Jsf：视图层实现框架；| 类似，struts2；

Java7中版本2.2；

参考：

Ibm三篇文章，

<http://www.ibm.com/developerworks/cn/java/j-jsf2fu1/index.html>

官方文档，java7文档中，

<http://docs.oracle.com/javaee/>

7，简介；8，Facelets ；9，el表达式；

10，jsf控件：html标签，f标签；

11-转换，监听，验证；

12，控件使用；(控件，bean赋值)

13，ajax；14，复合组件；

15，创建自定义ui组件和其它自定义对象

16，应用；

api：

<https://javaserverfaces.java.net/nonav/docs/2.1/>

或上面java7文档中的；

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

.xhtml页面；

主内容：

页面；

配置文件；

业务逻辑中的相关；

Facelets-模版框架，

Composite-复合组件；

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

目录：

页面控件使用；

页面控件高级使用；

功能实现；

附录-控件；

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

页面控件使用：

页面赋值：

#{name}

基本控件1，ui:composition、c:set、h:ead、h:body；

<ui:composition xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:h=*"http://java.sun.com/jsf/html"*

xmlns:f=*"http://java.sun.com/jsf/core"*

xmlns:ui=*"http://java.sun.com/jsf/facelets"*

xmlns:a4j=*"http://richfaces.org/a4j"*

xmlns:rich=*"http://richfaces.org/rich"*

xmlns:shiro=*"http://xproject.flightCenter/tag"*

xmlns:c=*"http://java.sun.com/jsp/jstl/core"*>

<c:set var=*"path"* value=*"${request.contextPath}"* />

<h:head>

</h:head>

<h:body>

</h:body>

</ui:composition>

基本控件2，

表单，

<h:form prependId="false">

块，

<h:panelGrid columns="2">

输入框

<h:inputText value="#{place.streetAddress}" size="15"/>

-id,name,value；| size,

文本行；

<h:outputText value="#{UserNumberBean.maximum}"/>

超链，

<h:commandLink action="#{showcart.details}">

<h:outputText value="#{item.item.title}"/>

</h:commandLink>

按钮，

<h:commandButton value="#{msgs.goButtonText}"

style="font-family:Palatino;font-style:italic"

action="#{place.fetch}"/>

-id,value,action；

//密码

<h:inputSecret redisplay="false"

value="#{loginBean.password}" />

转换格式输出，

//输出内容为“hello,（f:param的value值）”

<h:outputFormat value="Hello, {0}!">

<f:param value="#{hello.name}"/>

</h:outputFormat>

//dataTable中取得集合数据，

<h:dataTable id="items" value="#{cart.items}" var="item"

<h:column>

信息；

<h:message

超链，

<h:link outcome="personal" value="personal greeting page!"

includeViewParams="true">

<f:param name="Result" value="#{hello.name}"/>

</h:link>

单选框：

<h:selectBooleanCheckbox title="#{bundle.receiveEmails}"

value="#{custFormBean.receiveEmails}">

</h:selectBooleanCheckbox>

<h:outputText value="#{bundle.receiveEmails}">

多选框：

<h:selectManyCheckbox id="newslettercheckbox"

layout="pageDirection"

value="#{cashierBean.newsletters}">

<f:selectItems value="#{cashierBean.newsletterItems}"/>

</h:selectManyCheckbox>

下拉列表；

<h:selectOneMenu id="shippingOption"

required="true"

value="#{cashierBean.shippingOption}">

<f:selectItem itemValue="2"

itemLabel="#{bundle.QuickShip}"/>

<f:selectItem itemValue="5"

itemLabel="#{bundle.NormalShip}"/>

<f:selectItem itemValue="7"

itemLabel="#{bundle.SaverShip}"/>

</h:selectOneMenu>

~

<h:selectOneMenu id=*"repayWay"* styleClass=*"bigselect"* value=*"#{borrowAction.borrow.repayWay}"*>

