**RichFaces-jsf，a4j，rich项目中总结的标签说明**

一：Jsf标签**:** <%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>

<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>

1．我们使用了JSF 的core 与html 标签库，core 是有关于UI 组件的处理，而html 则

是有关于HTML 的进阶标签。

**2.<f:view>**与<html>有类似的作用，当您要开始使用JSF 组件时，这些组件一定要在

<f: view>与</f:view>之间，就如同使用HTML 时，所有的标签一定要在<html>与<

/html>之间。

3. html 卷标库（即<h>标签）中几乎都是与HTML 卷标相关的进阶卷标，<h:form>会产生一个窗体，

我们使用<h: inputText>来显示user 这个Bean 对象的name 属性，而

<h:commandButton>会产生一个提交按钮，我们在action 属性中指定将根据之前定义

的login 页面流程中前往welcome.jsp 页面。

Jsp标签：<jsp:include flush="true" page="../../template/header.jsp" />

4. <h:dataTable>配合<h:column>来以表格的方式显示数据，<h:column>表示一列。< h:column>中只能包括 JSF 组件或者是<f:facet>，JSF支援两种facet：header 与footer表格的表头与表尾文字。

<h:dataTable value="#{tableBean.userList}" var="user">

<h:column>

<f:facet name="header">

<h:outputText value="Name"/>

</f:facet>

<h:outputText value="#{user.name}"/>

<f:facet name="footer">

<h:outputText value="\*\*\*\*"/>

</f:facet>

</h:column>

<h:dataTable>

5. <h:panelGroup>:这个组件用来将数个JSF 组件包装起来，使其看来像是一个组件. <h:panelGroup>

<h:commandButton value="submit" action="login"/>

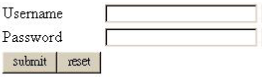
<h:commandButton value="reset" type="reset"/>

</h:panelGroup>

在<h:panelGroup>中包括了两个<h:commandButton>，这使得<

h:panelGrid>在处理时，将那两个<h:commandButton>看作是一个组件来看待，其完

成的版面配置如下所示：



6.关于Jsf标签**rendered**（）属性的使用：主要是用来rendered的判断，如下面代码，减少了大量的js的验证，简单好用。

<h:outputText value="#{lbl.User\_Add\_Title }" rendered="#{empty userBean.userPanel.id }"/>

<h:outputText value="#{lbl.User\_Update\_Title }" rendered="#{not empty userBean.userPanel.id }"/>

7.<h:inputTextarea>:对应<textarea>

8. <h:selectBooleanCheckbox>单选框

<h:selectOneRadio>单选       属性layout="pageDirection竖列

<h:selectOneRadio value="#{user.education}">

<f:selectItem itemLabel="高中" itemValue="高中"/>

<f:selectItem itemLabel="大学" itemValue="大学"/>

<f:selectItem itemLabel="研究所以上" itemValue="研究所以上"/>

</h:selectOneRadio><p>

**9.<h:inputHidden>** inputHidden：对应<input type=”hidden”>隐藏字段，即输出<input> HTML 卷标，其type 属性设定为hidden，隐藏字段的值，用于保留一些讯息于客户端，以在下一次发送窗体时一并送出。

10. <f:loadBundle>  可以使用<f:loadBundle>卷标来指定加载讯息资源

<f:loadBundle basename="messages" var="msgs"/>

11.<h:inputSecret value="#{user.password}"

required="true">

<f:validateLength minimum="6"/>

   </h:inputSecret><p>我们设定了required 属性为true，这

表示这个字段一定要输入值，我们也在</h:inputSecret>设定了<f: validateLength>，

并设定其minimum 属性为6，这表示这个字段最少需要6 个字符。  
**12.验证器（**validator**）**：Jsf简单验证器的设计 不过一般项目中验证器都是写在js中  这里我只是小谈一下。例如：

UserBean.java

package onlyfun.caterpillar;

import javax.faces.application.FacesMessage;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.validator.ValidatorException;

public class UserBean {

....

public void validate(FacesContext context,UIComponent component,Object obj)

throws ValidatorException {

String password = (String) obj;

if(password.length() < 6) {

FacesMessage message = new FacesMessage(

FacesMessage.SEVERITY\_ERROR,

"字符长度小于6",

"字符长度不得小于6");

throw new ValidatorException(message);

}

if(!password.matches(".+[0-9]+")) {

FacesMessage message = new FacesMessage(

FacesMessage.SEVERITY\_ERROR,"密码必须包括字符与数字","密码必须是字符加数字所组成");

throw new ValidatorException(message);

}

}

}

接着可以在页面下如下使用验证器：

.....

