The Application of XML Deploy JSP

XML部署JSP应用

摘要：用XML标记来部署JSP三层模型，可使程序结构清晰、实现构件重用，系统容易调整和扩展。本文以一个花卉定购站点为实例，从四个方面实际地论述了XML在JSP中的应用。

关键词：可扩展标识语言 JSP 电子商务

王劲松 王 健 （南京东南大学 计算机科学与技术系 210096）

1 引言

随着Internet网的迅速发展，电子商务、电子政务、远程教育等，基于服务器和客户端的技术逐渐得到普及。Web服务器的编程，正经历一场从产生、发展到成熟的变革，CGI、PHP、ASP到JSP，一浪高过一浪。特别是JSP，以Java语言作为基础，一处编程多处运行，有着良好的跨平台性，以线程作为运行的单位，大大提高了网络的性能。计算机、手机等所有可联网的家用电器都能通过网络与JSP服务器交换信息，J2EE、J2SE、J2ME分别为不同可编程设备规范了平台，更进一步拓广了JSP服务器的应用范围。

比较成熟的JSP技术采用MVC三层结构（即模型Model、视图View、控制器Controller）。模型对流程中的数据对象进行抽象和封装，视图将各种数据以界面的形式展现给终端用户，控制器体现商务逻辑控制整个程序的流程。采用MVC技术，结构清晰，各功能模块划分明确。擅长美工、外观布局的Web开发人员可专门从事JSP显示页面的制作工作，精通Java网络原理的软件程序员可以集中精力进行商务逻辑的设计工作，而各种模型类则由熟悉数据库和数据结构的人员来完成。三个部分的开发工作能同步进行，系统便于维护和扩展。

纯Java的Java bean、JSP和Servlet虽然实现了MVC三层结构。但是，一个Servlet的改变将直接影响到调用它的所有JSP页面，使它们都要作相应的改动。同时，为了完成显示逻辑，在JSP页面中仍然包含大量的Java代码，降低了页面的可读性。而且，难以实现代码重用和大粒度的软件构件，缺乏数据交换的通用性、兼容性。

解决这些问题的最好方法是引入XML技术，XML即扩展标识语言（eXtensible Markup Language），具有一套统一的数据格式，非常适于应用程序之间数据的交换，特别是松耦合的应用程序，如：分布式Web系统。XML可以促进应用程序代码的重用，提高应用程序对需求变化的适应能力。所以，把XML处理数据方面的跨平台性与JSP处理商务逻辑的跨平台性进行组合将是一种较为理想的结合。下面就以一个具体的JSP三层模型花卉站点来应用XML技术。

2 将XML技术引入JSP服务器

这是一个电子商务系统（花卉在线定购网站）采用JSP三层模型结构，系统平台为Windows 2000，以Apache作为服务器，以Tomcat作为JSP引擎，数据库采用SQL Server 2000。本系统在下列四个方面应用了XML技术。

2.1 使用XML部署文件 映射Servlet动作

在JSP默认的运行目录下的WEB-INF目录下有个web.xml文件，JSP可以使用这个XML文件来映射Servlet动作。如图1所示：

默认的运行目录为star，视图JSP文件都在star目录中，控制器Servlet类都在classes目录中。如果menu.jsp（总菜单jsp）要调用

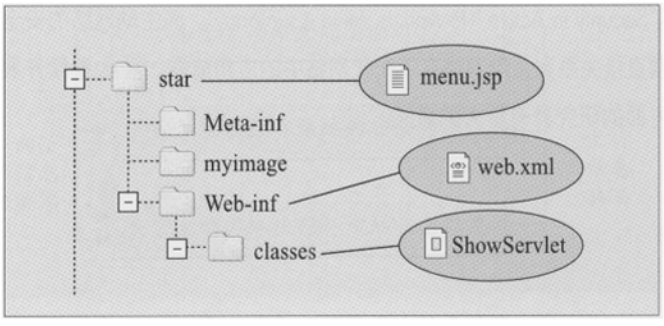

图 1

ShowServlet.class（查询显示Servlet），那么在menu.jsp中应有类似的HTML语句：

```
<a href="servlet/ShowServlet">1、花卉信息查询</a>
```

这种用名称直接调用的方法使页面代码不便于理解和记忆。而映射的方法则是在Web－inf目录下的web.xml文件中加入内容：

```
<servlet>
<!- Servlet映射开始->
        <servlet-name>
          to-show        <!- 使用时的映射名称->
        </servlet-name>
        <servlet-class>
          ShowServlet      <!- Servlet实际的类名->
        </servlet-class>
    </servlet>          <!-Servlet映射结束->
```