<f:selectItems value=*"#{borrowAction.repayWays}"* />

</h:selectOneMenu>

单选按钮，

<h:selectBooleanCheckbox id="fanClub"

rendered="false"

binding="#{cashierBean.specialOffer}" />

<h:outputLabel for="fanClub"

rendered="false"

binding="#{cashierBean.specialOfferText}"

value="#{bundle.DukeFanClub}" />

</h:outputLabel>

h:inputHidden

基本控件3：

包装，

<f:facet name="header">

<h:outputText value="#{bundle.ItemTitle}"/>

</f:facet>

参数，

<f:param

格式，

<h:outputText value="#{item.createTimestamp}"><f:convertDateTime pattern="yyyy-MM-dd HH:mm:ss" /></h:outputText>

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

控件高级使用：

图片；

<h:graphicImage library="images" name="cloudy.gif"/>

//图标组件，

<util:icon image="#{resource['images:back.jpg']}"

actionMethod="#{places.logout}"

style="border: thin solid lightBlue"/>

引入js和css外部文件，

<h:outputStylesheet library="css" name="styles.css" target="body"/>

<h:outputScript library="javascript" name="util.js" target="head"/>

##### Facelets模版：/templates/masterLayout.xhtml

<div class="pageHeading">

<**ui:insert** name="heading">

#{msgs.placesHeading}

</ui:insert>

</div>

<div class="menuLeft">

<**ui:insert** name="menuLeft"/>

</div>

<**ui:insert** name="menuRight">

<ui:include src="/sections/shared/sourceViewer.xhtml"/>

</ui:insert>

ui:define，覆盖内容；

login 视图

<ui:composition xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://java.sun.com/jsf/facelets"

template="/templates/masterLayout.xhtml">

<ui:define name="menuLeft">

<ui:include src="/sections/login/menuLeft.xhtml"/>

</ui:define>

<ui:define name="content">

<ui:include src="/sections/login/content.xhtml"/>

</ui:define>

</ui:composition>

~

也可以修改menuRight内容；

<ui:define name="menuRight">

<ui:include src="/sections/showSource/menuRight.xhtml"/>

</ui:define>

实现定制组件：---组件开发模式；

·定义组件，然后使用；| 示例：login组件；

·复合嵌套组件；

Ajax化：

Ajax验证，

<h:inputText id="name" value="#{cc.attrs.managedBean.name}"

**valueChangeListener**="#{cc.attrs.managedBean.validateName}">

<**f:ajax** event="**blur**" render="**nameError**"/>

</h:inputText>

<h:outputText id="**nameError**"

value="#{cc.attrs.managedBean.**nameError**}"

style="color: red;font-style: italic;"/>

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

转换：

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

<f:converter converterId="javax.faces.Integer" />

</h:inputText>

<h:outputText value="#{cashierBean.shipDate}">

<f:convertDateTime pattern="EEEEEEEE, MMM dd, yyyy" />

</h:outputText>

~

<f:convertDateTime type="date" dateStyle="full" />

~

<f:convertDateTime dateStyle="full"

locale="es"

timeStyle="long" type="both" />

<h:outputText value="#{cart.total}">

<f:convertNumber currencySymbol="$" type="currency"/>

</h:outputText>

监听：

<h:inputText id="name"

size="30"

value="#{cashierBean.name}"

required="true"