<h:inputSecret value="#{user.password}"

required="true"

validator="#{user.validate}"/>

**14.<h:messages>**：错误讯息处理

在使用标准转换器或验证器时，当发生错误时，会有一些预设的错误讯息显示，这些讯

息可以使用<h:messages>或<h:message>标签来显示出来，而这些预设的错误讯息也

是可以修改的，您所要作的是提供一个讯息资源文件，例如：

messages.properties

javax.faces.component.UIInput.CONVERSION=Format Error.

javax.faces.component.UIInput.REQUIRED=Please input your data.

....

javax.faces.component.UIInput.CONVERSION 是用来设定当转换器发现错误时

显示的讯息，而javax.faces.component.UIInput.REQUIRED 是在标签设定了

required 为true，而使用者没有在字段输入时显示的错误讯息。

您要在faces-config.xml 中告诉JSF 您使用的讯息文件名称，例如：

faces-config.xml

<?xml version="1.0"?>

<!DOCTYPE faces-config PUBLIC"-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"

"http://java.sun.com/dtd/web-facesconfig\_1\_0.dtd">

<faces-config>

<application>

<local-config>

<default-locale>en</default-locale>

<supported-locale>zh\_TW</supported-locale>

</local-config>

<message-bundle>messages</message-bundle>

</application>

.....

</faces-config>

在这边我们设定了讯息档案的名称为messages\_xx\_YY.properties，其中xx\_YY 是

根据您的Locale 来决定，转换器或验证器的错误讯息如果有设定的话，就使用设定值，如

果没有设定的话，就使用默认值。

**15.<h:panelGrid>**

这个卷标可以用来作简单的组件排版，它会使用HTML 表格卷标来绘制表格，并将组

件置于其中，主要指定columns 属性，例如设定为 2：

<h:panelGrid columns="2">

<h:outputText value="Username"/>

<h:inputText id="name" value="#{userBean.name}"/>

<h:outputText value="Password"/>

<h:inputText id="password" value="#{userBean.password}"/>

<h:commandButton value="submit" action="login"/>

<h:commandButton value="reset" type="reset"/>

</h:panelGrid>

则自动将组件分作 2 个 column 来排列，排列出来的样子如下：

[](http://blog.photo.sina.com.cn/showpic.html#url=http://s12.sinaimg.cn/orignal/a26264c9n7afb9972695b)

<h:panelGrid>的本体间只能包括JSF 组件，如果想要放入非JSF 组件，例如简单的

样版（template）文字，则要使用 <f:verbatim>包括住，例如：

<h:panelGrid columns="2">

<f:verbatim>Username</f:verbatim>

<h:inputText id="name" value="#{userBean.name}"/>

<f:verbatim>Password</f:verbatim>

<h:inputText id="password" value="#{userBean.password}"/>

<h:commandButton value="submit" action="login"/>

<h:commandButton value="reset" type="reset"/>

</h:panelGrid>

**16.<h:panelGroup>**

这个组件用来将数个JSF 组件包装起来，使其看来像是一个组件，例如：

<h:panelGrid columns="2">

<h:outputText value="Username"/>

<h:inputText id="name" value="#{userBean.name}"/>

<h:outputText value="Password"/>

<h:inputText id="password" value="#{userBean.password}"/>

<h:panelGroup>

<h:commandButton value="submit" action="login"/>

<h:commandButton value="reset" type="reset"/>

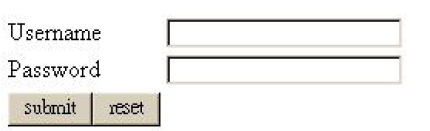
</h:panelGroup>

</h:panelGrid>

在<h:panelGroup>中包括了两个<h:commandButton>，这使得<

h:panelGrid>在处理时，将那两个<h:commandButton>看作是一个组件来看待，其完

成的版面配置如下所示：

[](http://blog.photo.sina.com.cn/showpic.html#url=http://s6.sinaimg.cn/orignal/a26264c9ncdd404b07ef5)  
**17.<h:selectBooleanCheckbox>:**在视图上呈现一个复选框，例如：  
我同意 <h:selectBooleanCheckbox value="#\{user.aggree\}"/>  
value 所绑定的属性必须接受与传回boolean 型态。这个组件在网页上呈现的外观如下：  
我同意 [http://s7.sinaimg.cn/mw690/a26264c9ncde5eee6b4e6&690](http://blog.photo.sina.com.cn/showpic.html#url=http://s7.sinaimg.cn/orignal/a26264c9ncde5eee6b4e6)