能实现to-show到ShowServlet的映射，以后要调用ShowServlet.class只需用语句：

```
<a href="servlet/to-show">1、花卉信息查询</a>
```

这样做的好处在于若ShowServlet.class发生变动（改换或更名）时只需在web.xml文件中找到上述<servlet-class>标记：将Servlet实际的类名改为新的名称即可，例如将ShowServlet改为PlayServlet。而所有调用它的JSP页面都无需作任何改动，就能把to-show动作改为映射到PlayServlet.class。也就是说控制器Servlet类的更新不会直接影响到调用它的视图JSP页面，这样做可以使两部分（视图和控制器）的设计工作同时进行，自成体系。

2.2 使用XML定制标记 封装Java代码

三层模型虽然把控制逻辑封装在Servlet类中，在一定程度上减轻了JSP页面的份量，也简化了JSP的设计，但是为了实现显示逻辑，JSP页面中仍然含有大量的Java代码。解决这个问题的一个方法是使用XML定制标记，在JSP中使用XML定制标记表示的是特殊域的功能。按照JSP1.1规范；执行时，JSP页面的实现过程将使用可用的Tag实例……接着停止使用Tag实例……然后释放Tag实例。Javax.servlet.jsp.tagext包提供了实现定制标记需要的接口和类，几乎所有的标记处理程序均通过扩展TagSupport或BodyTagSupport类来实现Tag接口。下面具体说明这一过程。

若JSP页面通过session从Javabean模型中获得单价：pricebuy购买数量：sum然后计算出总价（float型）sumprice= pricebuy*sum，由于是货币在显示时需保留两位小数且应转化为字符串。并且所有包含货币显示的页面都要这么去做，因此可把实现这种功能的Java代码可封装在tostringTag.java标记类中，部分语句如下：

```
public class tostringTag extends TagSupport
{  private float value;//用来存放待加工的float型变量
   public void setPrice (float sumprice)
  //获得需要转化为字符串的float型变量
  {this. value = sumprice;}//将这一获得的值传给value
   public int doStartTag () throws JspException
  { ˇˇˇˇ//这里是把float型的value保留两位小数且转化为String型并存储在sback中的Java代码。ˇˇˇˇ
  try {pageContext.getOut (). print (sback);}
  //将结果值sback输出到JSP文件中
     catch(java.io.IOException ex) {
       throw new JspException (ex.getMessage ());
     }//捕获输出异常
  return SKIP_BODY;
  //继续执行本标记后面的JSP页面内容
   }}
```

注意：标记类首先执行属性设置函数setPrice (float sumprice)，从JSP页面标记中获得要转化的float型变量，然后执行标记开始函数doStartTag()，产生需要的String型变量并把它输出到页面。

为使用这一定制标记还要对图2位置的web.xml和mylib.tld两个文件进行配置，其中mylib.tld是一个自定义的标记处理映射文件，相当于一个自定义的标记库，通过它可以实现tostring、togb2312等标记到相应的标记处理类的映射。配置如下：

（1）在web.xml文件中加入内容：实现使用mylib标记来映射/WEB-INF/tlds/mylib.tld文件。

```
<!-使用mylib标记来映射标记处理文件->
<taglib>
        <taglib-uri>mylib</taglib-uri>  <taglib-location>/WEB-INF/tlds/mylib.tld</taglib-location>
</taglib>
```

