**JSF Document**

**Prerequisites:**

1. JSF requires JDK 1.7 or higher.
2. Java 7 or higher
3. To use JSF you need:

* JSF Implementation (in the form of the JSF jars)
* The JSTL tags library
* A Java runtime environment
* A web-container to use JSF in (for example JBoss)

**Installation:**

1. Download JBoss Wildfly 8 plugin.
2. Install eclipse kepler version.

**JSF Main features**

JSP has the following main features:

* JSP is based on the Model-View-Controller concept
* JSP has a stateful UI component model, e.g. each component is aware of its data
* JSF separates the functionality of a component from the display of the component. The renderer is responsible of displaying the component for a certain client. This renderer can get exchanged. The standard renderer for JSF components is the HTML renderer.
* JSP support listeners on UI components
* JSP support data validation, data binding and data conversion between the UI and the model

## JSF configuration files

### 1. Overview

JSF is based on the following configuration files:

* web.xml - General web application configuration file
* faces-config.xml - Contains the configuration of the JSF application.

### 2.web.xml

JSF requires the central configuration list *web.xml* in the directory WEB-INF of the application. This is similar to other web-applications which are based on servlets.

You must specify in *web.xml* that a "FacesServlet" is responsible for handling JSF applications. "FacesServlet" is the central controller for the JSF application. "FacesServlet" receives all requests for the JSF application and initializes the JSF components before the JSP is displayed.

### 3. faces-config.xml

"faces-config.xml" allows to configure the application, managed beans, convertors, validators, and navigation.

## Installation

### 1. Eclipse

For JSP development you need the Eclipse WTP and an installed Tomcat. See [**Installation of Eclipse WTP and Tomcat**](http://www.vogella.com/tutorials/EclipseWTP/article.html).

### 2. JSF library

A JSF library is required. We will later use Eclipse to download and install the Apache MyFaces JSF implementation during project creation.

### 3. JSLT library

Download the JSLT library from [**https://jstl.dev.java.net/**](https://jstl.dev.java.net/).

## First JSF project

**Tell Eclipse about java version**

Window🡪Preferences🡪Java🡪Installed JREs🡪press “add”, choose “Standard VM” navigate to JDK folder. (not “bin” subdirectory)

Eg. C:\Java\jdk 1.7.0\_55

**Creating a New Web Project**

File🡪New🡪Dynamic Web Project

Step 1:Create the class file and add it to the managed bean in faces-config.xml.

Step 2: Create a new jsp file in the Web content.

Step 3:Using export option create a war file.

Step 4:Then deploy it in a JBoss Wildfly 8.

Step 5:Then run in the localhost.

**To see the Error in the server log:**

In JBoss wildfly 8🡪standalone🡪log🡪server.log. Right click and open with textpad.

Then stop the server by using ctrl+c shortkey in keyboard in the server command prompt.

Delete all the contents in the server.log file which is opened using textpad.Then save it.

Then again start the server by pressing n in the command prompt.

Then do deployment for your application. Saw the error in the server.log in textpad.

**Use of annotations.**

If we use annotations in the java then we need not to create the separate managed bean in the faces-config.xml.

Then change the url pattern in the web.xml by /\* instead of /faces/\*.

**Creating the class file:**

By using Eclipse create a java file. It automatically creates the class file when saving it.

Then this class file stored in side eclipse workspace inside the project🡪build🡪classes🡪package name🡪class file.

Then copy this class file with packages and place it in the WEB-INF🡪classes🡪paste it.

Then inside the WEB-INF🡪lib🡪 Only jar files.

Then inside the WEB-INF🡪faces-config.xml,web.xml

Then inside the folder only WEB-INF and .xhtml or .jsp files.

**JSF-Spring Integration**

## Step 1. Add DelegatingVariableResolver

Add a variable-resolver entry in faces-config.xml to point to spring class **DelegatingVariableResolver**.