requiredMessage="#{bundle.ReqCustomerName}">

<f:valueChangeListener

type="javaeetutorial.dukesbookstore.listeners.NameChanged" />

</h:inputText>

验证：

<h:inputText id="quantity" size="4" value="#{item.quantity}">

<f:validateLongRange minimum="1"/>

</h:inputText>

<h:message for="quantity"/>

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

Ajax：

<h:commandButton id=*"submit"* value=*"Submit"*>

<f:ajax execute=*"userNo"* render=*"result"* />

</h:commandButton>

<h:outputText id=*"result"* value=*"#{userNumberBean.response}"* />

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

功能实现：

**表 2. JSF 2 托管 bean 注释（@...Scoped 注释只对 @ManagedBean 有效）**

| **托管 bean 注释** | **描述** | **属性** |
| --- | --- | --- |
| @ManagedBean | 以托管 bean 的形式注册一个类实例，然后将其放入到使用其中一个 @...Scoped 注释指定的范围内。如果没有指定任何范围，JSF 将把此 bean 放入请求范围，如果没有指定任何名称，JSF 将把类名的第一个字母转换为小写，形成一个托管 bean 名称；例如，如果类名为UserBean，那么 JSF 将创建一个托管 bean，其名为 userBean。eager 和 name 属性都是可选的。  注释必须结合使用一个实现零参数构造器的 Java 类。 | eager，name |
| @ManagedProperty | 为托管 bean 设置一个属性。注释必须放在类成员变量的声明之前。name 属性指定特性的名称，默认情况下为成员变量的名称。value 属性是特性的值，可以是一个字符串，也可以是一个 JSF 表达式，比如 #{...}。 | value，name |
| @ApplicationScoped | 在应用程序范围内存储托管 bean。 |  |
| @SessionScoped | 在会话范围内存储托管 bean。 |  |
| @RequestScoped | 在请求范围内存储托管 bean。 |  |
| @ViewScoped | 在视图范围内存储托管 bean。 |  |
| @NoneScoped | 将托管 bean 指定为没有范围。无范围的托管 bean 在被其他 bean 引用时比较有用。 |  |
| @CustomScoped | 在定制范围内存储托管 bean。  定制范围就是指可以由页面创建者访问的地图。可以通过编程的方式控制定制范围内的 bean 的可视性和生命周期。value 属性指向一个地图。 | value |

项目业务逻辑类，

@ManagedBean(name = "productAction")

@ViewScoped

**public** **class** ProductAction **implements** Serializable{

//service类，

@ManagedProperty(name = "financialProductService", value = "#{financialProductService}")

**private** FinancialProductService financialProductService;

//get/set;

//实体类---理财产品对象

@ManagedProperty(name = "financialProduct", value = "#{financialProduct}")

**private** FinancialProduct financialProduct;

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

开发一个managed baen：Guess Number Facelets Application

页面；

<h:form>

<h2>

Hi, my name is Duke. I am thinking of a number from

#{userNumberBean.minimum} to #{userNumberBean.maximum}.

Can you guess it?

</h2>

<p>

<h:inputText id=*"userNo"*

title=*"Enter a number from 0 to 10:"*

value=*"#{userNumberBean.userNumber}"*>

<f:validateLongRange minimum=*"#{userNumberBean.minimum}"*

maximum=*"#{userNumberBean.maximum}"*/>

</h:inputText>

<h:commandButton id=*"submit"* value=*"Submit"*>

<f:ajax execute=*"userNo"* render=*"result"* />

</h:commandButton>

<h:outputText id=*"result"* value=*"#{userNumberBean.response}"* />

</p>

<h:message showSummary=*"true"* showDetail=*"false"*

style="color: *#d20005*;

font-family: *'New Century Schoolbook', serif*;

font-style: *oblique*;

text-decoration: *overline*"

id=*"errors1"*

for=*"userNo"*/>

</h:form>

//错误提示，将ajax中的render值改为message的id；

~

类，

@ManagedBean(name = "userNumberBean")

@SessionScoped

**public** **class** UserNumberBean **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 5443351151396868724L;

Integer randomInt = **null**;

Integer userNumber = **null**;

String response = **null**;

**private** **int** maximum = 10;

**private** **int** minimum = 0;

**public** UserNumberBean() {

Random randomGR = **new** Random();

randomInt = **new** Integer(randomGR.nextInt(maximum + 1));

// Print number to server log

System.***out***.println("Duke's number: " + randomInt);

}

**public** **void** setUserNumber(Integer user\_number) {

userNumber = user\_number;

