Developer Guide

# Developing applications using RichFaces *(draft)*

by Sean Rogers (Red Hat)

DRAFT

DRAFT

DRAFT

DRAFT

# Introduction

The RichFaces framework is a rich component library for JavaServer Faces (JSF). It allows integration of Ajax capabilities into enterprise web application development without needing to use JavaScript.

RichFaces leverages several parts of the JSF2 framework including lifecycle, validation, conversion facilities, and management of static and dynamic resources. The RichFaces framework includes components with built-in Ajax support and a customizable look-and-feel that can be incorporated into JSF applications.

RichFaces provides a number of advantages for enterprise web application development:

- Build on the benefits of JavaServer Faces with support for Ajax. RichFaces is fully integrated into the JSF lifecycle: it uses the action and value change listeners, and invokes server-side validators and converters during the Ajax request-response cycle.
- Extend Ajax capability in existing JSF applications. The core Ajax library (`a4j`) adds extra Ajax functionality into existing pages, such that additional JavaScript code is unnecessary and existing components do not need to be replaced with Ajax ones. RichFaces enables page-wide Ajax support instead of the traditional component-wide support, and events can be defined on the page for invoking an Ajax requests and JSF Component Tree synchronization.
- Create complex application views using out-of-the-box components. The RichFaces user interface (UI) library (`rich`) contains components for adding rich interactive features to JSF applications. It extends the RichFaces framework to include a large set of Ajax-enabled components that come with extensive skinning support. Additionally, the RichFaces framework is designed to be used seamlessly with other 3d-party libraries on the same page, so you have more options for developing applications.
- Write your own customized rich components with built-in Ajax support. The Component Development Kit (CDK), used for the RichFaces UI library creation, includes a code-generation facility and a templating facility using XHTML (extended hyper-text markup language) syntax.
- Package dynamic resources with application Java classes. Ajax functionality in RichFaces extends support for the management of different resources, such as pictures, JavaScript code, and CSS stylesheets. The resource framework makes it possible to pack dynamic resources along with the code for any custom components.
- Generate binary resources on the fly. The resource framework can generate images, sounds, **Microsoft Excel** spreadsheets, and more during run-time.
- Create a modern rich user-interface with skinning technology. RichFaces provides a skinning feature that allows you to define and manage different color schemes and other parameters of the look and feel. It is possible to access the skin parameters from JSP code and Java code during run-time. RichFaces comes packaged with a number of skins to get you started, but you can also easily create your own customized skins too.

DRAFT

# Getting started with RichFaces

Follow the instructions in this chapter to configure the RichFaces framework and get started with application development. RichFaces applications can be developed using JBoss Tools, as described in *Section 2.3, "Creating a project with JBoss Tools"*; or using Maven, as described in *Section 2.4, "Creating a project with Maven"*.

If you have existing projects that use a previous version of RichFaces, refer to the *RichFaces Migration Guide*.

## 2.1. Technical Requirements

The minimum technical requirements needed to get started with RichFaces are outlined below.

- Java Development Kit (JDK) 1.5 or higher
- JBoss Tools 3.1
- A JavaServer Faces 2 (JSF 2) implementation
- An application server, such as JBoss Application Server 6 or Apache Tomcat 6.
- A web browser, such as Firefox 3.5 or Internet Explorer 7

RichFaces supports additional products not listed here. Refer to *???* for a full list of technical requirements and supported envrionments, browsers, and tools.

## 2.2. Setting up RichFaces

Follow the instructions in this section to set up the RichFaces framework and begin building applications.

1. **Download RichFaces archive**

   Download RichFaces from the JBoss RichFaces Downloads area at *http://www.jboss.org/richfaces/download.html*. The binary files (available in `.bin.zip` or `.bin.tar.gz` archives) contain a compiled, ready-to-use version of RichFaces with a set of basic skins.

   - **Compiling from source**

     Instead of downloading the pre-compiled binaries, you can download the source files and compile them yourself. Refer to *???* for further instructions.

