JBoss AS 6.0 JSF Guide

JSF with JBoss Application Server 6

by Stan Silvert

Part I. Introduction and Quick Start

Introduction

In the past, using a JEE application server meant using the JSF implementation that ships with it. However, there are subtle differences between JSF implementations. Applications written for Mojarra don't always run well on MyFaces. There are sometimes backward compatibility issues between JSF specification levels. An application written for JSF 1.2 won't always run on JSF 2.0.

JBoss AS6 is designed for maximum flexibility in JSF deployments. With JBoss AS6, you can use the default JSF implementation, use a secondary JSF implementation, or bundle your own JSF implementation with the WAR. You can have different applications in the same server instance that use different implementations. Also, you can create your own JSF Configurations that include a JSF implementation, extra libraries, and configuration parameters. Then assign the configuration to one or more applications.

In this guide, we'll step through a simple JSF example. Then we will go through all the powerful deployment options for JSF applications.

Quick Start

In this chapter, we demonstrate the world's simplest JSF "Hello World" application.

2.1. Create your WAR structure

Go to your `JBOSS_HOME/server/default/deploy` directory and create these two subdirectories:

- `hellojsf.war`
- `hellojsf.war/WEB-INF`

2.2. Create a minimal web.xml

This `web.xml` only needs the minimum declarations shown below. Place the file in `/WEB-INF`.

```
<?xml version="1.0"?>
<web-app>
</web-app>
```

> **Note**
>
> As shown, you don't necessarily need to declare a FacesServlet or mappings in web.xml. If you leave this out, JBoss AS6 will add it automatically with default mappings as demonstrated at the end of this chapter.

2.3. Create a minimal faces-config.xml

This `faces-config.xml` only needs the minimum declarations shown below. Place the file in `/WEB-INF`.

```
<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">

</faces-config>
```

The faces-config.xml is only there to signal to JBoss AS that this is a JSF application. There are many other ways that JBoss AS6 could recognize this as a JSF application. This is explained in detail in chapter 3.

2.4. Create Your JSF Markup

We will use a single facelet. Create the file `index.xhtml` and put it in your `deploy/hellojsf.war` directory.

We use a little JSF2/EL 2.2 trick to avoid the need for a backing bean. We can grab the input value directly from the request object using a parameterized EL expression.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

    <f:view>
      <h:form id="form1">
        <h:outputText value="Enter Your Name:"/>
        <h:inputText id="name"/>
        <h:commandButton value="Submit" />
      </h:form>
      <h:outputText rendered="#{not empty request.getParameter('form1:name')}"
                value=" Hello #{request.getParameter('form1:name')}"/>
    </f:view>

</html>
```

2.5. Run the application

Now we're done! We only needed three files and two of those were just placeholders.

Start JBoss AS6 and put any of the following URLs into your browswer:

- *http://localhost:8080/hellojsf/index.jsf*
- *http://localhost:8080/hellojsf/index.faces*
- *http://localhost:8080/hellojsf/faces/index.xhtml*

Part II. Configuration

Deploying Your JSF Applications

In this chapter, we cover all the deployment options for your JSF applications.

3.1. The JSF Deployer

JSF integration for JBoss AS6 has been re-written to take advantage of the JBoss deployer architecture. So instead of having JSF tightly coupled to the Servlet container, it is now an independent deployer that adds JSF to your WAR when needed.

The JSF Deployer is located in the `deployers/jsf.deployer` directory. By default, JBoss AS6 ships with three JSF implementations located in the `jsf.deployer/Mojarra-2.0`, `jsf.deployer/MyFaces-2.0`, and `jsf.deployer/Mojarra-1.2` directories. These directories contain *JSF Configurations*.

> **Note**
>
> A JSF Configuration is more than just the implementation jars. It can contain supporting jars such as parsers and JSF component libraries. It also contains initialization and configuration settings that are applied to your application when the JSF Configuration is added to your WAR deployment.

Inside the `jsf.deployer/META-INF` directory you will find a file called `jsf-integration-deployer-jboss-beans.xml`. You can use this file for advanced configuration of the deployer, which we will describe in some of the sections that follow.

3.2. How the JSF Deployer Recognizes your JSF Application

In past versions of JBoss AS, *every* web application had a JSF implementation and its supporting jars placed on its classpath. In addition, every web application went through at least some of the JSF initialization process - even when it didn't use JSF. With JBoss AS6, JSF jars are only added to the classpath when needed.

When a web application is deployed, the JSF Deployer determines if it is a JSF application. It recognizes a web application if any of the following are true:

- A FacesServlet is declared in `WEB-INF/web.xml` or a `web-fragment.xml`
- A `faces-config.xml` file is found in `WEB-INF`
- A `faces-config.xml` file is found in `META-INF` of some jar in `WEB-INF/lib`
- A `*.faces-config.xml` file is found in `META-INF` of some jar in `WEB-INF/lib`

- The `javax.faces.CONFIG_FILES` context param is declared in `WEB-INF/web.xml` or a `web-fragment.xml`
- The `org.jboss.jbossfaces.JSF_CONFIG_NAME` context param is declared in `WEB-INF/web.xml` or a `web-fragment.xml`
- "alwaysAddJSF" is set to true in `jsf-integration-deployer-jboss-beans.xml`

3.3. Auto-adding of the JSF FacesServlet

If the JSF Deployer determines that a WAR is a JSF application, but `javax.faces.webapp.FacesServlet` is not already declared as a servlet, the deployer will add an instance of this servlet for you.

If it adds the FacesServlet, it will also add the following mappings for it:

- /faces/*
- *.jsf
- *.faces

> **Note**
> MyFaces and pre-2.0 versions of Mojarra still assume that you will declare a FacesServlet in your web.xml. So if you are using those versions you can not depend on auto-adding.

3.4. Using a Non-standard FacesServlet

Though it is not recommended, some applications use a non-standard servlet to control JSF services. You can configure the JSF Deployer to recognize a non-standard servlet as a JSF application. Edit the file `jsf.deployer/META-INF/jsf-integration-deployer-jboss-beans.xml` and add your servlet to the facesServlets property.

In this example, we add `org.foo.MyWrapperFacesServlet`. When an application is deployed with this servlet it will be recognized as a JSF application.

