Jsf;

Ajax,jsf集成框架，ajax4jsf；

JBoss 收购 A4J 后，又将其集成到 RichFaces 中RichFaces；---公司4.35版；

~

ajax4jsf，无刷新操作实现；

最新版本：5.0.x；

参考：

RichFaces官网文档，

<http://docs.jboss.org/richfaces/latest_4_3_X/Component_Reference/en-US/html_single/>

示例，

<http://showcase.richfaces.org/>

-----------------------------

目录：

配置文件；

功能；

控件使用；

//////////////////////////////////////////////////////////////////////////////////////

Xhtml：

Richfaces标签库引入每个xhtml页面，  
<ui:composition xmlns:r="http://richfaces.org">

……

</ui:composition>

配置文件：

Web.xml：

// JSF2 设置(当前JVM)默认时区

<context-param>

<param-name>javax.faces.DATETIMECONVERTER\_DEFAULT\_TIMEZONE\_IS\_SYSTEM\_TIMEZONE</param-name>

<param-value>true</param-value>

</context-param>

//

<context-param>

<param-name>javax.faces.PROJECT\_STAGE</param-name>

<param-value>Development</param-value>

</context-param>

//Richfaces皮肤，

<context-param>

<param-name>org.richfaces.SKIN</param-name>

<param-value>blueSky</param-value>

</context-param>

//<param-name>org.richfaces.enableControlSkinning</param-name> 值对 RichFaces 和（出乎意料的是）标准 JSF 组件的外观有影响。如果其值为 true，标准控件都涂色。如果省略该元素，则默认值为 true。

<context-param>

<param-name>org.richfaces.enableControlSkinning</param-name>

<param-value>false</param-value>

</context-param>

//过滤器配置：

richfaces内置了几个解析器，TIDY解析器用于复杂的或非标准的标记，这些标记由过滤器做严格效验并更正。NEKO过滤器在解析代码时没有严格效验，大大加快了ajax请求的处理，但使用时你要确保你的应用程序标记是严格用于这个过滤器，否则可能会导致很多错误和恶化布局结果。

所以TIDY解析器相对NEKO解析器来说会慢很多。默认是TIDY解析器。

<context-param>

<param-name>org.ajax4jsf.xmlparser.ORDER</param-name>

<param-value>NEKO</param-value>

</context-param>

<context-param>

<param-name>org.ajax4jsf.xmlparser.NEKO</param-name>

<param-value>.\*\..\*</param-value>

</context-param>

<context-param>

//该配置将强制容器在发送到客户端之前压缩所有的Richfaces资源，包括图片、样式表、JavaScript等。它将大大的减少页面的加载时间。

<context-param>

<param-name>org.ajax4jsf.COMPRESS\_SCRIPT</param-name>

<param-value>true</param-value>

</context-param>

//该配置将保证所有和样式相关的文件在第一次访问Richfaces应用的请求到达时全部加载到客户端。

<context-param>

<param-name>org.richfaces.LoadStyleStrategy</param-name>

<param-value>all</param-value>

</context-param>

//JSF优化技巧将保证所有和Richfaces相关的JavaScript、文件、库等在第一次请求到达时被下载到客户端。

<context-param>

<param-name>org.richfaces.LoadScriptStrategy</param-name>

<param-value>all</param-value>

</context-param>

//最大上传大小20m

<param-name>org.richfaces.fileUpload.maxRequestSize</param-name>

<param-value>20971520</param-value>

</context-param>

//上传文件的存储方式：true，临时文件；false，内存中；

<context-param>

<param-name>org.richfaces.fileUpload.createTempFiles</param-name>

<param-value>true</param-value>

</context-param>

//指定facelets路径

<context-param>

<param-name>javax.faces.FACELETS\_LIBRARIES</param-name>

<param-value>/WEB-INF/shiro-face.taglib.xml</param-value>

</context-param>

//通过ContextLoaderListener加载contextConfigLocation指定的Spring配置文件。

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath\*:/applicationContext.xml,

classpath\*:/applicationContext-shiro.xml

</param-value>

</context-param>

//jsf servlet配置，

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<listener>

<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>

</listener>

//路径映射配置

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Faces Servlet</servlet-name>

<url-pattern>\*.xhtml</url-pattern>

</servlet-mapping>

//session和首页配置

<session-config>

<session-timeout>

30

</session-timeout>

</session-config>

<welcome-file-list>

**<welcome-file>greeting.xhtml</welcome-file>**

</welcome-file-list>

//shiro过滤器；

<filter>

<filter-name>shiroSecurityFilter</filter-name>

<filter-class>

org.springframework.web.filter.DelegatingFilterProxy

</filter-class>

</filter>

<filter-mapping>

<filter-name>shiroSecurityFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

//字体编码过滤器；

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

//错误页面

<error-page>

<exception-type>com.sun.faces.context.FacesFileNotFoundException</exception-type>

<location>/sessionout.xhtml</location>

</error-page>

<error-page>

<error-code>404</error-code>

<location>/sessionout.xhtml</location>

</error-page>

-----------------------------------------------

faces-config.xml，

<?xml version="1.0" encoding='UTF-8'?>

<faces-config version="2.2" xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_2.xsd">

...

