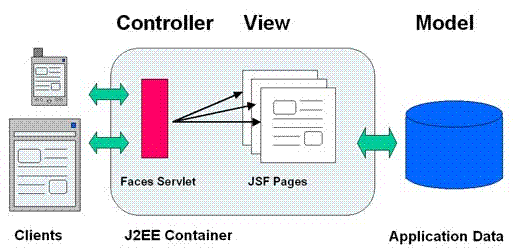
**JSF 体系结构：**

JSF 的主要优势之一就是它既是 Java Web 应用程序的用户界面标准又是严格遵循模型-视图－控制器 (MVC) 设 计模式的框架。用户界面代码（视图）与应用程序数据和逻辑（模型）的清晰分离使 JSF 应用程序更易于管理。为了准备提供页面对应用程序数据访问的 JSF 上下文和防止对页面未授权或不正确的访问，所有与应用程序的用户交互均由一个前端FacesServlet（控制器）来处理。



**JSF 生命周期：**

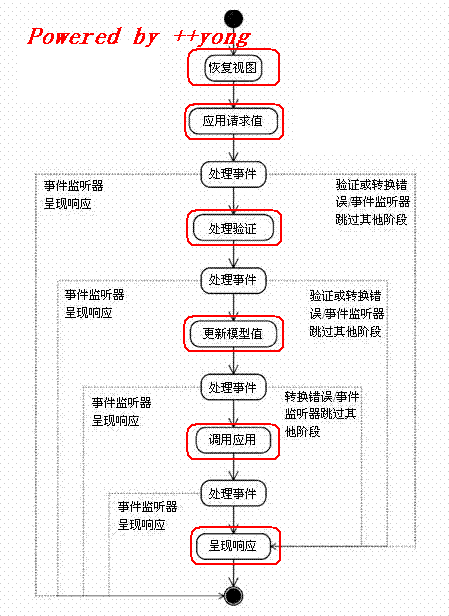
FacesServlet 充当用户和 JSF 应用程序之间的纽带。它在明确限定的 **JSF 生命周期（规定了用户请求之间的整个事件流）的范围内工作。**

1．   当JSF页面上的一个事件发生时（比如：用户单击了一个按钮），事件通知通过HTTP发往服务器。服务器端使用FacesServet这个特殊的Servlet处理该通知。

2．   FacesServlet一接收到用户的请求就创建一个FacesContext对象（JSF上下文，它存放了应用程序的所有数据）。在处理过程中，主要修改的就是这个FaceContext对象。

3．   接着就是处理过程，处理器是一个叫作Lifecycle的对象。FacesServet把控制权转交给Lifecycle对象。该对象分6个阶段来处理FacesContext对象以生成响应，最后将响应发回客户端。

Lifecycle对象处理JSP请求所需要的一系列动作称为请求处理生命周期。过程状态图如下：



由于请求处理生命周期里的应用请求值、处理验证、更新模型值和调用应用程序等阶段都可以在当前的请求对应的FacesContext实例中添加事件，因此，JSF实现必须在这些阶段后处理这些事件。

|  |  |
| --- | --- |
| 阶段 | 说明 |
| 恢复视图 | 为选定的视图找到或创建组件树。  一旦用户单击JSP页面上的链接或按钮，就会启动此阶段。JSF应用里的JSP页面被表示成一个组件树。JSF实现会进一步将这些组件链接到事 件处理程序和验证程序，并将视图保存在FacesContext对象中，以备后面的处理过程所用。FacesContext对象包含了JSF用来管理当前 会话中当前请求的GUI组件状态所需要的所有状态信息。 |
| 应用请求值 | 使用请求中发送来的值来更新组件树的组件值。因为请求中发送来的值都是String类型的，所以在更新组件树的组件值之前，必须将这些值转换为相应类型。这个过程也是解码。若转换有错误，这些错误将添加到FacesContext对象。 |
| 处理验证 | 当每个组件的本地值被更新后，Lifecycle对象都会根据这些注册组件的验证规则来验证这些值的合法性。  如果输入的值不符合验证规则，就会将验证错误添加至FacesContext对象，并将组件标记为无效。JSF将转至呈现响应阶段，并显示带有验证错误消息的视图。  如果没有遇到验证错误，JSF将进入下一阶段。 |
| 更新模型值 | 更新与组件相关的后台bean（也叫管理bean）或者模型对象的值。只有那些与组件值绑定在一起的Bean属性才会被更新。 |
| 调用应用程序 | JSF控制器调用应用程序来处理应用程序级的事件，如提交一个表单。（此阶段可执行业务逻辑） |
| 呈现响应 | 使用当前的显示技术（如JSP）显示选定的视图。 |