```
<bean name="JSFImplManagementDeployer">


   <!--
     * Specify the servlet classes that signal this deployer to add JSF to a WAR.
   -->
   <property name="facesServlets">
```

```
    <collection elementClass="java.lang.String">
      <value>javax.faces.webapp.FacesServlet</value>
      <value>org.foo.MyWrapperFacesServlet</value>
    </collection>
  </property>
```

3.5. Bundling JSF Inside Your WAR

Some containers such as Tomcat 6 require you to bundle a JSF implementation in the `WEB-INF/lib` directory of your WAR. If you would like to use such a WAR with JBoss AS6 then you can signal the JSF Deployer to ignore your WAR and let it use the bundled JSF version.

To do that, just specify the WAR_BUNDLES_JSF context param in your web.xml file like this:

```
<context-param>
    <param-name>org.jboss.jbossfaces.WAR_BUNDLES_JSF_IMPL</param-name>
    <param-value>true</param-value>
</context-param>
```

> **Note**
>
> This context-param was available in earlier versions of JBoss AS. However, it only worked when bundling MyFaces and when using the default classloader configuration. Now in AS6 you can use this context-param any time you want to bundle your own JSF impl.

3.6. Changing the JSF Configuration for your WAR

JBoss AS ships with three JSF Implementations, Mojarra 1.2, Mojarra 2.0, and MyFaces 2.0. By default, JSF applications will use Mojarra 2.0. While most JSF 1.2 applications will run on JSF 2.0 without changes, there are a few rare instances where this is not the case. Also, when migrating to JBoss AS6 from AS5, you might want to first use the older JSF implementation and "ease into" the world of JSF 2.0 later.

If you look at the `deployers/jsf.deployer` directory you will see the JSF configurations that ship with JBoss AS6. To tell your application to use one of these JSF configurations, add this to your `web.xml`:

```
<context-param>
    <param-name>org.jboss.jbossfaces.JSF_CONFIG_NAME</param-name>
```

```
    <param-value>Mojarra-1.2</param-value>
 </context-param>
```

3.7. Adding a New JSF Configuration

A new JSF Configuration is useful when you want to add a new JSF implementation to JBoss AS or you just want to enhance an implementation with extra jars such as component libraries. This can save you from bundling the same jars over and over in your WARs.

It's also useful for testing the same application against different JSF implementations, library versions, and configurations. You can create a JSF Configuration and then apply it to your WAR with a a simple context param.

A JSF Configuration consists of some jars and a special `web.xml` file. When a JSF Configuration is added to a WAR by the JSF Deployer, the jars are added to the classpath and the elements in the `web.xml` file are activated. To add your own JSF Confguration, just create the directory structure below. This is usually done in the jsf.deployer directory:

- `jsf.deployer/MyJSFConfig/jsf-libs`
- `jsf.deployer/MyJSFConfig/META-INF`

Place your jar files in `/jsf-libs` and place your `web.xml` in `/META-INF`. When your JSF Configuration is activated for a WAR, all jars in the jsf-libs directory will be added to the classpath.

The elements in your special `META-INF/web.xml` file will also be added to your WAR. This can help you configure the JSF implementation and component libraries. However, note that only a few web.xml elements are allowed in this file. These elements are *servlet, servlet-mapping, filter, filter-mapping, listener, and context-param*. All other web.xml elements are currently ignored, but we may support more in the future.

3.8. Activating a New JSF Configuration

To allow the JSF Deployer to recognize your JSF Configuration, you will need to edit `deployers/jsf.deployer/META-INF/jsf-integration-deployer-jboss-beans.xml`:

```
<property name="jsfConfigurations">
  <map keyClass="java.lang.String" valueClass="java.lang.String">
    <entry>
      <key>Mojarra-1.2</key>
      <value>${jboss.server.home.url}deployers/jsf.deployer/Mojarra-1.2</value>
    </entry>
    <entry>
      <key>Mojarra-2.0</key>
```

```
      <value>${jboss.server.home.url}deployers/jsf.deployer/Mojarra-2.0</value>
    </entry>

    <entry>
      <key>MyJSFConfig</key>
       <value>${jboss.server.home.url}deployers/jsf.deployer/MyJSFConfig</value>
     </entry>
  </map>
</property>
```

```
<bean name="JSFUrlIntegrationDeployer-MyJSFConfig">
    <property name="JSFImplName">
      <value>MyJSFConfig</value>
    </property>
    <property name="JSFImplManagementDeployer">
      <inject bean="JSFImplManagementDeployer"/>
    </property>
</bean>
```

JSF and Serialization

Serialization can be one of the most costly operations in a JSF application. This is especially true when using client side state-saving, but it can also come into play when you use server-side state saving as well.

4.1. Using JBoss Serialization

JBoss AS6 ships with a serialization provider that allows you to use the JBossSerialization project with Mojarra 1.2 and 2.0. While your results will vary, using JBossSerialization may boost performance, especially with older versions of Java.

To enable JBossSerialization in Mojarra JSF, set the following context parameter in your `web.xml` file.