</faces-config>

//自定义组件；

<component>

<component-type>hasRole</component-type>

<component-class>com.tag.HasRole</component-class>

</component>

<render-kit>

<renderer>

<component-family>hasRole</component-family>

<renderer-type>hasRoleRenderer</renderer-type>

<renderer-class>com.tag.HasRoleRenderer</renderer-class>

</renderer>

</render-kit>

<component>

<component-type>token</component-type>

<component-class>com.tag.Token</component-class>

</component>

<component>

<component-type>htmlRefresh</component-type>

<component-class>com.tag.HtmlRefresh</component-class>

</component>

//jsf,spring集成；DelegatingVariableResolver

<application>

<variable-resolver>

org.springframework.web.jsf.DelegatingVariableResolver

</variable-resolver>

<action-listener>com.cxdai.filter.ActionFilter</action-listener>

<locale-config>

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

<supported-locale>de</supported-locale>

<supported-locale>zh</supported-locale>

</locale-config>

</application>

shiro-face.taglib.xml：

//自定义组件对应的标签，

<tag>

<tag-name>hasRole</tag-name>

<component>

<component-type>hasRole</component-type>

<renderer-type>hasRoleRenderer</renderer-type>

</component>

<attribute>

<description></description>

<name>name</name>

<required>true</required>

</attribute>

</tag>

<tag>

<tag-name>token</tag-name>

<component>

<component-type>token</component-type>

</component>

</tag>

-----------------------------------------

功能：

自定义皮肤：

///////////////////////////////////////////////////////////////////////////

控件使用：

Ajax：

1，<a4j:ajax>

<a4j:ajax event="keyup" render="outtext" />

2，<a4j:param> ---加入参数；

<h:form **id**="form">

    <a4j:commandButton **value**="Set name to Alex" **reRender**="rep">

        <a4j:param **name**="username" **value**="Alex" **assignTo**="#{paramBean.name}"/>

    </a4j:commandButton>

    <h:outputText **id**="rep" **value**="Name: #{paramBean.name}"/>

</h:form>

点击按钮，设置rep控件中的的me值为alex；

3.3. <a4j:actionListener>

3.4. <a4j:commandButton>

Id,name,value，

Onclick，---action提交前执行，一般为js事件；

Action,

Execute,---@form,提交时携带表单信息；@this,提交时不携带信息；

render,---控件指向，

data,--- data="#{borrowMoneyAction.resultMsg}"，action返回提示信息；

oncomplete 在客户端请求完成时调用的js代码;

3.5. <a4j:commandLink>

Id,name,value,

onclick；

3.6. <a4j:jsFunction>

<h:outputText id="showname" value="#{functionBean.text}" />

<h:form>

<a4j:jsFunction name="updateName" render="showname">

<a4j:param name="name" assignTo="#{functionBean.text}"/>

</a4j:jsFunction>

</h:form>

鼠标移走后，改变内容；

3.7. <a4j:poll>

3.8. <a4j:push>

4.1. <a4j:mediaOutput> ---资源，

<a4j:mediaOutput element="img" cacheable="false" session="false"

createContent="#{mediaBean.paint}" value="#{mediaData}"

mimeType="image/jpeg" />

可以生成一张图片；

5.1. <a4j:outputPanel>

5.2. <a4j:region>

7.1. <a4j:queue>

<a4j:queue requestDelay="1500"/> 队列请求延迟1500毫秒；

7.2. <a4j:log>

7.3. <a4j:status>

---------------------

execute ：

@All

每个组件被处理。

@none

没有部件被处理。

@This

与execute属性一起的请求组件被处理。

@form

form包含的请求组件被处理。

@region

region包含的请求组件被处理。使用<R：region>组件作为包装元素来指定区域。有些组件利用额外的关键字。

//////////////////////////////////////////////////////////////////////////////////

