Prev Next

Chapter 3. Getting Started with RichFaces

This chapter describes all necessary actions and configurations that should be done for plugging the RichFaces components into a JSF appplication. The description relies on a simple JSF with RichFaces application creation process from downloading the libraries to running the application in a browser. The process of application creation described here is common and does not depend on used IDE.

3.1. Downloading the RichFaces

The latest release of RichFaces components is available for download at JBoss RichFaces Downloads area at JBoss community. Binary files (uploaded there in `*.bin.zip` or `*.bin.tar.gz` archives) contains compiled, ready-to-use version of RichFaces with set of basic skins.

To start with RichFaces in computer file system create new folder with name "RichFaces", download and unzip the archive with binaries there.

For those who want to download and compile the RichFaces by themselfs there is an article at JBoss community that describes the RichFaces repository's structure overview and some aspects of working with it.

3.2. Simple JSF application with RichFaces

"RichFaces Greeter"—the simple application—is hello-world like application but with one difference: the world of RichFaces will say "Hello!" to user first.

Create standard JSF 1.2 project with all necessary libraries; name the project "Greeter" and follow the decription.

3.2.1. Adding RichFaces libraries into the project

Go to the folder with unzipped earlier RichFaces binary files and open `lib` folder. This folder contains three `*.jar` files with API, UI and implementation libraries. Copy that "jars" from `lib` folder to `WEB-INF/lib` folder of "Greeter" JSF application.

3.2.2. Registering RichFaces in web.xml

After RichFaces libraries where added into the project it is necessary to register them in project `web.xml` file. Add following lines in `web.xml`:

```
...
<!-- Plugging the "Blue Sky" skin into the project -->
<context-param>
  <param-name>org.richfaces.SKIN</param-name>
  <param-value>blueSky</param-value>
</context-param>

<!-- Making the RichFaces skin spread to standard HTML controls -->
<context-param>
    <param-name>org.richfaces.CONTROL_SKINNING</param-name>
    <param-value>enable</param-value>
</context-param>

<!-- Defining and mapping the RichFaces filter -->
<filter>
  <display-name>RichFaces Filter</display-name>
  <filter-name>richfaces</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>

<filter-mapping>
  <filter-name>richfaces</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
...
```

For more information on how to work with RichFaces skins read "Skinnabilty" chapter.

Finally the `web.xml` should look like this:

```
<?xml version="1.0"?>
<web-app version="2.5"
         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-app_2_5.xsd">
<display-name>Greeter</display-name>

<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>server</param-value>
</context-param>

<context-param>
  <param-name>org.richfaces.SKIN</param-name>
  <param-value>blueSky</param-value>
</context-param>

<context-param>
    <param-name>org.richfaces.CONTROL_SKINNING</param-name>
    <param-value>enable</param-value>
</context-param>

<filter>
  <display-name>RichFaces Filter</display-name>
  <filter-name>richfaces</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>

<filter-mapping>
  <filter-name>richfaces</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>

<listener>
  <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>

<!-- Faces Servlet -->
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<!-- Faces Servlet Mapping -->
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>

<login-config>
  <auth-method>BASIC</auth-method>
  </login-config>
</web-app>
```

3.2.3. Managed bean

The "RichFaces Greeter" application needs a managed bean. In project `JavaSource` folder create a new managed bean with name `user` in `demo` package and paste there the following simple code:

```
package demo;

public class user {
  private String name="";
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
}
```

3.2.4. Registering bean in faces-cofig.xml

With the next step the `user` bean should be registered in `faces-config.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="1.2"
              xmlns="http://java.sun.com/xml/ns/javaee"
              xmlns:xi="http://www.w3.org/2001/XInclude"
              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">
  <managed-bean>
    <description>UserName Bean</description>
    <managed-bean-name>user</managed-bean-name>
    <managed-bean-class>demo.user</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
      <property-name>name</property-name>
      <property-class>java.lang.String</property-class>
      <value/>
    </managed-property>
  </managed-bean>
</faces-config>
```

3.2.5. RichFaces Greeter index.jsp

The "RichFaces Greeter" application has only one JSP page. Create `index.jsp` page in root of `WEB CONTENT` folder and add there following code:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<!-- RichFaces tag library declaration -->
<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>

<html>
    <head>
        <title>RichFaces Greeter</title>
    </head>
    <body>
        <f:view>
            <a4j:form>
                <rich:panel header="RichFaces Greeter" style="width: 315px">
                    <h:outputText value="Your name: " />
                    <h:inputText value="#{user.name}" >
                        <f:validateLength minimum="1" maximum="30" />
                    </h:inputText>

                    <a4j:commandButton value="Get greeting" reRender="greeting" />

                    <h:panelGroup id="greeting" >
                        <h:outputText value="Hello, " rendered="#{not empty user.name}" />
                        <h:outputText value="#{user.name}" />
                        <h:outputText value="!" rendered="#{not empty user.name}" />
                    </h:panelGroup>
                </rich:panel>
            </a4j:form>
        </f:view>
    </body>
</html>
```

The application uses three RichFaces components: **<rich:panel>** is used as visual container for information; **<a4j:commandButton>** with built-in Ajax support allows rendering a greeting dynamically after a response comes back and **<a4j:form>** helps the button to perform the action.

Note, that the RichFaces tag library should be declared on each JSP page. For XHTML pages add following lines for tag library declaration:

```
<xmlns:a4j="http://richfaces.org/a4j">
<xmlns:rich="http://richfaces.org/rich">
```

That's it. Run the application on server. Point your browser to `index.jsp` page in browser: `http://localhost:8080/Greeter/index.jsf`

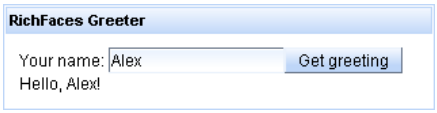

Figure 3.1. "RichFaces Greeter" application

3.3. Relevant Resources Links

JBoss Developer Studio comes with a tight integration with RichFaces component framework. Following links might be useful for those who already u RichFaces for developing applications and those who wish to improve their development process:

- "Rich Components" chapter in "Getting Started with JBoss Developer Studio Guide" describes how to add RichFaces components into a CRUD applic
- "JBoss Tools Palette" chapter in "Visual Web Tools Reference Guide" describes advantages that gives Tools Pallete (comes together with JBDS) for pages creation processs including RichFaces applications;
- "RichFaces Toolkit for developing Web application" video tutorial demonstrates some aspects of interaction with JBoss Developer Studio whil RichFaces.

Read also the quick overview to "Practical RichFaces " book by Max Katz at his blog.

Prev Top of page Front page Next

Chapter 2. Technical Requirements Chapter 4. Settings for different environments