```
<context-param>
  <param-name>com.sun.faces.serializationProvider</param-name>
    <param-value>org.jboss.web.jsf.integration.serialization.JBossSerializationProvider</param-value>
</context-param>
```

Part III. Reference

Reference

5.1. JSF Standard Context Params

Table 5.1.

Context Param	JSF Spec	Description	Default Value
javax.faces.CONFIG_FILES	1.2 and 2.0	Comma-delimited list of faces config files.	null
javax.faces.DEFAULT_SUFFIX	1.2 and 2.0	Change the default suffix for JSP views.	.jsp
javax.faces.LIFECYCLE_ID	1.2 and 2.0	ID for alternate Lifecycle implementations.	null
javax.faces.STATE_SAVING_METHOD	1.2 and 2.0	"server" or "client"	server
javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE	2.0	Controls if DateTimeConverter instances use the system timezone (if true) or GMT (if false).	false
javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE	2.0	Controls if DateTimeConverter instances use the system timezone (if true) or GMT (if false).	false
javax.faces.DISABLE_FACELET_JSF_VIEWHANDLER	2.0	Disables the built-in Facelet ViewHandler. Useful for applications that use legacy Facelets implementation.	false
javax.faces.FACELETS_LIBRARIES	2.0	Semicolon-separated list of paths to Facelet tag libraries.	null
facelets.LIBRARIES	2.0	Semicolon-separated list of paths to Facelet tag libraries. Used for backward-compatibility	null

Context Param	JSF Spec	Description	Default Value
		with legacy Facelets implementation.	
javax.faces.FACELETS_BUFFER_SIZE	2.0	The buffer size set on the response.	-1 (no assigned buffer size)
facelets.BUFFER_SIZE	2.0	The buffer size set on the response. Used for backward-compatibility with legacy Facelets implementation.	-1 (no assigned buffer size)
javax.faces.DECORATORS	2.0	Semicolon-delimited list of TagDecorator implementations. See javadoc for javax.faces.view .facelets.TagDecorator.	null
facelets.DECORATORS	2.0	Semicolon-delimited list of TagDecorator implementations. Used for backward-compatibility with legacy Facelets implementation.	null
javax.faces.FACELETS _REFRESH_PERIOD	2.0	Time in seconds that facelets should be checked for changes since last request. A value of -1 disables refresh checking.	implementation-specific
facelets.REFRESH_PERIOD	2.0	Time in seconds that facelets should be checked for changes since last request. A value of -1 disables refresh checking. Used for backward-compatibility with legacy Facelets implementation.	implementation-specific
javax.faces.FACELETS _RESOURCE_RESOLVER	2.0	An implementation of javax.faces .view.facelets .ResourceResolver. See javadoc for details.	null
facelets.RESOURCE_RESOLVER	2.0	An implementation of javax.faces .view.facelets	null

Context Param	JSF Spec	Description	Default Value
		.ResourceResolver. See javadoc for details. Used for backward-compatibility with legacy Facelets implementation.	
javax.faces.FACELETS_SKIP_COMMENTS	2.0	If true, strip XML comments out of Facelets before delivering to the client.	false
facelets.SKIP_COMMENTS	2.0	If true, strip XML comments out of Facelets before delivering to the client. Used for backward-compatibility with legacy Facelets implementation.	false
javax.faces.FACELETS_SUFFIX	2.0	Set the suffix for Facelet xhtml files.	.xhtml
facelets.SUFFIX	2.0	Set the suffix for Facelet xhtml files. Used for backward-compatibility with legacy Facelets implementation.	.xhtml
javax.faces.FACELETS_VIEW_MAPPINGS	2.0	Semicolon-separated list of Facelet files that don't use the default facelets suffix.	null
facelets.VIEW_MAPPINGS	2.0	Semicolon-separated list of Facelet files that don't use the default facelets suffix. Used for backward-compatibility with legacy Facelets implementation.	null
javax.faces.FULL_STATE_SAVING_VIEW_IDS	2.0	Semicolon-separated list of view IDs that must save state using the JSF 1.2-style state saving.	null
javax.faces.INTERPRET_EMPTY_STRING_SUBMITTED_VALUES_AS_NULL	2.0	If true, consider empty UIInput values to be null instead of empty string.	false
javax.faces.PARTIAL_STATE_SAVING	2.0	If true, use the JSF2 partial state saving for views.	false, if WEB-INF/faces-

Context Param	JSF Spec	Description	Default Value
			config.xml does not declare JSF 2.0 schema. true, otherwise
javax.faces.PROJECT_STAGE	2.0	Set the project stage to "Development", "UnitTest", "SystemTest", or "Production".	Production
javax.faces.VALIDATE_EMPTY_FIELDS	2.0	If "true", validate null and empty values. If "auto" validate when JSR-303 Bean Validation is enabled (in AS6 it is enabled by default).	auto
javax.faces.validator.DISABLE_DEFAULT_BEAN_VALIDATOR	2.0	If "true", disable JSR-303 Bean Validation.	false

5.2. Mojarra Context Params

These context params are only valid when using Mojarra JSF.

Table 5.2.

Context Param	Impl Ver	Description	Default Value
com.sun.faces.numberOfViewsInSession	1.2 and 2.0	For server state-saving, how many views, per logical view, can be stored in the session before oldest is removed?	15
com.sun.faces.numberOfLogicalViews	1.2 and 2.0	For server state-saving, how many logical views are allowed before oldest is removed?	15
com.sun.faces.preferXHTML	1.2 and 2.0	Set xhtml as the content type for browsers that support it.	false
com.sun.faces.compressViewState			false

Context Param	Impl Ver	Description	Default Value
	1.2 and 2.0	Compress the view after serialization but before encoding.	
com.sun.faces.disableVersionTracking	1.2 and 2.0	If true, don't allow JSF 1.1 implementations of certain interfaces.	false
