用 JSTL 构建纯标记电子商务网站

The Development of Electronic Commerce Website in Script-free Environment with JSTL

摘　要：JSP技术是J2EE的重要组成部分，而由Apache组织推出的JSTL，是JSP与XML技术结合的标准Tag Library。本文简要指出了JSTL在构建电子商务网站的优势,对基于JSTL编程与传统JSP动态网页编程的比较，着重探讨了JSTL在动态网页开发中常用的标记。

关键词：JSTL　Tag Library　标记　动态网页

林志伟　许　力（福州福建师范大学 计算机科学系　350007）

① 福建省教委资助项目：JB02150

1 JSTL简介

XML(eXtensible Markup Language)技术在电子商务网站开发中的应用日益广泛，它作为一种通用的数据描述语言，可以实现数据交换、定义文档类型等功能。将XML与JSP(Java Server Pages)技术结合的JSP Tag Library，应用于动态网页的设计，可以实现更强的功能，它把商业逻辑与页面设计完全分离，实现代码的可重用，提高网页开发的效率。

1.1 目的

电子商务网站大多用ASP、PHP或传统JSP编程等来达到动态显示和交互的效果，通过在HTML代码中插入包含在<%...%>的代码来实现。然而，由于电子商务网站规模，这样的开发方式存在效率低，容易出错，不易维护，难于扩充等缺点。比如在修改某个功能时误删了<%或包含在内的代码而导致整个页面出错。JSTL的设计完全遵循了Tag Library的规范和原则，是一组封装功能的行为集合，根本目的在于设计一套标准通用的标记，使动态网页设计更加简单。使用JSTL完全可以避免上述错误的发生，JSTL把包含在<%...%>中的scripting彻底清除出JSP，使得动态网站开发的代码纯标记，如变量的定义和提取、循环判断、数据库连接等动态操作都在<....>和</...>的标记中完成，与HTML语言基本相同，使页面设计者更专注于页面的设计。

JSTL（JSP Standard Tag Library）是通用的、标准的JSP Tag Library，是apache软件基金会设立了Jakarta项目的重要组成部分，它是JCP(Java Community Process)制定的开放的规范。打个比方，如果HTML是W3C制定的网页开发语言，那么JSTL就是JCP制定的用于动态网页设计的标记语言。

1.2 构成

用JSTL设计的动态网页的运行需要Tomcat4.0的支持，它包含了几个重要部分，如下表：

功能域	URL	prefix
Core	http://java.sun.com/jstl/core	C
relational db access (SQL)	http://java.sun.com/jstl/sql	sql
XML processing	http://java.sun.com/jstl/xml。	X
I18N capable formatting	http://java.sun.com/jstl/fmt	fmt

核心（core）：实现页面的流程控制等操作。

数据库访问(relational db access)：通过JSTL标记实现对数据库的查询等标准SQL操作。

XML解析(XML processing)：调用Java API for XML Processing对XML文档进行解析处理。

国际化格式(I18N capable formatting)：可以使开发的web应用适应不同国家语言的格式。

本文从笔者开发实践中提取动态网页开发中JSTL常用的技术，通过与传统JSP编程的比较，着重介绍在动态网站开发中如何常用的JSTL标记。

2 流程控制

程序设计中最常碰到的就是变量的定义，条件的判断，JSTL提供了流程控制的标记。为了在设计的动态页面中使用JSTL提供的标记，必须在JSP程序中加入<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>以定义一个前缀c，后面将使用这个前缀来实现流程控制的标记。

2.1 数据提取

在网页的设计中，常常需要将表单<form> ....</form>中的数据传到另外一个页面中去，传统的JSP编程是在<%%>中加入脚本request.

getParameter()来逐个获取。如常见的用户名和密码输入框：

```
<form action="in.jsp" method="post">
  用户名：<input type="text" name="name">
  密　码：<input type="password" name="pwd">
  <input type="submit" name="确定">
</form>
```

传统的脚本方式是用：

```
<%
name=request.getParameter("name");
password=request.getParameter("pwd") ;
 %>
```

来提取用户名和密码输入框中的数据，并把它赋给变量name和password。使用JSTL后，上面的这段代码变成了标记：

```
<c:set var="Tname" value="${param.name}"/>
<c:set var="Tpassword " value="${param.pwd}"/>
```

<c:set...>很像HTML中的标记，它用于定义变量，如上定义了两个变量Tname和Tpassword，而var和value就是<c:set...>标记的属性了。

2.2 输出数据

JSP中用<%out.println(name+"
");%>来输出变量name的值并换行，现在只要用<c:out ...>标记配合HTML语言中的
就可以实现了，如：

```
<c:out value="${Tname}"/>
<br>
```

2.3 条件判断

在验证用户的合法性时，要判断密码是不是正确的，原来使用如下语句：

```
<%
if(password=="******")
 out.print(name+"是合法的用户!"+"<br>");
 %>
```

现在可以替换成：

```
<c:if test="${Tpassword=='******'}">
 <c:out value="${Tname}"/>
 <c:out value="是合法的用户!"/>
 <br>
</c:if>
```