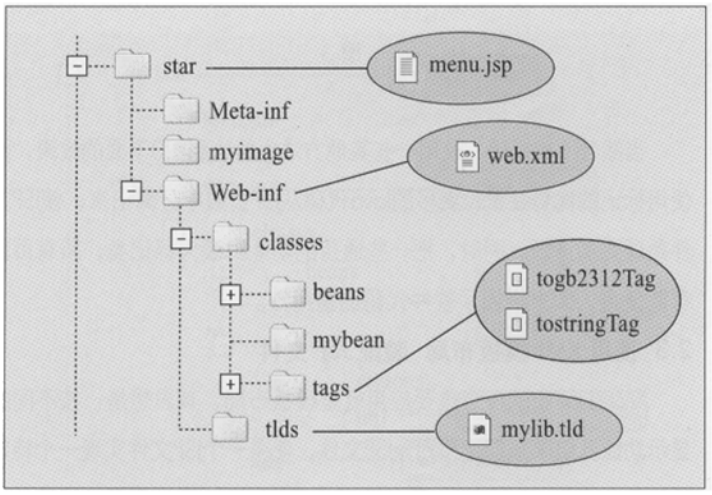

图2

（2）在mylib.tld文件中加入内容：实现tostring标记到处理类tostringTag.class的映射。

```
<!-把标记tostring映射到tags包下tostringTag类->
<tag>
<name>tostring</name>  //标记名
<tagclass>tags.tostringTag</tagclass> //类名
<bodycontent>empty</bodycontent> //标记主体为空
<attribute>
<name> sumprice </name>   //标记属性
<required>true</required>   //需要属性
</attribute>
</tag>
```

配置完成后只在需要使用标记的JSP文件头部导入mylib标记库：

```
<%@ taglib uri='mylib' prefix='mylib'%>
```

就可以使用标记了。

同理，也可以构造一个togb2312Tag.java标记类来实现ISO8859_1码到中文GB2312码的转换。

比如在购物车cart.jsp页面使用标记如下：

```
//以中文GB2312码显示花卉名称
<mylib:togb2312 zhnum='<%=mynamebuy %>'/>
//以两位小数且字符串形式显示购买金额
<mylib:tostring sumprice ='<%=pricebuy%>'/>
```

效果如图3所示：上面的未使用标记，下面的使用了标记。

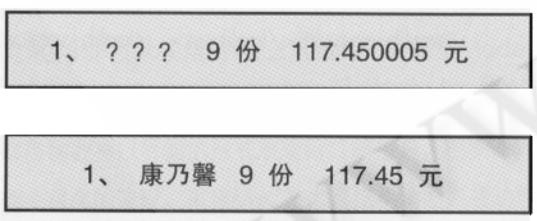

图 3

当然若不用标记，改用一些其他方法也能达到图3下面的效果，而使用标记的优势在于：更多的Java代码从JSP页面中分离出来，使JSP文件条理更加清晰，同时，把一些通用的功能封装为标记类，所有页面都能使用，大大增强了软件代码的重用性。

2.3 使用 XML 模板布局 装配 JSP 文件

XML在JSP中的一个重要应用就是模板布局，其思想是：按不同的显示功能及要求来创建和划分JSP文件，使得一个JSP文件实现一个特定的显示功能，如：greet.jsp显示欢迎信息，bigmenu.jsp显示功能菜单，log.jsp显示注册界面，foot.jsp显示返回菜单等。这些功能单一的JSP文件可以看成是一个个的构件，再由这些构件来组成功能完整的页面，如：主页面Star.jsp应包含欢迎信息greet.jsp、功能菜单bigmenu.jsp等，购物页面BuyHTML.jsp应包含物品列表viewHTML.jsp、购物车cart.jsp、返回菜单foot.jsp等。这样每个小页面就是一个大粒度的Web构件，可以装配到任何需要它的组合页面中，若某一显示功能需要修改，则只需修改实现这一功能的JSP小页面，其他页面都不用改动。同样，某个组合页面需要增删功能，则只需增加或减少相应的JSP小页面即可。如此设计就要求组合页面能合理的布局它所包含的小页面，仅仅用include语句显然是很不够的，这里采用XML模板布局。如图4所示。

（1）在tags/regions目录下创建实现模板的标记类：RegionTag、

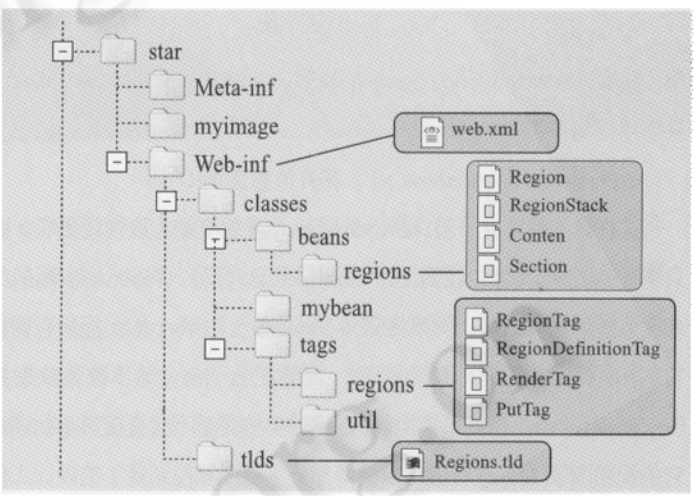

图4

RegionDefinitionTag、RenderTag、PutTag，（其中RegionTag是基类）；在beans/regions目录下创建这些标记类所使用的Java bean：Region、RegionStack、Content和Section。并建立标记类的引导文件库regions.tld，在web.xml文件中添加相应的映射内容，使得JSP中的调用能映射到标记类中。这些内容与上一节（1.2节）相似，这里不再描述。