com.sun.faces.sendPoweredByHeader	1.2 and 2.0	If true, send a header with the JSF spec level.	true
com.sun.faces.verifyObjects	1.2 and 2.0	If true, verify all JSF artifacts (such as managed beans) can be instantiated during initialization.	false
com.sun.faces.validateXml	1.2 and 2.0	If true, perform XML validation on config files.	false
com.sun.faces.displayConfiguration	1.2 and 2.0	If true, log the values of all Mojarra and JSF context params.	false
com.sun.faces.injectionProvider	1.2 and 2.0	Replace the default InjectionProvider implementation. By default this is set by JBoss AS6.	org.jboss.web .jsf.integration .injection .JBossDelegatingInjectionProvider
com.sun.faces.injectionProvider	1.2 and 2.0	Replace the default SerializationProvider implementation.	null
com.sun.faces.responseBufferSize	1.2 and 2.0	Buffer size for writing most generated content.	4096
com.sun.faces. clientStateWriteBufferSize	1.2 and 2.0	Buffer size for writing client state.	8192
com.sun.faces.compressJavaScript	1.2 and 2.0	Remove whitespace from javascript used in standard JSF components.	true

Context Param	Impl Ver	Description	Default Value
com.sun.faces.externalizeJavaScript	1.2 and 2.0	Allow browsers to cache javascript used in standard JSF components.	false
com.sun.faces.enableJSStyleHiding	1.2 and 2.0	Hide javascript from older browser implementations.	false
com.sun.faces.writeStateAtFormEnd	1.2 and 2.0	Controls if view state is written after opening a form tag (false) or closing a form tag (true).	true
com.sun.faces. enableLazyBeanValidation	1.2 and 2.0	If false, examine managed beans at startup. Otherwise, validate when referenced/ created.	true
com.sun.faces. enabledLoadBundle11Compatibility	1.2 and 2.0	Preserve JSF 1.1 behavior of f:loadBundle. *See issue here.* [https:// / javaserverfaces.dev.java.net/ issues/ show_bug.cgi?id=577]	false
com.sun.faces.clientStateTimeout	1.2 and 2.0	Time in seconds that client state is considered valid. If a request is received after timeout expired, ViewExpiredException is thrown.	null
com.sun.faces.serializeServerState	1.2 and 2.0	If true, serialize server-side component state.	false
com.sun.faces. enableViewStateIdRendering	1.2 and 2.0	If false, don't render id attribute on javax.faces.ViewState hidden field. *See issue here.* [https:// javaserverfaces.dev.java.net/	true

Context Param	Impl Ver	Description	Default Value
		issues/show_bug.cgi?id=433]	
com.sun.faces.enableScriptsInAttributeValues	1.2 and 2.0	If false, don't allow attribute values with "javascirpt:" or "script:".	true
com.sun.faces.disableUnicodeEscaping	1.2 and 2.0	"false", "true", or "auto". If "auto", escaping depends on response encoding. *See issue here.* [https://javaserverfaces.dev.java.net/issues/show_bug.cgi?id=751]	false
com.sun.faces.developmentMode	1.2	If true, reload Groovy and faces-config files when edited.	false
com.sun.faces.enableMultiThreadedStartup	1.2	If false, don't create worker threads at startup.	true
com.sun.faces.enableThreading	2.0	If false, don't create worker threads at startup.	false
com.sun.faces.resourceBufferSize	2.0	Read buffer size, in bytes, used by the default ResourceHandler implementation.	2048
com.sun.faces.defaultResourceMaxAge	2.0	Time in milliseconds that a resource from the ResourceHandler will be cached via an "Expires" response header. No caching if ProjectStage is "Development".	604800
com.sun.faces.resourceUpdateCheckPeriod	2.0	Frequency in minutes to check for changes to webapp artifacts that contain resources. 0 means never check for changes. -1 means always perform a new lookup when finding resources.	5

Context Param	Impl Ver	Description	Default Value
com.sun.faces.compressableMimeTypes	2.0	Comma-seperated list of compressable mime types. No compression if ProjectStage is "Development"	null
com.sun.faces.expressionFactory	2.0	Expression factory implementation class.	com.sun.el .ExpressionFactoryImpl
com.sun.faces.duplicateJARPattern	1.2 and 2.0	Regular expression to determine if two URLs point to the same jar. Set by JBoss JSF Deployer.	^tmp\d+(\S*\.jar)
com.sun.faces.faceletFactory	2.0	Set the FaceletFactory impl class.	null
com.sun.faces.enableHtmlTagLibValidator	2.0	Enable validation of standard Html TagLibs.	false
com.sun.faces.enableCoreTagLibValidator	2.0	Enable validation of standard Core TagLibs.	false
com.sun.faces. registerConverterPropertyEditors	2.0	If true, allow EL Coercion to use JSF Custom converters.	false
com.sun.faces.enableGroovyScripting	2.0	If true, allow Groovy.	false
com.sun.faces. generateUniqueServerStateIds	2.0	If true, generate random server state Ids. If false, create Ids sequentially.	true
com.sun.faces. autoCompleteOffOnViewState	2.0	If false, don't use autocomplete="off" when saving view state.	true
com.sun.faces.allowTextChildren	2.0	If true, allow children of h:inputText and h:outputText to be rendered. In 1.2, they would always be rendered before the value of tag. In 2.0, they will not be rendered at all unless this flag is set.	false

5.3. JBoss JSF Context Params

These context params are explained more in Chapter 3.

Table 5.3.

Context Param	Description	Default Value
org.jboss.jbossfaces.JSF_CONFIG_NAME	The name of the JSF Configuration to use for this WAR.	Mojarra-2.0
org.jboss.jbossfaces.WAR_BUNDLES_JSF_IMPL	If true, allow WAR to use the JSF implementation found in WEB-INF/lib.	false