CommonsOsgi

Commons OSGi

The purpose of this page is to record progress in Commons of *OSGi-enabled* releases and any notes on specific issues with configuration of the OSGi manifest entries.

The Apache Felix project (an OSGi implementation) have requested that Commons components include OSGi meta data in their jars so that they are *ready-to-use* in an OSGi environment. This involves OSGi entries in the jar's manifest file.

The Felix project has developed the maven-bundle-plugin which makes this easier and this is now configured in the `commons-parent` pom.xml (since version 9) with *default* instructions. For components which need to override the *default* instructions, this can be done using *properties* in the component pom (see below for more details).

SpringSource Bundle Repository

The SpringSource Enterprise Bundle Repository contains a collection of open source libraries and each jar file in the repository is a valid OSGi bundle. This includes a number of re-packaged Commons components, so if you require a component that has not yet been released as an OSGi bundle, you may find it in the SpringSource repository.

N.B. As of writing (May 2008) the status of SpringSource repository is BETA.

Proper

Component	Last Release	First OSGi release	Notes / Comments
attributes	2.2		No m2 build
beanutils	1.7.0	1.8.0	
betwixt	0.8		
chain	1.1	1.2	javax.portlet
cli	1.1	1.2	
codec	1.3	1.4	
collections	3.2	3.2.1	Felix Bundle doesn't import org.apache.commons.collections.* - asked on dev@felix
compress		1.0	
configuration	1.5	1.6	Felix Bundle uses alot of dynamic import
daemon	1.0.1		
dbcp	1.2.2	1.3	
dbutils	1.1	1.2	
digester	1.8	1.8.1	Felix Bundle does dynamic import of org.apache.commons.logging.impl
discovery	0.4		
el	1.0		
email	1.1	1.2	
exec		1.0	
fileupload	1.2	1.2.1	javax.portlet
io	1.3.2	1.4	
jci	1.0		
jelly	1.0		No m2 build
jexl	1.1	2.0	
jxpath	1.2	1.3	See Felix Bundle
lang	2.3	2.4	
launcher	1.1		
logging	1.1.1		OSGi is N/A; See Below
math	1.1	1.2	See MATH-180
modeler	2.0.1		
net	1.4.1	2.0	
pool	1.3	1.4	
primitives	1.0		
proxy	n/a	1.0	
scxml	0.7	0.8	
transaction	1.2		

validator	1.3.1		Felix Bundle has org.apache.oro.*;resolution:=optional
vfs	1.0		

Commons Logging

The things that commons-logging does with classloaders in order to try and work in various servlet engine configurations are not compatible with the classloaders that OSGi environments set up. Therefore adding OSGi attributes to commons-logging is not useful, as commons-logging is not usable in an OSGi environment. See this thread and http://wiki.ops4j.org/display/paxlogging/Pax+Logging

Having said that the Felix project has a bundle to re-package logging - see here

Configuring Commons Components for OSGi with Maven1

This can be done by specifying appropriate manifest entries for the jar plugin (Pool 1.4 is an example) - however it is much easier using maven2 and the `maven-bundle-plugin`.

Configuring Commons Components for OSGi with Maven2

The latest version of the commons-parent pom configures the maven-bundle-plugin's *instructions* using *properties* in the following way:

```
<instructions>
    <!-- stops the "uses" clauses being added to "Export-Package" manifest entry -->
    <_nouses>true</_nouses>
    <Bundle-SymbolicName>${commons.osgi.symbolicName}</Bundle-SymbolicName>
    <Export-Package>${commons.osgi.export}</Export-Package>
    <Private-Package>${commons.osgi.private}</Private-Package>
    <Import-Package>${commons.osgi.import}</Import-Package>
    <DynamicImport-Package>${commons.osgi.dynamicImport}</DynamicImport-Package>
    <Bundle-DocURL>${project.url}</Bundle-DocURL>
</instructions>
```

This allows component poms to easily override the *default* values, by specifying alternative values for these properties in their pom. The default values for these properties in commons-parent are:

```
<properties>
    <commons.osgi.symbolicName>org.apache.commons.${commons.componentid}</commons.osgi.symbolicName>
    <commons.osgi.export>org.apache.commons.*;version=${pom.version}</commons.osgi.export>
    <commons.osgi.import>*</commons.osgi.import>
    <commons.osgi.dynamicImport></commons.osgi.dynamicImport>
    <commons.osgi.private></commons.osgi.private>
</properties>
```

For example, depending on the nature/structure of the project, some packages might contain private classes that are not meant to be used by client code. In these cases, <Export-Package> should list all public packages while <Private-Package> should contain the private packages. This can be achieved by overriding the `<commons.osgi.export>` and `<commons.osgi.private>` properties in the component pom.

Version 8 of commons-parent required specifying `<packaging>bundle</packaging>` in the component pom - with version 9 this is no longer required. In version 9, the `maven-bundle-plugin` is configured to generate a `MANIFEST.MF` file containing the OSGi *meta data* in the `target/osgi` folder. The `maven-jar-plugin` is configured to merge this generated `MANIFEST.MF` file with the manifest entries it is configured with. If a component does not want to include OSGi meta-data in their pom's manifest file they can configure an alternative or blank manifest file location using the `<commons.manifestfile>` property.