项目中的: 

<rich:column>  
   <f:facet name="header">  
       <h:selectBooleanCheckbox onclick="checkedAll(this,{formId:'mainForm'});"/>  
   </f:facet>  
   <h:selectBooleanCheckbox value="#{aul.checked }">  
        <a4j:support event="onclick" ajaxSingle="true">  
        </a4j:support>  
   </h:selectBooleanCheckbox>  
</rich:column>  
**18.<f:convertDateTime>**中，我们使用pattern 指定日期的样式为dd/MM/yyyy，

即「日/月/公元」格式。

**二：Richfaces的**A4j**标签：**<%@taglib uri="http://richfaces.org/a4j" prefix="a4j"%>1

1. **<a4j:loadScript src="resource://jquery.js" />**  
与<script>相似，但它能从其他资源加载脚本比如jar 包  
<script type="text/javascript" src="<%=webroot%>/js/global.js"></script>

**2.<a4j:support event="onkeyup" reRender="rep"/>**  
为非 Ajax 控件添加Ajax能力。   
    <h:form **id**="myform">  
        <h:panelGrid **columns**="3">  
            <h:inputText **id**="myinput" **value**="#{userBean.name}">  
                <a4j:support **event**="onkeyup" **reRender**="outtext，outtext1" />  
            </h:inputText>  
            <h:outputText **id**="outtext" **value**="#{userBean.name}" />  
        </h:panelGrid>  
   </h:form>

在这里我们在父标签(<h:inputText>)中添加了一个AJAX 支持. 该支持绑定了JavaScript事件“onkeyup” .因此, 每一次该事件发布给父标签时,我们的程序将发送一个AJAX请求到Server.这意味着我们的受管理的bean将包含该“text” 域中我们输入的最新数据.

<a4j:support> 标签的“reRender” 属性(attribute)定义我们的页面的哪一部分被更新. 在这里,该页面唯一被更新的部位是<h:outputText>标签,因为他的ID值和“reRender” 的属性值向匹配. 在一个页面中更新多个元素(elements)也是很简单的:仅仅把他们的IDs放在 “reRender” 属性中就可以了.   
**<a4j:support>控件说明**

 两个主要的属性：  
event ： 定义javascript事件，当此事件被触发时，触发 Ajax support来实现和后台的交互

onblur 事件会在对象失去焦点时发生。  
onchange   
onclick  
ondblclick  
  
  
reRender : 定义当后台返回请求数据后，画面上需要刷新的组建的 ID，复数时用逗号【，】分隔

其他常用属性：  
action，actionListener：指定后台事件  
onsubmit：指定在提交时执行的javascript代码  
oncomplete：指定当请求处理完成后需要执行的javascript代码  
focus:指定当请求处理完成后需要设定focus的控件的ID

**2.<a4j:support event="onchange" ajaxSingle="true"></a4j:support>**

当ajaxSingle为true时，只是提交当前组件，而对于其他组件是不提交的，它的附带效果就是只对当前组件进行验证，而对其他的组件不会作任何验证。这一点上对于button和link来说感觉和immediate="true"的效果很相似，即跳过所有验证，当然只是效果除此以外它们完全是不同的。

**3.“immediate”** 值为“true”时，立即执行“ActionListener”（即在应用请求值阶段），而不是等到调用应用阶段。使用immediate=”是一些数据模型方式此其他不能成功通过而被需要提交表单按钮链接常<a4j:commandButton action="…" immediate="true" … />

**4.** **<a4j:commandButton**   value="#{lbl.Query}" action="#{roleBean.findRoleByLike}"styleClass="button white"></a4j:commandButton>