}

**public** Integer getUserNumber() {

**return** userNumber;

}

**public** String getResponse() {

**if** ((userNumber == **null**) || (userNumber.compareTo(randomInt) != 0)) {

**return** "Sorry, " + userNumber + " is incorrect.";

} **else** {

**return** "Yay! You got it!";

}

}

**public** **int** getMaximum() {

**return** (**this**.maximum);

}

**public** **void** setMaximum(**int** maximum) {

**this**.maximum = maximum;

}

**public** **int** getMinimum() {

**return** (**this**.minimum);

}

**public** **void** setMinimum(**int** minimum) {

**this**.minimum = minimum;

}

}

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

Jsf(自定义)组件开发：

参考：

官网文档15章，

<http://blog.csdn.net/goalstar/article/details/1638687>

<http://blog.sina.com.cn/s/blog_4d4e3d0f010009mk.html>

JSF 自带的标准组件集里面有 JSP 绑定，可以生成 HTML 渲染。

组件的功能通常围绕着两个动作：解码和编码数据。

解码（decoding） 是把进入的请求参数转换成组件的值的过程。

编码 (encode) 是把组件的当前值转换成对应的标记（也就是 HTML）的过程。

JSF 组件由两部分构成：组件和渲染器。JSF 组件 类定义 UI 组件的状态和行为；渲染器 定义如何从请求读取组件、如何显示组件 —— 通常通过 HTML 渲染。

每个组件都关联着一个组件类型，组件类型用于在 face 的上下文配置文件（faces-config.xml）中登记组件

<component>

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

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

</component>

getFamily（）方法主要是返回在faces-config.xml中定义了相关联的渲染器

<render-kit>

<renderer>

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

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

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

</renderer>

</render-kit>

自定义组件类，继承UIComponentBase，

**public** **class** HasRole **extends** UIComponentBase{

@Override

**public** String getFamily() {

// **TODO** Auto-generated method stub

**return** "hasRole";

}

@Override

**public** **void** encodeBegin (FacesContext context) **throws** IOException {

@Override

public Object saveState(FacesContext context) {

Object values[] = new Object[2];

values[0] = super.saveState(context);

values[1] = current;

return (values);

}

public void restoreState(FacesContext context, Object state) {

Object values[] = (Object[]) state;

super.restoreState(context, values[0]);

current = (String) values[1];

}

定义标签库描述文件，shiro-face.taglib.xml；

<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>

----------

公司：faces-config.xml，shiro-face.taglib.xml，tag包类文件；

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

附录-控件：

Html标签，

| **Tag** | **Functions** | **Rendered As** | **Appearance** |
| --- | --- | --- | --- |
| h:column | 列 | A column of data in an HTML table | A column in a table |
| h:commandButton | 提交表单到应用程序 | An HTML <input type=*value*> element for which the type value can be "submit", "reset", or "image" | A button |
| h:commandLink | 链接到另一页面或本页一位置 | An HTML <a href>element | A link |
| h:dataTable | 数据表 | An HTML <table> element | A table that can be updated dynamically |
| h:form | 表单 | An HTML <form> element | No appearance |
| h:graphicImage | 显示图像 | An HTML <img> element | An image |
| h:inputFile | 上传文件控件 | An HTML <input type="file"> element | A field with a**Browse...** button |
| h:inputHidden | 隐藏控件 | An HTML <input type="hidden"> element | No appearance |
| h:inputSecret | 加密控件 | An HTML <input type="password"> element | A field that displays a row of characters instead of the actual string entered |
| h:inputText | 文本框 | An HTML <input type="text"> element | A field |
| h:inputTextarea | 文本框域 | An HTML <textarea>element | A multirow field |
| h:message | 显示一个本地消息 | An HTML <span> tag if styles are used | A text string |
| h:messages | 显示本地消息 | A set of HTML <span> tags if styles are used | A text string |
| h:outputFormat | 显示格式化信息 | Plain text | Plain text |
| h:outputLabel | 文本 | An HTML <label> element | Plain text |
| h:outputLink | 链接到页面上的其他页面或位置，而不会产生一个动作事件 | An HTML <a> element | A link |
| h:outputText | 行文本 | Plain text | Plain text |
| h:panelGrid | 表格 | An HTML <table> element with <tr> and <td>elements | A table |
| h:panelGroup | 一个组件下的表格 | A HTML <div> or <span>element | A row in a table |
| h:selectBooleanCheckbox | 选择框 | An HTML <input type="checkbox"> element | A check box |
| h:selectManyCheckbox | 多选框 | A set of HTML <input>elements of type checkbox | A group of check boxes |
| h:selectManyListbox | 允许用户选择一组项目的同时显示多个项目 | An HTML <select>element | A box |
| h:selectManyMenu | 允许用户从一组项目中选择多个项目 | An HTML <select>element | A menu |
| h:selectOneListbox | 允许用户从一组项目中选择一个项目一次性全部显示 | An HTML <select>element | A box |
| h:selectOneMenu | 允许用户从一组项目中选择一个项目一次性全部显示 | An HTML <select>element | A menu |
| h:selectOneRadio | 选择一个按钮 | An HTML <input type="radio"> element | A group of options |