<c:if...>.....</c:if>是进行条件判断的标记，表达式Tpassword=='******'返回一个真或假的值给test属性。

2.4 循环控制

JSTL中使用<c:forEach...>..... </c:forEach>标记来实现程序设计的循环。如实现从1开始到10为止，步长为2的循环：

```
<c:forEach var="i" begin="${1}" end="${10}" step="${2}">
  <c:out value="${i}"/>
</c:forEach>
```

将输出1 3 5 7 9这5个数。

<c:forEach...>..... </c:forEach>标记还有另外一种表示方式，它将Blue，Red和Green保存在items中，通过循环逐个取出items中的值。如：

```
<c:forEach var="color" items="Blue,Red,Green">
  <c:out value="${color}"/><br>
</c:forEach>
```

而且，items还可以是Enumeration等其他类型的赋值。

2.5 文件包含

动态网页设计为了实现“分而治之”的方法以及文件的共享，常把网页分成几个部分，放在不同的网页文件中，然后在一个文件中把它们包含进来。JSTL中使用<c:import url="includedfile"/>就可以包含一个文件，如：

```
<c:import url="inc.jsp"/>包含了inc.jsp这个网页
<c:import url="http://www.yahoo.com" >就把yahoo的主页面也包含进来了。
```

2.6 重定向

```
<c:redirect url="redir.jsp"/>将当前程序转向redir.jsp上执行。
<c:redirect url=" http://www.yahoo.com"/>重定向到yahoo网站。
```

3 数据库访问

数据库访问是动态网页的重要组成部分，它可以实现页面数据与后端数据库的交互。比如，新用户注册时需要将用户输入的信息保存到数据库，而用户登录时需要从数据库端提取数据对用户身份进行校验，等等。采用JSTL标记进行数据访问的优势在于，开发者只需关注要与哪一个数据库连接，获得什么样的结果就可以了，而具体的连接细节交给后端的Java代码来执行，这比在页面中加入Java脚本，显得更为简单方便，而且更易于排除错误。使用JSTL进行数据连接等操作之前，也同样也在程序中加入<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>，以定义一个进行数据库操作的前缀sql。

3.1 建立连接

在对数据库进行操作之前，要先对数据库建立一个连接，以后每次标准SQL操作都建立在这个连接之上进行的。比如，在JSP脚本中是这样建立连接：

```
<%
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  //设置驱动器
 Connection conn=DriverManager.getConnection("jdbc:odbc:xml");
 //通过JDBC-ODBC与xml数据库别名建立连接
%>
```

采用JSTL的<sql:setDataSource.../>标记后，就变成如下的代码：

```
<sql:setDataSource
 var="conn"
 driver="sun.jdbc.odbc.JdbcOdbcDriver"
 url="jdbc:odbc:xml"
/>
```

这里也是建立一个conn连接，以后的查

询操作都可以采用conn来进行。

3.2 进行查询

<sql:query...>标记中有两个关键属性var和dataSource，下面代码在执行中var属性声明了查询返回的数据集rs，dataSource属性说明了所执行的操作是在conn连接基础上进行的：

```
<sql:query var="rs" dataSource="${conn}
">
    SELECT * FROM t1
</sql:query>
```

只要把要执行操作的SQL语句写在<sql:query...>.....</sql:query>之间就可以生成查询的数据集rs,下面对查询的结果rs进行输出：

```
<c:forEach var="r" items="${rs.rows}">
  <c:out value="${r.ID}"/><
  <c:out value="${r.NAME}"/>
</c:forEach>
```

假如执行查询的表中含有id和name两个字段,在循环输出的过程中，把rs的数据集合存放在items中，通过r把items中的数据逐个取出。

3.3 数据库更新

JSTL通过把SQL语句包含在<sql:update...>....</sql:update>标记中，实现数据库内容的更新。使用update标记可以实现除查询以外的所有标准SQL操作，包括create、update、insert、drop等。程序开发人员只要设置标记中的属性，把想要实现的SQL放在标记中就可以了。

3.4 使用参数

动态网页设计中与数据库的交互，数据往往是以参数形式传送的。比如，要把页面上FORM中输入框的数据保存(insert)到数据中，根据用户的请求动态查询(select)数据库，等等。下面是使用参数向数据库插入数据的例子：'

```
<sql:update dataSource="${conn}">
    INSERT INTO xml (id,name)VALUES (?,?)
    <sql:param value="3"/>
    <sql:param value="${newName}"/>
</sql:update>
```

在insert语句中，使用了两个问号（?），说明有两个参数要传入，再使用<sql:param .../>标记把动态数据有顺序地传入。

参 考 文 献

1 http://www.delphibbs.com/delphibbs/dispq.asp?lid=995899 [EB/OL]，2002-04-29。

2 JSTL1.0 Spec-Public Draft：http://jcp.org/aboutJava/communityprocess/review/jsr052 [EB/OL]. 2002-03-04。