<faces-config>

<application>

<variable-resolver>

org.springframework.web.jsf.DelegatingVariableResolver

</variable-resolver>

...

</faces-config>

## Step 2. Add Context Listeners

Add **ContextLoaderListener** and **RequestContextListener** listener provided by spring framework in web.xml

<web-app>

...

<!-- Add Support for Spring -->

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<listener>

<listener-class>

org.springframework.web.context.request.RequestContextListener

</listener-class>

</listener>

...

</web-app>

## Step 3. Define Dependency

Define bean(s) in applicationContext.xml which will be used as dependency in managed bean

<beans>

<bean id="messageService"

class="com.tutorialspoint.test.MessageServiceImpl">

<property name="message" value="Hello World!" />

</bean>

</beans>

## Step 4. Add Dependency

**DelegatingVariableResolver** first delegates value lookups to the default resolver of the JSF and then to Spring's WebApplicationContext. This allows one to easily inject spring based dependencies into one's JSF-managed beans.

We've injected messageService as spring based dependency here

<faces-config>

...

<managed-bean>

<managed-bean-name>userData</managed-bean-name>

<managed-bean-class>com.tutorialspoint.test.UserData</managed-bean-class>

<managed-bean-scope>request</managed-bean-scope>

<managed-property>

<property-name>messageService</property-name>

<value>#{messageService}</value>

</managed-property>

</managed-bean>

</faces-config>

## Step 5. Use Dependency

//jsf managed bean

public class UserData {

//spring managed dependency

private MessageService messageService;

public void setMessageService(MessageService messageService) {

this.messageService = messageService;

}

public String getGreetingMessage(){

return messageService.getGreetingMessage();

}

}

## Example Application

Let us create a test JSF application to test spring integration.

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Create a project with a name *helloworld* under a package *com.tutorialspoint.test* as explained in the *JSF - First Application* chapter. |
| 2 | Modify *pom.xml* as explained below. |
| 3 | Create *faces-config.xml* in *WEB-INF* folder as explained below. |
| 4 | Modify *web.xml* as explained below. |
| 5 | Create *applicationContext.xml* in *WEB-INF* folder as explained below. |
| 6 | Create *MessageService.java* under package *com.tutorialspoint.test* as explained below. |
| 7 | Create *MessageServiceImpl.java* under package *com.tutorialspoint.test* as explained below. |
| 8 | Create *UserData.java* under package *com.tutorialspoint.test* as explained below. |
| 9 | Modify *home.xhtml* as explained below. Keep rest of the files unchanged. |
| 10 | Compile and run the application to make sure business logic is working as per the requirements. |
| 11 | Finally, build the application in the form of war file and deploy it in Apache Tomcat Webserver. |
| 12 | Launch your web application using appropriate URL as explained below in the last step. |

**JSF with Spring Integration:**

**Step 1**:In the view page when we press the submit button it goes to the method specified in the controller class.

**Step 2:** From the controller it calls to the managed bean and utility service classes.

**Step 3:** In the utility service class it performs some operations output the result into the controller class.

**Step 4:** From the controller class returns the response view page.

**Example**:

**Web.xml:**

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

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

id="WebApp\_ID" version="2.5">

<welcome-file-list>

<welcome-file>login.xhtml</welcome-file>

</welcome-file-list>

<context-param>

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

<param-value>/WEB-INF/applicationContext.xml</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

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

</servlet-mapping>

<servlet-mapping>

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

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

</servlet-mapping>

</web-app>

**Faces-config.xml:**

<?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\_2\_0.xsd"