一个jsf的实例

配置文件

/JSFLoginDemo/WebRoot/WEB-INF/faces-config.xml

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

[复制代码](javascript:void(0);)

<?xml version='1.0' encoding='UTF-8'?>  
  
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig\_1\_2.xsd"  
 version="1.2">  
   
 <managed-bean>  
 <managed-bean-name>loginBean</managed-bean-name>  
 <managed-bean-class>  
 com.qdu.sun.jsf.LoginBean  
 </managed-bean-class>  
 <managed-bean-scope>request</managed-bean-scope>  
 <managed-property>  
 <property-name>username</property-name>  
 <property-class>java.lang.String</property-class>  
 <value></value>  
 </managed-property>  
 <managed-property>  
 <property-name>password</property-name>  
 <property-class>java.lang.String</property-class>  
 <value></value>  
 </managed-property>  
 </managed-bean>  
 <navigation-rule>  
 <from-view-id>/login.jsp</from-view-id>  
 <navigation-case>  
 <from-outcome>success</from-outcome>  
 <to-view-id>/success.jsp</to-view-id>  
 </navigation-case>  
 <navigation-case>  
 <from-outcome>fail</from-outcome>  
 <to-view-id>/fail.jsp</to-view-id>  
 </navigation-case></navigation-rule>  
 <navigation-rule>  
 <from-view-id>/success.jsp</from-view-id>  
 </navigation-rule>  
 <navigation-rule>  
 <from-view-id>/fail.jsp</from-view-id>  
 </navigation-rule></faces-config>

[复制代码](javascript:void(0);)

前端页面

/JSFLoginDemo/WebRoot/login.jsp

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

[复制代码](javascript:void(0);)

<%@ page language="java" pageEncoding="ISO-8859-1"%>  
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>  
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>  
  
<%  
 String path = request.getContextPath();  
 String basePath = request.getScheme() + "://"  
 + request.getServerName() + ":" + request.getServerPort()  
 + path + "/";  
%>  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
 <head>  
 <base href="<%=basePath%>">  
  
 <title>My JSF 'login.jsp' starting page</title>  
  
 <meta http-equiv="pragma" content="no-cache">  
 <meta http-equiv="cache-control" content="no-cache">  
 <meta http-equiv="expires" content="0">  
 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  
 <meta http-equiv="description" content="This is my page">  
 <!--  
 <link rel="stylesheet" type="text/css" href="styles.css">  
 -->  
  
 </head>  
  
 <body>  
 <f:view>  
 <br>  
 <h:form>  
  
 <h:panelGrid columns="3">  
  
 <h:outputLabel for="username" value="User Name:" />  
  
 <h:inputText id="username" value="#{loginBean.username}" required="true" />  
  
 <h:message for="username" />  
  
 <h:outputLabel for="password" value="Password:" />  
  
 <h:inputSecret id="password" value="#{loginBean.password}" required="true" />  
  
 <h:message for="password" />   
  
 </h:panelGrid>  
  
 <h:panelGrid>  
  
 <h:panelGroup>  
  
 <h:commandButton value="Login" action="#{loginBean.login}" />  
  
 </h:panelGroup>  
  
 </h:panelGrid>  
  
 </h:form>  
  
  
  