6.1. <rich:validator> ---验证，

<rich:panel header="User information">

<h:panelGrid columns="3">

<h:outputText value="Name:" />

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

<rich:validator />

</h:inputText>

<rich:message for="name" />

<h:outputText value="Email" />

<h:inputText value="#{validationBean.email}" id="email">

<rich:validator />

</h:inputText>

<rich:message for="email" />

<h:outputText value="Age" />

<h:inputText value="#{validationBean.age}" id="age">

<rich:validator />

</h:inputText>

<rich:message for="age" />

<h:outputText value="I agree the terms" />

<h:selectBooleanCheckbox value="#{validationBean.agree}" id="agree">

<rich:validator/>

</h:selectBooleanCheckbox>

<rich:message for="agree" />

</h:panelGrid>

</rich:panel>

@ManagedBean

@RequestScoped

public class UserBean {

@Size(min=3, max=12)

private String name = null;

@Pattern(regexp = "^[\\w\\-]([\\.\\w])+[\\w]+@([\\w\\-]+\\.)+[a-zA-Z]{2,4}$" , message="Bad email")

private String email = null;

@Min(value = 18)

@Max(value = 99)

private Integer age;

//...

//Getters and Setters

}

<rich:graphValidator>

·示例代码；---可用于登录页面验证；

--------------------------

8. Rich inputs

8.1. <rich:autocomplete> 自动补全；

8.2. <rich:calendar> 日期控件；

8.3. <rich:editor> 富文本；

8.4. <rich:fileUpload> 上传；

8.5. <rich:inplaceInput> 文本框；

8.6. <rich:inplaceSelect> 下拉框；---有一个集合；

8.7. <rich:inputNumberSlider> 数字滑块；

8.8. <rich:inputNumberSpinner> 数字微调；

8.9. <rich:select> 下拉框；

8.10. <rich:orderingList> 列表；

8.11. <rich:pickList> 左右互选列表；

9. Panels

9.1. <rich:panel> 面板

9.2. <rich:accordion>

9.3. <rich:collapsiblePanel>

9.4. <rich:popupPanel> 浮层面板；

9.5. <rich:tabPanel> 选项卡；

9.6. <rich:togglePanel>

10. Tables and grids

10.1. <a4j:repeat> 表格数据循环---加载list数据；

10.2. <rich:dataTable> 表格；

10.3. <rich:column> 列；

10.4. <rich:columnGroup> 合并列；

10.5. <rich:collapsibleSubTable>

10.6. <rich:extendedDataTable> 可扩展表格(滚动条)

10.7. <rich:dataGrid> 网格表；

10.8. <rich:list> 列数据；

10.9. <rich:dataScroller> 滚轮表格；

10.10. Table filtering

11. Trees

11.1. <rich:tree> 树；

11.2. Tree adaptors

12，Menus and toolbars

12.1. <rich:dropDownMenu> 下拉菜单；

12.2. <rich:contextMenu>

12.3. Menu sub-components

12.3.1. <rich:menuItem>

12.3.2. <rich:menuGroup>

12.3.3. <rich:menuSeparator>

12.4. <rich:panelMenu> 面板菜单---可用于后台左侧菜单；

12.5. <rich:toolbar> 工具栏；

13. Output and messages

13.1. <rich:message> 输出信息到output控件；

13.2. <rich:messages>

13.3. <rich:notify>

13.4. <rich:notifyMessage>

13.5. <rich:notifyMessages>

13.6. <rich:notifyStack>

13.7. <rich:progressBar> 进度条；

13.8. <rich:tooltip> 提示；

14. Drag and drop

14.1. <rich:dragSource>

14.2. <rich:dropTarget>

14.3. <rich:dragIndicator>

15. Layout and appearance

15.1. <rich:jQuery>

16. Functions

16.1. rich:clientId

16.2. rich:component

16.3. rich:element

16.4. rich:jQuery

16.5. rich:findComponent

16.6. rich:isUserInRole

17. Functionality extension

17.1. <rich:componentControl>

17.2. <rich:focus>

17.3. <rich:hotKey>

17.4. <rich:hashParam>

17.5. <rich:placeholder>

/////////////////////////////////////////////////////////////////////////////

<rich:popupPanel*弹出窗体；*

 <h:commandButton value=*"Call the popup"*>

<rich:componentControl target=*"popup"* operation=*"show"* />

</h:commandButton>

<rich:popupPanel id=*"popup"* modal=*"false"* autosized=*"true"* resizeable=*"false"*>

<f:facet name=*"header"*>

<h:outputText value=*"Simple popup panel"* />

</f:facet>

<f:facet name=*"controls"*>

<h:outputLink value=*"#"* onclick="#{rich:component('popup')}.hide(); return false;">

X

</h:outputLink>

</f:facet>

<p>Any content might be inside this panel.</p>

<p>

The popup panel is open and closed from the javascript function of component client side object. The following code

<a href=*"#"* onclick="#{rich:component('popup')}.hide()">hide this panel</a>:

<f:verbatim>&#35;</f:verbatim>{rich:component('popup')}.hide()

</p>

</rich:popupPanel>