（2）创建组合页面布局使用的模板module.jsp文件，主要代码如下：

```
<body>
<table align="center">
   <tr >   <td>   <region: render section='sidebar'/> //侧面
   </td>
   <td>   <table><tr><td>
<region: render section='header'/>   //头部
</td></tr><tr><td>
<region: render section='content'/>   //内容
```

```
</td></tr><tr><td>
<region: render section='footer'/>     //脚部
</td></tr></table>
   </td>  </tr> </table>
</body>
```

从文件中可以看出这个模板包含四个部分：侧面sidebar、头部header、内容content和脚部footer，很多组合页面都能使用这一模板。使用模板后的页面可读性好、结构清晰。如：主页面Star.jsp

```
<%@ taglib uri='regions' prefix='region' %>
<region: render template='module.jsp'>
   <region: put section='header' content='greet.jsp'/>
   <region: put section='content' content='bigmenu.jsp'/>
</region: render>
```

第一行导入标记库'regions，第二行使用布局模板module.jsp，第三行将欢迎信息小页面greet.jsp放在头部，第四行将功能菜单小页面bigmenu.jsp放在内容中，第五行标记结束。又如：购物页面BuyHTML.jsp

```
<%@ taglib uri='regions' prefix='region' %>
<region: render template='module.jsp'>
   <region: put section='sidebar' content='Bigcart.jsp'/>
   <region: put section='content' content='viewHTML.jsp'/>
   <region: put section='footer' content='foot.jsp'/>
</region: render>
```

所不同的地方是，购物车小页面cart.jsp放在侧面，物品列表小页面viewHTML.jsp放在内容，返回菜单小页面foot.jsp放在脚部。

从这两个组合页面可以看到一个共同的特点，那就是实现了，如：购物车小页面cart.jsp、菜单小页面bigmenu.jsp等JSP形式的大粒度构件，可以在任何需要它们的组合页面中灵活装配。整个页面完全符合XML格式，结构清晰，一目了然，容易装配和维护。

图 5

（3）进而，还可以创建多块模板来实现不同的布局风格（不同的摆放位置、不同的背景、不同的广告甚至不同的JavaScript动画特效等）。例如刚才的主页面Star.jsp，第二行使用布局模板module.jsp时显示效果如图5。

还是主页面Star.jsp，第二行使用布局模板改为module2.jsp时显示效果如图6。仅显示效果不同，功能完全相同。

可见，使用XML模板布局来装配JSP文件，使网站灵活布局以满足不断变化的需要。

图 6

2.4 使用 XML 报表输出 统一 Data 格式

一个电子商务网站，总要产生一些数据报表，以便管理部门作为分析资料或送货公司作为订单依据来处理。数据信息在网上最通用的格式是XML，因此，本系统中将产生XML形式的客户订货报表文件。

下面就用XML定制标记来实现XML形式的报表文件，如图7所示。

（1） 在目录star\WEB-INF\classes\tags\util\下创建报表的数据模型三个Java bean，它们是Buylist（报表单类）、Guest（客户类）和Item（购买项目类）。一个报表文件包含一个报表单，一个报表单可以包含多个客户，而每个客户又可以有多个购买项目。所以，报表单

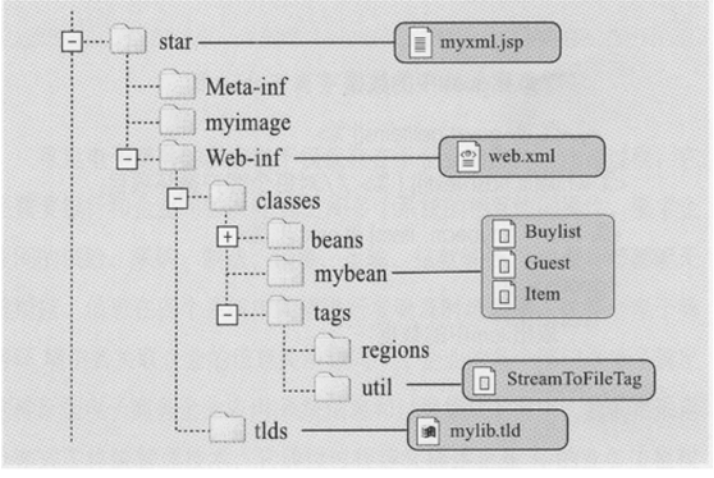

图7

Buylist中有一个矢量guests用来装多个客户Guest，客户Guest中有一个矢量items用来装多个购买项Item。而购买项Item中封装了花卉名称、单价、数量、定购日期等信息。