 </f:view>  
 </body>  
</html>

[复制代码](javascript:void(0);)

后台代码

/JSFLoginDemo/src/com/qdu/sun/jsf/LoginBean.java

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif代码

[复制代码](javascript:void(0);)

/\*\*  
 \*   
 \*/  
package com.qdu.sun.jsf;  
  
/\*\*  
 \* @author sun1  
 \*  
 \*/  
public final class LoginBean extends Object {  
  
 /\*\*  
 \*   
 \*/  
 private String password;  
 /\*\*  
 \*   
 \*/  
 private String username;  
 /\*\*  
 \* @return the password  
 \*/  
 public String getPassword() {  
 return password;  
 }  
 /\*\*  
 \* @param password the password to set  
 \*/  
 public void setPassword(String password) {  
 this.password = password;  
 }  
 /\*\*  
 \* @return the username  
 \*/  
 public String getUsername() {  
 return username;  
 }  
 /\*\*  
 \* @param username the username to set  
 \*/  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String login() {   
  
 if ((username == null) || (username.length() < 1))   
  
 return "fail";   
  
 if ((password == null) || (password.length() < 1))   
  
 return "fail";   
  
 if ((username.equals("孙更新")) && (password.equals("123")))   
  
 return "success";   
  
 else   
  
 return "fail";   
  
 }   
  
  
}

[复制代码](javascript:void(0);)

JSF有两组标准的标签，即core标签（共有18个，平均每个标签有2.8个属性）和html标签（25个，平均每个标签有26.2个属性）。core标签的具体的作用还没看到例子，不过从“The core library is the poor stepchild of the HTML library—the former exists entirely to support the latter.”这句话可以看到，它不如html标签那么重要。

那么来看看html标签，html标签大致可分为：Inputs，Outputs，Commands，Selection，Others。看上去和struts的html标签也差不多。

标签 说明

view 创建顶层视图

subview 创建视图的子视图

facet 向组件添加 facet

param 向组件添加参数

attribute 向组件添加属性

actionListener 在父组件上注册操作监听器

valueChangeListener 向父组件注册值变更监听器

converter 注册任意转换器

convertDateTime 注册日期时间转换器

convertNumber 注册数量转换器

validateLength 验证组件值的长度

validateDoubleRange 验证组件值的双精度范围

validateLongRange 验证组件值的长整型范围

validator 向组件添加验证器

loadBundle 加载资源包

verbatim 向页面中添加标签

selectItem 指定 UISelectOne 或 UISelectMany 组件的一个项目

selectItems 指定 UISelectOne 或 UISelectMany 组件的多个项目

组件 标签 说明

UIForm form 表示 HTML表单元素

UIInput InputText 单行文本输入控件

inputTextarea 多行文本输入控件

InputSecret 密码输入控件

inputHidden 隐藏字段

UIOutput outputLabel 显示文本

outputLink 链接到另一页面或某页面上的特定位置

outputFormat 显示格式化的信息

outputText 当行文本输入

UICommand commandButton 表示一个按钮

commandLink 表示一个链接

UISelectOne SelectOneListbox 单选列表框

selectOneMenu 单选菜单

selectOneRadio 单选按钮集

UISelectBoolean selectBooleanCheckbox 单个复选框

UISelectMany selectManyListbox 复选框集

selectManyCheckbox 多行选择列表框

selectManyMenu 多选菜单

UIPanel panelGrid 表示一 个 HTML 表格

panelGroup 组合一系列组件

UIMessage message 显示一个组件的最近使用的消息

messages 显示所有消息

UIGraphic graphicImage 显示一个图像

UIData dataTable 功能丰富的表格控件

UIColumn column 表示表格中的列

表单 h:form 产生 HTML form 元素