Red Hat JBoss Web Framework Kit 2.7
Snowdrop Guide

---

for use with Red Hat JBoss Web Framework Kit

Red Hat Customer Content Services

Red Hat JBoss Web Framework Kit 2.7 Snowdrop Guide

for use with Red Hat JBoss Web Framework Kit

Red Hat Customer Content Services

Legal Notice

Abstract

This book is a guide for using Snowdrop with JBoss Web Framework Kit.

Table of Contents

Chapter 1. Snowdrop Deprecation

Warning

Snowdrop is deprecated from the release of JBoss Web Framework Kit 2.7.0. This means that no new features will be introduced to the Snowdrop framework in JBoss Web Framework Kit 2.7.0 or later. As a tier 1 framework, Red Hat continues to support Snowdrop and provide fixes under our standard support terms and conditions. For more information about the Red Hat support policy, see https://access.redhat.com/support/policy/updates/jboss_notes/.

Chapter 2. Introduction to Red Hat JBoss Web Framework Kit

2.1. About Red Hat JBoss Web Framework Kit

Red Hat JBoss Web Framework Kit is a set of enterprise-ready versions of popular open source frameworks. Together, these frameworks provide a solution for developing light and rich Java-based web applications.

JBoss Web Framework Kit comprises enterprise distributions of JBoss community web application frameworks and tested third party frameworks. These leading frameworks support fast and easy client-side and server-side application development and testing, with frameworks including RichFaces, jQuery, jQuery Mobile, Hibernate Search, Spring, and Arquillian.

The breadth of JBoss Web Framework Kit frameworks provides choice for Java application development. Each framework has been tested and certified for use in applications deployed to Red Hat JBoss Enterprise Application Platform, Red Hat JBoss Enterprise Web Server, or Red Hat OpenShift JBoss EAP cartridge. Inclusion in JBoss Web Framework Kit ensures stable versions of frameworks are available over long-term enterprise product life cycles, with regular releases for fixes and nonintrusive feature updates. Further, Red Hat JBoss Developer Studio (an Eclipse-based development environment) provides integrated project templates, quickstarts and tooling for many of the JBoss Web Framework Kit frameworks.

For the complete list of the frameworks composing JBoss Web Framework Kit and the certified platform and framework configurations, see https://access.redhat.com/site/articles/112543 on the Red Hat Customer Portal.

2.2. About the JBoss Web Framework Kit Tiers

The frameworks composing JBoss Web Framework Kit are categorized in four distinct tiers. A description of each tier and the associated Red Hat support is detailed here.

Tier 1 - Included Components

These are components that are based wholly or partly on open source technologies that support broad collaboration and where Red Hat maintains a leadership role; as such Red Hat is able to support these components and provide upgrades and fixes under our standard support terms and conditions.

Tier 2 - Tested Frameworks

These are third party frameworks where Red Hat does not have sufficient influence and does not provide upgrades and fixes under our standard support terms and conditions. Commercially reasonable support is provided by Red Hat Global Support Services for these frameworks.

Tier 3 - Frameworks in Tested Examples

These are third party frameworks where Red Hat does not have sufficient influence and does not provide upgrades and fixes under our standard support terms and conditions. Red Hat supports the examples these frameworks are used in and the generic use cases that these examples intend to demonstrate.

Tier 4 - Confirmed Frameworks

These are third party frameworks that do not receive any support from Red Hat, but Red Hat verifies that the frameworks run successfully on Red Hat JBoss Enterprise Application Platform. Frameworks and versions not listed here have not been explicitly tested and

Platform. Frameworks and versions not listed here have not been explicitly tested and certified, and thus may be subject to support limitations.

For a list of JBoss Web Framework Kit frameworks by tier, see https://access.redhat.com/site/articles/112543 on the Red Hat Customer Portal.

2.3. About the JBoss Web Framework Kit Distribution

The frameworks composing JBoss Web Framework Kit are distributed from a range of sources and in a variety of formats:

- The component frameworks are available from the Red Hat Customer Portal. They are distributed in two alternative formats: a binary for each framework or together as one Maven repository. In addition, the source code for each framework is provided for inspection.
- The third party frameworks are not distributed by Red Hat and each must be obtained from its own source.

