <c:choose>
                <c:when test="${fn:indexOf(fn:toLowerCase(query), 'select') >= 0}">
                  <sql:query dataSource="${dataSource}" var="result">
                     ${query}
                  </sql:query>

                  <c:choose>
                     <c:when test="${result.rowCount == 0}">
                        <p>La consulta no retorn&oacute; filas</p>
                     </c:when>
                     <c:otherwise>
                        <table>
                          <thead>
                              <tr>
                              <c:forEach items="${result.columnNames}" var="col">
                                  <td>${col}</td>
                              </c:forEach>
                              </tr>
                          </thead>
                          <tbody>
                          <c:forEach items="${result.rowsByIndex}" var="row">
                              <tr>
                                  <c:forEach items="${row}" var="value">
                                     <td>${value}</td>
                                  </c:forEach>
                              </tr>
                           </c:forEach>
                          </tbody>
                       </table>
                     </c:otherwise>
                  </c:choose>
                </c:when>
                <c:otherwise>
                  <sql:transaction dataSource="${dataSource}" isolation="read_committed">
                     <sql:update var="updateCount">
```

```
                    ${query}
                  </sql:update>
                </sql:transaction>

                  <c:choose>
                    <c:when test="${updateCount > 0}">
                      <p>${updateCount} fila(s) afectada(s)</p>
                    </c:when>
                    <c:otherwise>
                      <p>El comando se ejecut&oacute; exitosamente</p>
                    </c:otherwise>
                  </c:choose>
                </c:otherwise>
              </c:choose>
            </c:forTokens>
          </c:if>
        </c:catch>
        <c:if test="${!empty exception}">
          <p>Error: ${exception}</p>
        </c:if>
    </body>
    </html>
```