2. **Unzip archive**

   Create a new directory named `RichFaces`, then unzip the archive containing the binaries there.

## 2.3. Creating a project with JBoss Tools

Follow the procedure in this section to create a new RichFaces application with JBoss Tools.

1. **Create a new project**

   Create a new project based on the JSF 2 environment. In JBoss Tools, select **File**+**New** → **JSF Project** from the menu. Name the project, select **JSF 2** from the **JSF Environment** drop-down box, and click the **Finish** button to create the project.

2. **Add the RichFaces libraries to the project**

   Add `core-ui.jar`, `richfaces-api.jar`, and `richfaces-impl.jar` into your project by copying them from the location where you unzipped the RichFaces archive to the `WebContent/WEB-INF/lib/` directory of your project in **JBoss Tools**.

3. **Reference the tag libraries**

   The RichFaces tag libraries need to be referenced on each XHTML page in your project:

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                xmlns:a4j="http://richfaces.org/a4j"
                xmlns:rich="http://richfaces.org/rich">
   ...
</ui:composition>
```

You are now ready to begin constructing your RichFaces applications. RichFaces components can be dragged and dropped into your application's XHTML pages from the RichFaces palette in JBoss Tools, shown in *Figure 2.1, “RichFaces palette in JBoss Tools”*

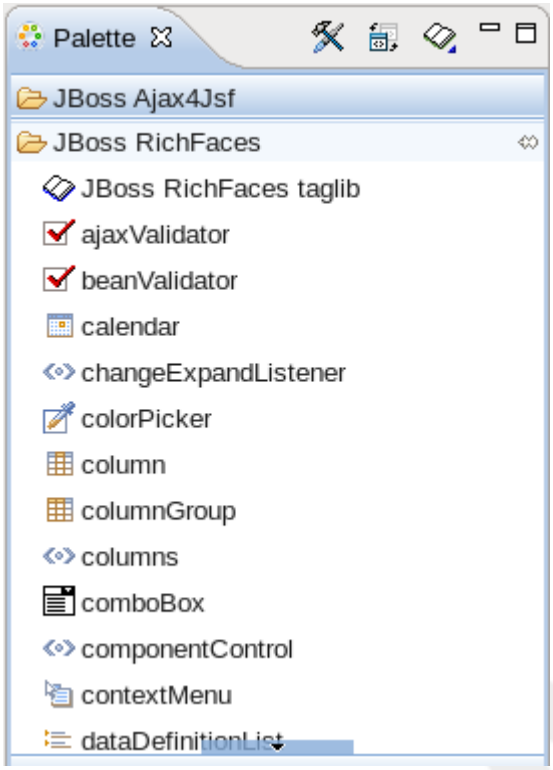


**Figure 2.1. RichFaces palette in JBoss Tools**

## 2.4. Creating a project with Maven

Apache Maven is a build automation and project management tool for Java projects. Follow the instructions in this section to create a Maven project for RichFaces.

### 2.4.1. Setting up Maven

Maven can be downloaded and installed from Apache's website at *http://maven.apache.org/download.html*. Version 2.2.1 is recommended.

Once Maven has been installed, no further configuration is required to begin building Maven projects.

### 2.4.2. Using the RichFaces project archetype

A Maven archetype is a template for creating projects. Maven uses an archetype to generate a directory structure and files for a particular project, as well as creating `pom.xml` files that contain build instructions.

The RichFaces Component Development Kit includes a Maven archetype named `richfaces-archetype-simpleapp` for generating the basic structure and requirements for a RichFaces application project. Maven can obtain the archetype from the JBoss repository at `https://repository.jboss.org/nexus/content/groups/public/`. The archetype is also included with

the RichFaces source code. Follow the procedure in this section to generate a new Maven-based RichFaces project using the archetype.

1. **Add required repository**

   The details for the JBoss repository need to be added to Maven so it can access the archetype. Add a profile in the *`maven_installation_folder`*`/conf/settings.xml` file under the `<profiles>` element:

```
<profiles>
  ...
  <profile>
    <id>jboss-public-repository</id>
    <repositories>
      <repository>
        <id>jboss-public-repository-group</id>
        <name>JBoss Public Maven Repository Group</name>
        <url>https://repository.jboss.org/nexus/content/groups/public/</url>
        <layout>default</layout>
        <releases>
          <enabled>true</enabled>
          <updatePolicy>never</updatePolicy>
        </releases>
        <snapshots>
          <enabled>true</enabled>
          <updatePolicy>never</updatePolicy>
        </snapshots>
      </repository>
    </repositories>
    <pluginRepositories>
      <pluginRepository>
        <id>jboss-public-repository-group</id>
        <name>JBoss Public Maven Repository Group</name>
        <url>https://repository.jboss.org/nexus/content/groups/public/</url>
        <layout>default</layout>
        <releases>
          <enabled>true</enabled>
          <updatePolicy>never</updatePolicy>
        </releases>
        <snapshots>
          <enabled>true</enabled>
          <updatePolicy>never</updatePolicy>
        </snapshots>
      </pluginRepository>
```

```
    </pluginRepositories>
  </profile>
</profiles>
```

The profile then needs to be activated in the `<activeProfiles>` element:

```
<activeProfiles>
  <activeProfile>jboss-public-repository</activeProfile>
</activeProfiles>
```

2. **Generate the project from the archetype**

The project can now be generated with the `richfaces-archetype-simpleapp` archetype. Create a new directory for your project, then run the following Maven command in the directory:

```
mvn archetype:generate -DarchetypeGroupId=org.richfaces.archetypes
 -DarchetypeArtifactId=richfaces-archetype-simpleapp -
DarchetypeVersion=4.0.0-SNAPSHOT -DgroupId=org.docs.richfaces -
DartifactId=new_project
```

The following parameters can be used to customize your project:

`-DgroupId`
: Defines the package for the Managed Beans

`-DartifactId`
: Defines the name of the project

The command generates a new RichFaces project with the following structure:

```
new_project
 ### pom.xml
 ### src
  ### main
      ### java
      #   ### org
      #       ### docs
      #           ### richfaces
      #               ### RichBean.java
      ### webapp
          ### index.xhtml
          ### templates
          #   ### template.xhtml
          ### WEB-INF
```

```
                ### faces-config.xml
                ### web.xml
```

3. **Add test dependencies (optional)**

   Your root directory of your project contains a project descriptor file, `pom.xml`. If you wish to include modules for test-driven JSF development, add any dependencies for the tests to the `pom.xml` file. For full details on how to use the jsf-test project, refer to *http://community.jboss.org/wiki/TestDrivenJSFDevelopment* [http://community.jboss.org/docs/DOC-13155].

4. **Build the project**

   Build the project from the command line by entering the `mvn install` command.

   The `BUILD SUCCESSFUL` message indicates the project has been assembled and is ready to import into an IDE (integrated development environment), such as JBoss Tools.

5. **Import the project into an IDE**

   Import the Maven project into your IDE. For **Eclipse** and **JBoss Tools**, you can import the project using the M2Eclipse plug-in.

   To install the plug-in, choose **Help** → **Install New Software** from the menu. Type `Maven` to locate the **Maven Integration for Eclipse Update Site** entry, then type `Maven` in the filter to show the available plug-ins. Follow the prompts to install the **Maven Integration for Eclipse** plug-in.

   With the plug-in installed, open the importing wizard by choosing **File** → **Import** from the menu. Select **Maven** → **Existing Maven Projects** as the import source and choose the `pom.xml` file for your project.

Your project is now ready to use. Once components and functionality have been added, you can run the application on a server and access it through a web browser at the address `http://localhost:8080/jsf-app/`.

## 2.5. Using RichFaces in existing JSF2 projects

RichFaces can be added to existing JSF2 projects by adding references to the new RichFaces libraries. Refer to *Step 2* and *Step 3* in *Section 2.3, "Creating a project with JBoss Tools"* for details.