A number of defined Maven JBoss stacks are provided as part of the JBoss Web Framework Kit distribution. All of the BOMs defining the JBoss stacks are available in the Maven repository `.zip` file available to download from the Red Hat Customer Portal or from http://www.jboss.org/developer-materials/ on the JBoss Developer Framework website.

An extensive set of examples are also provided as part of the JBoss Web Framework Kit distribution:

- TicketMonster is a moderately complex application demonstrating a number of the JBoss Web Framework Kit frameworks working together.
- Quickstarts and Maven archeypes illustrate subsets of the JBoss Web Framework Kit frameworks used to create simple applications.
- RichFaces, Snowdrop and Seam demonstrations showcase the power of each framework in web application development.

All of these examples are available from the Red Hat Customer Portal, with TicketMonster, the quickstarts, and the Maven archetypes also available from http://www.jboss.org/developer-materials/ on the JBoss Developer Framework website.

Chapter 3. Introduction to Snowdrop

Snowdrop is a utility package that contains JBoss-specific extensions to the Spring Framework. These extensions are either:

- extensions to Spring Framework classes that can be used wherever the generic implementations provided by the framework do not integrate correctly with JBoss Enterprise Application Platform.
- extensions for deploying and running Spring applications with the JBoss Enterprise Application Platform.

The Snowdrop distribution contains two types of artifacts:

Utility archives

Libraries that contain utility classes and can be packaged in applications

Server-side components

Pre-packaged components that can be installed in the application server as subsystems.

The server-side components may include some of the utility libraries, where they are needed by the deployer or, respectively, subsystem.

Snowdrop provides the following utility libraries:

`snowdrop-vfs.jar`

A library that contains the support classes for resource scanning (scanning the classpath for bean definitions, or using "classpath*:"-style patterns).

`snowdrop-weaving.jar`

A library that contains the support classes for load-time weaving.

`snowdrop-interceptors.jar`

Contains utility classes used internally by the Spring deployer. It is not intended for direct use by developers.

`snowdrop-namespace.jar`

Custom namespace for easier configuration of beans that need access to JBoss internals (JMX server locator, JCA activation spec and resource adapter for JMS endpoints)

Snowdrop provides the following pre-packaged deployers and subsystems:

`snowdrop-subsystem-as7.zip`

This is a packaged distribution of the Spring subsystem for Red Hat JBoss Enterprise Application Platform 6, which bootstraps and registers the application contexts to be used by your Java EE applications.

Chapter 4. Component Usage

This chapter details how to use each of the components included in Snowdrop.

4.1. VFS-enabled Application Contexts

Note

From Spring 3.0 onward, the **`ApplicationContext`** implementations shipped with the Spring framework are VFS-compatible. The components described in this section are included with Snowdrop to provide backwards compatibility, but are not necessarily required.

The **`snowdrop-vfs.jar`** library supports resource scanning in the JBoss Virtual File System (VFS). It must be included in Spring-based applications that use classpath and resource scanning.

When the Spring framework performs resource scanning, it assumes that resources are either from a directory or a packaged JAR, and treats any URLs it encounters accordingly.

This assumption is not correct for the JBoss VFS, so Snowdrop provides a different underlying resource resolution mechanism by amending the functionality of the **`PathMatchingResourcePatternResolver`**.

This is done by using one of two **`ApplicationContext`** implementations provided by **`snowdrop-vfs.jar`**:

`org.jboss.spring.vfs.context.VFSClassPathXmlApplicationContext`

Replaces the Spring **`org.springframework.context.support.ClassPathXmlApplicationContext`**.

`org.jboss.spring.vfs.context.VFSXmlWebApplicationContext`

Replaces the Spring **`org.springframework.web.context.support.XmlWebApplicationContext`**.

In many cases, the **`VFSClassPathXmlApplicationContext`** is instantiated on its own, using:

```
ApplicationContext context =
new VFSClassPathXmlApplicationContext("classpath:/context-definition-
file.xml");
```

The **`XmlWebApplicationContext`** is not instantiated directly. Instead, it is bootstrapped by either the **`ContextLoaderListener`** or the **`DispatcherServlet`**. Both classes have configuration options that allow users to replace the default application context type with a custom application context type.