（2）在目录star\WEB-INF\classes\tags\util\下创建能产生XML数据报表的主体标记类StreamToFileTag其主要代码如下：

```
public class StreamToFileTag extends BodyTagSupport {
 private String filename;  // filename属性用来保存文件名
 public void setFile(String filename) {  //设置文件名
  this.filename  = filename;}
 public int doAfterBody () throws JspException {
  try {FileWriter writer=
 new FileWriter(new File(filename)); //建立写文件流
  writer. write(bodyContent.getString().trim()); //执行写入
  writer. close();   } //关闭写文件流
  catch(java.io.IOException e) {  //捕获异常
   throw new JspException (e.getMessage ());}
 return SKIP_BODY;   }      } //继续执行后面的JSP内容
```

执行过程是：从标记获得属性setFile()（要生成的文件名），然后对标记主体求值（产生文件的内容），然后是主体后操作doAfterBody ()（把内容写入文件）。

（3）要使用这一标记同前两节一样，需对web.xml和mylib.tld文件实施相应的布置。最后使用这一标记的myxml.jsp文件代码如下：

```
<mylib: streamToFile file='C:/tomcat/webapps/star/list.xml'>
// streamToFile标记开始，其中file是标记的属性
//将创建的XML文件路径是C:/tomcat/webapps/star/list.xml
<?xml version="1.0" encoding="gb2312"?> //产生XML文件头
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
 <list>  //根元素list表单开始
  <% while(it.hasNext()) { %> //对表单循环产生客户
   <% guest = (mybean. Guest) it.next (); %>
   <guest>
    ~~~~取出guest中的数据~~~~
    <% itt=guest.getItems();%>
  <% while(itt.hasNext()) { %> //对每个客户循环项目
   <% item = (mybean. Item) itt.next (); %>
   <goods>
    ~~~~取出item中的数据~~~~
   </goods>
     <%} %>
   </guest>
    <%} %>
  </list>  //根元素list表单结束
 </mylib:streamToFile>  //标记结束
```

此JSP的运行将产生一个名为list.xml的报表文件。图8是某一次运行时产生的list.xml文件再配上相应的style.xsl式样文件时的显示效果：

客户：王劲松
共支付：59.20元

购买品种	数量	合计金额	日期
非洲菊	2份	24.00元	Sun Feb 09 17:13:59 CST 2003
白玫瑰	4份	35.20元	Sun Feb 09 17:13:59 CST 2003

客户：令狐冲
共支付：86.50元

购买品种	数量	合计金额	日期
贝壳花	5份	86.50元	Sun Feb 09 17:15:31 CST 2003

客户：东方不败

图8

3 总结

在JSP三层模型中使用XML定制标记进行布局，能使网站页面结构清晰，形成良好的文件格式，易于理解，易于维护。能大大提高程序的代码重用，创建大粒度的构件，灵活装配，灵活使用。整个系统的组装就像一台机器上组装零件一样直观和方便。

JSP的语言基础是Java，由于其良好的平台兼容性，使得它能实现Internet网，无线电讯网，以及将来的家用电器设备网的编程。而XML来源于SGML，其数据的自我描述性，在表示数据方面有着突出优势，已逐渐成为Internet网上的通用格式。JSP的程序跨平台性与XML的数据跨平台性相结合是开发跨平台网络系统的最佳组合。能充分满足当前不断发展中的网络编程的需要，具有广阔的前景。

参 考 文 献

1 （美）Aaron Skonnard、Martin Gudgin 编著，牛韬、英宇 译，《XML 精要》，人民邮电出版社 2002、10。

2 （美）David M.Geary 编著，贺民 译，《advanced JavaServerPages》，科学出版社，2002、9。

3 （美）Mark Wutka 编著，程显华译，《JSP 和 Servlet 程序设计使用专辑》，机械工业出版社，2002、3。