**5. <**a4j:commandLink reRender="popPanel" immediate="true" action="#{roleBean.add }" style="color: #ffffff;"  oncomplete="Richfaces.showModalPanel('RolePanel');">

<h:graphicImage url="/images/add.png" /></a4j:commandLink>

**6.reRender**属性－就如我们前面提到过的,在一个Ajax响应到来时重新渲染(reRendered)组件.如果要刷新多个id，则需要要逗号隔开。如:<a4j:commandLink reRender="dataTable messageAreaMain" style="color: #ffffff;" immediate="true" onclick="if(!confirmDialog()){return;}"action="#{userBean.delete }">

比如<a4j:commandButton .... reRender="number"     .... >  
这个属性会把id为number的元素重新渲染一遍。  
如果是一个会显示后台数据的元素，则会重新去后台取数据。

**7.rendered**的判断是否显示组件,是在服务端解析EL表达式.所以你可以在BackBean中用简单的逻辑值替换复杂的表达式.  
JSF是会生成一些看起来多余的标记,这是为了组件的通用性.

**8.  6和7的结合：.......**

<h:panelGroup id="errorMsg">  
<h:outputLabel rendered="#{XXXXX.errorMsgIsRendered}" value="Name is required！"/>  
</h:panelGroup>  
.......  
<a4j:commandButton id="save" value="Save" eventsQueue="coding"   
action="#{XXXXX.showErrorMessage}"  
oncomplete="#{XXXXX.popupProcessControl}"  
reRender="errorMsg" />  
在XXXXX的showErrorMessage这个方法中是通过页面上的name字段是否填写，来给errorMsgIsRendered这个属性赋值的,rendered值为false则则不显示元素，为true则显示错误信息。   
reRender是重新渲染某些元素，当rendered的值改变后，要重新渲染才会有变化。  
oncomplete的值是一些javascript语句。

**9. immediate** : “immediate” 值为“true”时，立即执行“ActionListener”（即在应用请求值阶段），而不是等到调用应用阶段。使用immediate =”true ”是更新一些数据模型值的种方式,此时其他值因不能成功通过验证而被更新。在不需要提交表单的按钮或链接中，常使用此属性如：

<a4j:commandButton action="…" immediate="true" … />  
**10. <a4j:region>**

1）**<a4j:region>**组件用来限制部分组件树被服务端处理。包括在解码、转换、验证和模型更新期间的数据处理。注意，整个表单仍然被提交，但只是进入区域（region）的部分会被处理。

Example:Example: Example: Example:Example:

<h:form>

...

<a4j:region>

<a4j:commandLink />

</a4j:region>

...

<h:form>

上例中，通过调用 <a4j:commandLink>会提交整个表单， 但只有在<a4j:region>包裹的区域会被服务器处理。region可以嵌套。服务器只提取和解码包含初始化请求组件的区域。

<h:panelGrid>

<a4j:region>

<a4j:region>

 <h:inputText id="input1">

<a4j:support event="onblur" />

 </h:inputText>

</a4j:region>

 <h:inputText id="input2">

<a4j:support event="onblur" />

 </h:inputText>

</a4j:region>

 <h:inputText id="input3">

<a4j:support event="onblur" />

 </h:inputText>

</h:panelGrid>

上例中，如果由input1input1 input1 触发请求提交，那么只有当前的区域将被服务器处理。如果 input2i触发 请求提交， 那么input1 和input2 将同时被服务器处理。如果是input3触发的请求提交，因为它属于整个页面区域,所以个页面都将被服务器处理。

使用"renderRegionOnly"属性，在渲染响应阶段仅更新当前region。这时区域直接被传入解码，减少了执行时间。不过使用时应注意，它不准许更新活动区域之外的数据。如:

<h:form>

...

<a4j:region renderRegionOnly="true">

<a4j:commandLink value="Link 1" id="link1"/>

</a4j:region>

...

<a4j:region renderRegionOnly="false">

<a4j:commandLink value="Link 2" id="link2"/>

</a4j:region>

...