version="2.0">

<application>

<variable-resolver>

org.springframework.web.jsf.DelegatingVariableResolver

</variable-resolver>

</application>

<navigation-rule>

<from-view-id>/login.xhtml</from-view-id>

<navigation-case>

<from-outcome>welcome</from-outcome>

<to-view-id>/welcome.xhtml</to-view-id>

</navigation-case>

</navigation-rule>

<managed-bean>

<managed-bean-name>userBean</managed-bean-name>

<managed-bean-class>com.tutorialspoint.User</managed-bean-class>

<managed-bean-scope>request</managed-bean-scope>

</managed-bean>

<managed-bean>

<managed-bean-name>loginController</managed-bean-name>

<managed-bean-class>com.tutorialspoint.LoginController</managed-bean-class>

<managed-bean-scope>request</managed-bean-scope>

<managed-property>

<property-name>user</property-name>

<property-class>com.tutorialspoint.User</property-class>

<value>#{userBean}</value>

</managed-property>

<managed-property>

<property-name>myConnection</property-name>

<value>#{myConnection}</value>

</managed-property>

</managed-bean>

</faces-config>

**applicationContext.xml:**

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

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:annotation-config />

<context:component-scan base-package="com.tutorialspoint">

</context:component-scan>

<bean id="myConnection" class="com.tutorialspoint.MyConnection"></bean>

</beans>

**Login.xhtml:**

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

<!DOCTYPE html >

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

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

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

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

<h:head>

<title>Login Page</title>

</h:head>

<h:body>

<h:form>

<h:panelGrid columns="2">

<h:outputLabel value="username"></h:outputLabel>

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

<h:outputLabel value="password"></h:outputLabel>

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

</h:panelGrid>

<h:commandButton action="#{loginController.sayHello}" value="submit"></h:commandButton>

</h:form>

</h:body>

</html>

**Welcome.xhmtl:**

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

<!DOCTYPE html >

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

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

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

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

<h:head>

<title> Welcome page</title>

</h:head>

<h:body>

<f:view>

<h:form>

<h:outputLabel value="#{userBean.message}"></h:outputLabel>

</h:form>

</f:view>

</h:body>

</html>

**LoginController.java**

**package** com.tutorialspoint;

**import** org.springframework.stereotype.Controller;

**import** com.tutorialspoint.MyConnection;

@Controller

**public** **class** LoginController {

**private** User user;

**private** MyConnection myConnection;

**public** User getUser()

{

**return** user;

}

**public** **void** setUser(User user)

{

**this**.user = user;

}

**public** MyConnection getMyConnection()

{

**return** myConnection;

}

**public** **void** setMyConnection(MyConnection myConnection)

{

**this**.myConnection = myConnection;

}

**public** String sayHello()

{

myConnection.login(user);

**return** "welcome";

}

}

**User.java**

**package** com.tutorialspoint;

**public** **class** User {

**private** String name;

**private** String pass;

**private** String message;

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name)

{

**this**.name = name;

}

**public** String getPass()

{

**return** pass;

}

**public** **void** setPass(String pass)

{

**this**.pass = pass;

}

**public** String getMessage()

{

**return** message;

}

**public** **void** setMessage(String message)

{

**this**.message = message;

}

}

**MyConnection.java**

package com.tutorialspoint;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class MyConnection {

public void login(User user)

{

try{

Boolean B=false;

Connection conn=null;

ResultSet rst=null;

Statement stmt=null;

String QS="Select login\_id,password FROM dbo.subscriber";

try

{

Class.forName("net.sourceforge.jtds.jdbc.Driver");

conn=DriverManager.getConnection("jdbc:jtds:sqlserver://10.10.0.155:1433/CDC","cdc","cdcadmin");

if(conn!=null)

{ System.out.println("Connection is Established");

stmt=conn.createStatement();

rst=stmt.executeQuery(QS);

while(rst.next())

{

String loginid = rst.getString("login\_id");

String password = rst.getString("password");

if(user.getName().equals(loginid) && user.getPass().equals(password))

{

B=true;

continue;

}

}

}

stmt.close();

rst.close();

conn.close();

}

catch(Exception sle)

{

sle.printStackTrace();

}

if(B)

{

user.setMessage("Welcome " +user.getName());

System.out.println("Login Successful");

}

else

{

user.setMessage("Incorrect Login Id");

System.out.println("Login Failed.");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}