To change the type of application context created by the **`ContextLoaderListener`**, add the **`contextClass`** parameter as shown in the following example code:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath*:spring-contexts/*.xml</param-value>
</context-param>
<context-param>
```

```
  <param-name>contextClass</param-name>
  <param-value>
    org.jboss.spring.vfs.context.VFSXmlWebApplicationContext
  </param-value>
</context-param>
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

To change the type of application context created by the **`DispatcherServlet`**, use the same **`contextClass`** parameter on the **`DispatcherServlet`** definition as shown:

```
<servlet>
  <servlet-name>spring-mvc-servlet</servlet-name>
  <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/mvc-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>contextClass</param-name>
    <param-value>
      org.jboss.spring.vfs.context.VFSXmlWebApplicationContext
    </param-value>
  </init-param>
</servlet>
```

Important: `ZipException`

If you encounter the **`ZipException`** when attempting to start the application, you need to replace the default **`ApplicationContext`** with one of the VFS-enabled implementations.

```
Caused by: java.util.zip.ZipException: error in opening zip file
...
at
org.springframework.core.io.support.PathMatchingResourcePatternResol
ver
.doFindPathMatchingJarResources(PathMatchingResourcePatternResolver.
java:448)
```

4.2. The JBoss Custom Namespace

Snowdrop includes snowdrop-namespace.jar which adds Spring namespace support for Red Hat JBoss Enterprise Application Platform. The goal of this custom namespace is to simplify the development of Spring applications that run on JBoss, by reducing the amount of proprietary code and improving portability.

The amount of proprietary code is reduced by replacing bean definitions that include references to specific JBoss classes with namespace-based constructs. All the knowledge about the proprietary classes is encapsulated in the namespace handlers.

The applications are more portable because certain proprietary classes may change when upgrading to a different version of the application server. In such cases, the runtime will be detected automatically by Snowdrop which will set up beans using the classes that are appropriate for that specific runtime.

Set up the custom namespace as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:jboss="http://www.jboss.org/schema/snowdrop"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.jboss.org/schema/snowdrop
http://www.jboss.org/schema/snowdrop/snowdrop.xsd">
</beans>
```

4.2.1. Accessing the Default JBoss MBean Server

Access the default MBean server of JBoss Enterprise Application Platform as follows:

```
<jboss:mbean-server/>
```

The bean will be installed with the default id 'mbeanServer'. If necessary, developers can specify a different bean name:

```
<jboss:mbean-server id="customName"/>
```

Important

The location of the MBean server has changed between versions of JBoss Enterprise Application Platform, so using the following configuration fails with deployment errors:

```
<bean id="mBeanServer" class="org.jboss.jmx.util.MBeanServerLocator"
factory-method="locateJBoss" />
```

The workaround for this issue is to use this configuration instead:

```
<bean id="mbeanServer"
class="org.springframework.jmx.support.MBeanServerFactoryBean">
  <property name="locateExistingServerIfPossible" value="true" />
</bean>
```

4.2.2. JCA/JMS Support Beans

Spring JMS message listeners (including message-driven POJOs) can use a JCA-based MessageListenerContainer. The configuration of a JCA-based listener container in Spring requires the setup of a number of beans based on application-server specific classes. Using the JBoss custom namespace, set up the ResourceAdapter and ActivationSpec configuration as follows:

```
<jboss:activation-spec-factory id="activationSpecFactory"/>
  <jboss:resource-adapter id="resourceAdapter"/>
```

This can be further used in a JCA message listener configuration:

```
<jms:jca-listener-container resource-adapter="resourceAdapter"
acknowledge="auto"
   activation-spec-factory="activationSpecFactory">
    <jms:listener destination="/someDestination" ref="messageDrivenPojo"
      method="pojoHandlerMethod"/>
</jms:jca-listener-container>
```

4.3. Load-time weaving

Note

From Spring 3.0 onward, load-time weaving on Red Hat JBoss Enterprise Application Platform is supported out of the box. The component described in this section can be used to facilitate backward compatibility, but configuring a custom load-time weaver is not required when using Spring 3.0 or later.

Load-time weaving support is provided by the **`snowdrop-weaving.jar`** library.

To perform load-time weaving for the application classes in Spring (either for using load-time support for AspectJ or for JPA support), the Spring framework needs to install its own transformers in the classloader. For JBoss Enterprise Application Platform, a classloader-specific **`LoadTimeWeaver`** is necessary.

Define the **`JBossLoadTimeWeaver`** in the Spring application context as shown here:

```
<context:load-time-weaver
weaver-class="org.jboss.instrument.classloading.JBossLoadTimeWeaver"/>
```

4.4. The Spring Deployer

The Spring deployer allows you to bootstrap a Spring application context, bind it in JNDI, and use it to provide Spring-configured business object instances.

4.4.1. JBoss + Spring + EJB 3.0 Integration

Snowdrop contains a JBoss deployer that supports Spring packaging in Red Hat JBoss Enterprise Application Platform. This means it is possible to create JAR archives with a **`META-INF/jboss-spring.xml`** file to have the Spring bean factories deploy automatically.

EJB 3.0 integration is also supported. Spring beans created in such archive deployments can be injected directly into an EJB by using the **`@Spring`** annotation.

4.4.2. Installation on Red Hat JBoss Enterprise Application Platform 6

4.4.2.1. Manual Installation of Snowdrop

Warning

- Spring 2.5 is deprecated from the release of JBoss Web Framework Kit 2.4.0. This means that Spring 2.5 will continue to be tested and supported in JBoss Web Framework Kit 2.x. Note that subsequent Spring 2 releases will not be tested or supported in JBoss Web Framework Kit.
- Spring 3.0.x and 3.1.x are deprecated from the release of JBoss Web Framework Kit 2.5.0. This means that Spring 3.0.x and 3.1.x will continue to be tested and supported in JBoss Web Framework Kit 2.x. Note that Spring 3.2.x and later are not deprecated and continue to be tested and supported in JBoss Web Framework Kit.

To install the Snowdrop Deployment subsystem, navigate to the **`snowdrop-subsystem-as7`** directory in the JBoss Web Framework Kit distribution. Create the subsystem and Spring modules in JBoss Enterprise Application Platform by copying the contents of the **`module-deployer`** directory and one of the **`module-spring-2.5/`**, **`module-spring-3/`**, **`module-spring-3.1/`**, **`module-spring-3.2/`**, **`module-spring-4.0/`** or **`module-spring-4.1/`** directories in the **`$JBOSS_HOME/modules/system/add-ons/snowdrop`** directory of your JBoss Enterprise Application Platform installation.

The above step creates two modules inside JBoss Enterprise Application Platform:

`org.jboss.snowdrop:main`

The module that contains the JBoss Enterprise Application Platform subsystem.

`org.springframework.spring:snowdrop`

A module that contains the Spring JARs required by Snowdrop. It can contain Spring 2.5, Spring 3, Spring 3.1, Spring 3.2, Spring 4.0 or Spring 4.1 JARs, depending on the previously chosen version. Users may add other JARs to the module, case in which they need to adjust the **`module.xml`** file accordingly. It is a dependency of **`org.jboss.snowdrop:main`**

The Web Framework Kit distribution does not contain Spring archives, so you will need to install them separately. Copy one of the files of corresponding versions from Maven Central.

Spring 2.5.6.SEC03

- aspectjrt.jar
- aspectjweaver.jar
- aopalliance.jar
- spring-aop.jar
- spring-beans.jar

- spring-core.jar
- spring-context.jar
- spring-context-support.jar
- spring-web.jar

Spring 3.0.7.RELEASE and Spring 3.1.4.RELEASE

- aspectjrt.jar
- aspectjweaver.jar
- aopalliance.jar
- spring-aop.jar
- spring-asm.jar
- spring-beans.jar
- spring-core.jar
- spring-expression.jar
- spring-context.jar
- spring-context-support.jar
- spring-web.jar

3.2.13.RELEASE, 4.0.9.RELEASE, and 4.1.4.RELEASE

- aspectjrt.jar
- aspectjweaver.jar
- aopalliance.jar
- spring-aop.jar
- spring-beans.jar
- spring-core.jar
- spring-expression.jar
- spring-context.jar
- spring-context-support.jar
- spring-web.jar

The final step in the installation is to change **`$JBOSS_HOME/standalone/configuration/standalone.xml`** by including **`<extension module="org.jboss.snowdrop"/>`** inside the **`<extensions>`** element, as well as including **`<subsystem xmlns="urn:jboss:domain:snowdrop:1.0"/>`** inside the **`<profile>`** element.

4.4.2.2. Automatic Installation of Snowdrop

For easy installation of Snowdrop module, use the Snowdrop installer. The installer copies Snowdrop and Spring jars to their appropriate location within **${JBOSS_HOME}/modules** directory. The installer also creates new **$JBOSS_HOME/standalone/configuration/standalone-snowdrop.xml** file based on **$JBOSS_HOME/standalone/configuration/standalone.xml** to register the snowdrop extension and subsystem. You can run JBoss Enterprise Application Platform with this new configuration using **$JBOSS_HOME/bin/standalone.sh --server-config=standalone-snowdrop.xml**.

To install Snowdrop using the installer, on the command line, navigate to the **snowdrop-module-installer** directory in the Web Framework Kit distribution and execute the following command:

```
mvn package -DJBOSS_HOME=/path/to/jboss_home
```

By default, the installer installs Snowdrop 3.1.1.Final-redhat-1 and Spring 4.1.4.RELEASE. To install a different version, execute the following command:

```
mvn package -P${desired-spring-version} -DJBOSS_HOME=/path/to/jboss_home
-Dversion.snowdrop=${desired-snowdrop-version}
```

For details of the supported configurations click here.

4.4.3. Defining the JNDI name

You can specify the JNDI name explicitly by putting it in the description element of the Spring XML.

```
<beans>
  <description>BeanFactory=(MyApp)</description>
  ...
  <bean id="springBean" class="example.SpringBean"/>
</beans>
```

MyApp will be used as the JNDI name in this example.

4.4.4. Parent Bean factories

Sometimes the deployed Spring bean factory must be able to reference beans deployed in another Spring deployment. This can be done by declaring a parent bean factory in the description element in the Spring XML, as follows:

```
<beans>
<description>BeanFactory=(AnotherApp) ParentBeanFactory=
(MyApp)</description>
...
</beans>
```

4.4.5. Injection into EJBs

Once an **ApplicationContext** has been successfully bootstrapped, the Spring beans defined in it can be used for injection into EJBs. To do this, the EJBs must be intercepted with the **SpringLifecycleInterceptor**, as in the following example:

```
@Stateless
@Interceptors(SpringLifecycleInterceptor.class)
```

```
public class InjectedEjbImpl implements InjectedEjb
{
 @Spring(bean = "springBean", jndiName = "MyApp")
 private SpringBean springBean;

 /* rest of the class definition ommitted */
}
```

In this example, the EJB **`InjectedEjbImpl`** will be injected with the bean named **`springBean`**, which is defined in the **`ApplicationContext`**.

Migration Notes

This section of the guide will track any breaking changes introduced in new releases, and identify any steps required to accommodate those changes in your application.

A.1. Migrating to Snowdrop 3.1.1.Final-redhat-1

If you are using the Snowdrop API, specifically the snowdrop-deployers.jar, and you want to use the latest version of Snowdrop from the JBoss Web Framework Kit distribution then refactor your project to use the snowdrop-interceptors.jar. Some older deprecated classes were removed but the classes used in the API have not changed, so this does not require any other modifications.

In case you do not have the possibility to rebuild your project you can replace or remove the appropriate packages directly from the archive. The archive then becomes compatible with the latest Snowdrop API.

A.2. Migrating from Snowdrop

Snowdrop is deprecated from the release of JBoss Web Framework Kit 2.7.0. As a result, you may be looking to migrate any existing applications that currently use Snowdrop off this deprecated framework.

To assist you in the migration process, an example migrated application is distributed with this release of JBoss Web Framework Kit. The Sportsclub application provides a real-world-inspired example of integrating Spring with JBoss Enterprise Application Platform using Snowdrop. A second version of this application, Snowdropless-Sportsclub, is also provided that demonstrates the same real-world example integrating Spring with JBoss Enterprise Application Platform without using Snowdrop. The **`Snowdrop-removal.md`** distributed with the latter example details the necessary steps taken to achieve the migration off Snowdrop.

Revision History

Revision 2.7.0-4	**Fri Jan 23 2015**	**Michelle Murray**

WFKDOC-121: Online distribution info added

Revision 2.7.0-2	**Thu Jan 22 2015**	**Michelle Murray**

JBQA-11348: QE feedback incorporated

Revision 2.7.0-1	**Mon Jan 19 2015**	**Michelle Murray**

Generated for 2.7.0 release