</h:form> 有一点要记住的是，region只在一个AJAX请求方面的工作。region不使用非AJAX请求。

2）**<a4j:region>可以与迭代组件一起使用（如<rich:column> 或 <rich:scrollableDataTable>等）**，不用更新整个表格，且不用任何另外的侦听器，就可以重新渲染表中的一行。如：

<rich:column>

<a4j:region>

<a4j:commandLink reRender="out"/>

</a4j:region>

</rich:column>

<rich:column>

<h:outputText id="out">

</rich:column>

大多数情况下没有必要使用<a4j:region>，ViewRoot是一个默认的<a4j:region>

**11.<a4j:support event="onblur" ajaxSingle="true"/>**onblur:onblur 事件会在对象失去焦点时发生。

三：Richfaces的Rich标签：<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>

**1.<rich:dataTable>：**用于渲染输出表格，内置Ajax支持。

<rich:dataTable>除具有 <h:dataTable> 功能外增加了Ajax支持，还row和column span，支持"header"、"footer"和"caption" facet，支持主子表模式。该组件基于<a4j:repeat>创建，因此可以利用Ajax进行局部更新。利用"ajaxKeys"属性定义在一个Ajax请求之后被更新的行键。

**扩展表功能，比如，可滚动数据、行选择，以及列重新排序、隐藏列，应使用<rich:extendedDataTable>组件。**

例如：

<rich:dataTable value="#{capitalsBean.capitals}" var="cap" rows="5">

  <f:facet name="caption">

    <h:outputText value="United States Capitals" />

  </f:facet>

  <f:facet name="header">

     <h:outputText value="Capitals and States Table" />

  </f:facet>

  <rich:column>

    <f:facet name="header">State Flag</f:facet>

      <h:graphicImage value="#{cap.stateFlag}"/>

    <f:facet name="footer">State Flag</f:facet>

  </rich:column>

  <rich:column>

    <f:facet name="header">State Name</f:facet>

      <h:outputText value="#{cap.state}"/>

    <f:facet name="footer">State Name</f:facet>

  </rich:column>

  <rich:column >

    <f:facet name="header">State Capital</f:facet>

       <h:outputText value="#{cap.name}"/>

    <f:facet name="footer">State Capital</f:facet>

  </rich:column>

  <rich:column>

    <f:facet name="header">Time Zone</f:facet>

      <h:outputText value="#{cap.timeZone}"/>

    <f:facet name="footer">Time Zone</f:facet>

  </rich:column>

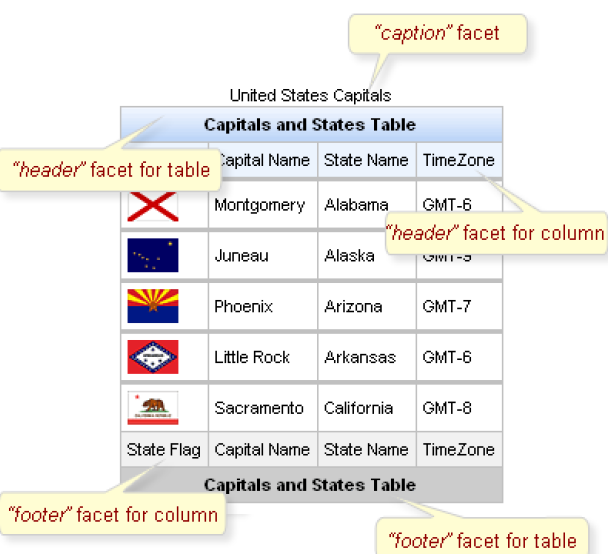
  <f:facet name="footer">

    <h:outputText value="Capitals and States Table" />

 </f:facet>

</rich:dataTable>

结果为：

[](http://blog.photo.sina.com.cn/showpic.html#url=http://s3.sinaimg.cn/orignal/a26264c9n7afd73332db2)

**2. <rich:extendedDataTable>**见pdf

**rich:extendedDataTable组件提供如下特性**

* 水平和垂直滚动
* 垂直滚动时Ajax延迟加载
* 冻结的列，即水平滚动是不滚动的列。rich:extendedDataTable的frozenColumns属性设置冻结的列数

* **rich:extendedDataTable组件行选择的用法**

selectionMode属性控制选择模式，有3个可选值

* multiple（多行选择）
* single（单行选择）
* none（无选择模式）

<rich:extendedDataTable enableContextMenu="false" border="1" width="400px" height="300px" binding="#{roleBean.table}" selection="#{roleBean.selection}" rows="10"cellspacing="0.5" cellpadding="0" id="dataTable"  value="#{roleBean.allRoleList }" var="arl">