Core标签：

| **Tag** | **Function** |
| --- | --- |
| f:actionListener | 添加一个动作监听父组件 |
| f:phaseListener | 添加一个PhaseListener到一个页面 |
| f:setPropertyActionListener | 注册一个特殊的动作监听器，其唯一目的是当表单被提交时，推动一个值到一个托管bean |
| f:valueChangeListener | 添加一个值变化，监听到父组件 |
| f:converter | 添加任意转换器到父组件 |
| f:convertDateTime | 添加一个DateTimeConverter实例到父组件 |
| f:convertNumber | 添加一个NumberConverter实例到父组件 |
| f:facet | 封闭标签的嵌套组件 |
| f:metadata | 注册一个方面到父组件 |
| f:selectItem | 代表项目列表中的一个项目 |
| f:selectItems | 代表一组项目 |
| f:attribute | 父组件的属性/添加属性到父组件 |
| f:loadBundle | Map到资源包 |
| f:param | 参数 |
| f:ajax | ajax |
| f:event | 允许在组件上安装ComponentSystemEventListener |

***Facelets 模版标签***

| **Tag** | **Function** |
| --- | --- |
| ui:component | Defines a component that is created and added to the component tree. |
| ui:composition | Defines a page composition that optionally uses a template. Content outside of this tag is ignored. |
| ui:debug | Defines a debug component that is created and added to the component tree. |
| ui:decorate | Similar to the composition tag but does not disregard content outside this tag. |
| ui:define | Defines content that is inserted into a page by a template. |
| ui:fragment | Similar to the component tag but does not disregard content outside this tag. |
| ui:include | Encapsulates and reuses content for multiple pages. |
| ui:insert | 添加内容到模版； |
| ui:param | Used to pass parameters to an included file. |
| ui:repeat | Used as an alternative for loop tags, such as c:forEach or h:dataTable. |
| ui:remove | 从页面移除内容； |

***Composite 组件标签***

| **Tag** | **Function** |
| --- | --- |
| composite:interface | Declares the usage contract for a composite component. The composite component can be used as a single component whose feature set is the union of the features declared in the usage contract. |
| composite:implementation | Defines the implementation of the composite component. If a composite:interface element appears, there must be a corresponding composite:implementation. |
| composite:attribute | Declares an attribute that may be given to an instance of the composite component in which this tag is declared. |
| composite:insertChildren | Any child components or template text within the composite component tag in the using page will be reparented into the composite component at the point indicated by this tag's placement within the composite:implementation section. |
| composite:valueHolder | Declares that the composite component whose contract is declared by the composite:interface in which this element is nested exposes an implementation of ValueHolder suitable for use as the target of attached objects in the using page. |
| composite:editableValueHolder | Declares that the composite component whose contract is declared by the composite:interface in which this element is nested exposes an implementation of EditableValueHoldersuitable for use as the target of attached objects in the using page. |
| composite:actionSource | Declares that the composite component whose contract is declared by the composite:interface in which this element is nested exposes an implementation of ActionSource2 suitable for use as the target of attached objects in the using page. |