Building Controls

Table of contents

1 Overview....2
2 JAR Projects....2
3 Integrating into Existing Projects....3
4 Source Control.... 3

1. Overview

Beehive Controls are a JavaBean resource abstraction layer for enterprise Java applications. The source artifacts for Controls are simply .java files that can be built along with an enterprise Java module / project or as part of a standalone JAR file. This document describes how to build Controls in each of these project types.

Because Controls use JSR 175 annotations as a metadata model, Controls require the annotation processing features available in J2SE 5.0. As such, a special Ant build macro is used to process annotations and produce any build-time configuration artifacts (such as XML files and deployment descriptors) and binary class files. The API for the Controls annotation processor is described here. This build macro can be used to compile Control sources for a project. It can be used in an Ant build file as:

```
<import file="${beehive.home}/ant/beehive-tools.xml"/>

<build-controls srcdir="src/"
                destdir="classes/"
                tempdir="tmp/"
                classpathref="project.classpath"/>
```

This will build the Control sources from `src/` into `classes/` and perform code generation of Control beans into `tmp/` using the classpath `project.classpath`. The resulting .class files can then be used in your Java projects just as any other .class files.

2. JAR Projects

It is often useful to build Beehive Controls into a standalone JAR that can be copied into or shared among many Java projects. This can be done by first building the controls and then JAR-ing them into a standard `.jar` file. The Controls runtime also provides an Ant task that is used to perform additional processing for JARs that contain Controls. The Control JAR task can be used like this example:

```
<taskdef name="control-jar"
classname="org.apache.beehive.controls.runtime.packaging.ControlJarTask"
classpath="${beehive.home}/lib/controls/beehive-controls.jar"
onerror="fail"/>

<control-jar destfile="foo-controls.jar"
basedir="classes/">
  <manifest>
    <attribute name="Extension-Name" value="Foo Control"/>
    <attribute name="Specification-Title" value="Foo
```

```
Control"/>
                    <attribute name="Specification-Vendor"
value="AnyCorp"/>
                    <attribute name="Specification-Version" value="1.0.1"/>
                     <attribute name="Implementation-Title" value="Foo
Control"/>
                    <attribute name="Implementation-Vendor"
value="AnyCorp"/>
                    <attribute name="Implementation-Version"
value="1.0.1"/>
                  </manifest>
                </control-jar>
```

The `<coontrol-jar>` task also merges any `.manifest` files that were generated while processing control annotations. Use of this Ant task is not required, but it should be used any time Control manifest attributes need to be merted into a `META-INF/MANIFEST.MF` file.

3. Integrating into Existing Projects

Controls can also be used in existing Java projects by adding both the `<build-controls>` call to build Control sources. Optionally, the `<control-jar>` can be used to build the result into a JAR file; while this is best practice, it is not required. If Beehive Controls are added to existing Java projects, take care to resolve source file dependencies correctly. Because Controls perform code generation of Control Bean classes, clients of Control Bean classes need to be compiled **after** Controls have been built.

4. Source Control

In order to correctly add a Controls project to source control, several resources need to be checked in. Both required and optional resources are listed in the table below:

Name	JAR file	Version	Required
Beehive Controls	beehive-controls.jar	*distribution*	Yes
Beehive EJB Control	beehive-ejb-control.jar	*distribution*	Yes; if using EJB control functionality
Beehive JDBC Control	beehive-jdbc-control.jar	*distribution*	Yes; if using JDBC control functionality
Beehive JMS Control	beehive-jms-control.jar	*distribution*	Yes; if using JMS control functionality
Commons Codec	commons-codec-1.3.jar	*distribution*	Yes
Jakarta Velocity-dep	velocity-dep-1.4.jar	*distribution*	Yes; required at build

			time

The Velocity JARs are used by Controls for code-generation and do not need to be committed to SCM if they are referenced from a Beehive distribution. They are not required at runtime. The system control JARs are needed only if they are used in an application.