<rich:column>

<f:facet name="header" >

<h:selectBooleanCheckbox onclick="checkedAll(this,{formId:'mainForm'});"/>

</f:facet>

<h:selectBooleanCheckbox value="#{arl.checked }">

<a4j:support event="onchange" ajaxSingle="true"></a4j:support>

</h:selectBooleanCheckbox>

<%-- <h:inputHidden value="#{arl.id }"/> --%>

</rich:column>

<rich:column>

<f:facet name="header" >

<h:outputText value="#{lbl.RoleName}"/>

</f:facet>

<h:outputText value="#{arl.roleName}" />

</rich:column>

</rich:extendedDataTable>

2. <rich:modalPanel>见项目和pdf：

主要是用来[http://s5.sinaimg.cn/mw690/a26264c9n00b06bd9d124&690](http://blog.photo.sina.com.cn/showpic.html#url=http://s5.sinaimg.cn/orignal/a26264c9n00b06bd9d124)

<h:panelGroupid="popPanel">

<rich:modalPanel id="RolePanel" width="350" height="200" >

<f:facet name="header">

<h:panelGroup>

<h:outputText value="#{lbl.Role\_Add\_Title}" rendered="#{emptyroleBean.rolePop.id}"/>

<h:outputText value="#{lbl.Role\_Update\_Title}" rendered="#{not empty roleBean.rolePop.id}"/></h:panelGroup>

</f:facet>

 3. <rich:calendar>日期控件 见pdf

 <rich:calendardatePattern="yyyy-MM-dd HH:mm" locale="#{locale}"id="pop\_occurTime" disabled="true"value="#{logBean.logDetailsPopVo.occurTime}"styleClass="searchbox required" ><a4j:support event="onchange"ajaxSingle="true"/></rich:calendar>

这几天一直在用Richfaces， 现在使用的是3.3的版本，碰到不少问题，先一条条记录下来，以后逐步完善：  
1.rich:scrollableDataTable 在Firefox中不能显示标题，这是richefaces的一个bug，还没有解决，只能暂时使用rich:extendedDataTable来代替以达到scrollalbe效果。  
2.richfaces所带的drag-and-drop在jboss部署时可能碰到问题，需要将jboss-service.xml 中的UseJBossWebLoader设置为true。否则都会出现Method not found错误，提示找不到dropListener方法。 参考： http://www.jboss.org/index.html?module=bb&op=viewtopic&p=4032847#4032847  
3.使用rich:modalPanel时，如果要在弹出的panel中使用表单功能，一定要确保两点：  
（1）<rich:modalPanel> 一定要放在主页面中form的外部。  
（2）在<rich:modalPanel>中要包含自己的form。  
4.当你遇到 Property 'xxxx' not found on type org.javassist.tmp.java.lang.Object 时，请查看一下是否已经将该变量的setter和getter方法在local interface中定义。如果没有定义将会报类似这样的错误。  
5.要使用richfaces，页面最好使用facelet实现。  
6.在使用各个组件时，如果要在外围加div或panel，id一定注意最好不要用‘XXXContainer’，因为richfaces默认将一些组件解析成html的时候会加一些id类似这样的层。例如：  
<rich:panel id="abcContainer" style="display:none;" rendered="true">  
<rich:modalPanel id="abc">  
afdsasdfs  
</rich:modalPanel>  
</rich:panel>  
这种情况下，这个modalPanel就不会显示出来，而是在原页面出现一个空框。原因就是richfaces自动生成一个id也为 abcContainer的层，跟外围的panel的id相同，所以modalPanle中的东西没有显示，只是将原来的panel显示出来了。  
7.使用<a4j:status>时，如果一个页面内有多个status组件，就可能报错： “duplicate Id for a component \_viewRoot“。  
解决方法，将status组件的forceId设置为true. 例如：  
<a4j:status forceId="true" id="xxStatus">  
xxx  
</a4j:status>  
8.Property  'XXX' not found on type org.jboss.seam.jsf.ListDataModel.当使用一个DataModel时，如果引用其中的属性，可能会报这个错误。我是在使 用rich:tree的时候碰到这个问题的，但使用DataModel时，如果在tree里面因为节点的子属性就会报这个错误。  
解决方法：将@DataModel改为@Out(required=false)并将对应的数据集合设置为一个@Factory.