OSGI conversion tips

Collection of OSGi conversion tips

jars with xmlbeans compilation

To use xmlbeans generated code you need access to the SchemaTypeSystemImpl, which is not imported by the maven-bundle-plugin. So you need to add something like this to the pom:

```
            <plugin>
                <groupId>org.apache.felix</groupId>
                <artifactId>maven-bundle-plugin</artifactId>
                <configuration>
                    <instructions>
                        <!--packages containing "impl" or "internal" are excluded by default -->
                        <Import-Package>org.apache.xmlbeans.impl.schema;version="2.4",*</Import-Package>
                        <!--<_versionpolicy>[$(version;==;$(@)),$(version;+;$(@)))</_versionpolicy>-->
                    </instructions>
                </configuration>
            </plugin>
```

Also, any package we may have with "impl" or "internal" needs to be explicitly listed in the Export-Packages.

If the xmlbeans compilation is to be used by an xmlbeans compilation in another bundle (for example, the "environment" element defined in the geronimo-service-builder compilation), extra steps are needed to make these available. In the defining bundle, all of the generated classes AND all of the packages under the schemaorg_apache_xmlbeans directory need to be exported. Because the bnd tool omits impl directories by default, these must be explicitly listed. Here's a sample from the geronimo-service-builder:

```
                    <instructions>
                        <!--<_versionpolicy>[$(version;==;$(@)),$(version;+;$(@)))</_versionpolicy>-->
                        <Import-Package>org.apache.xmlbeans.impl.schema,*</Import-Package>
                        <Export-Package>
                            org.apache.geronimo.deployment.xbeans.impl,
                            org.apache.geronimo.deployment.javabean.xbeans.impl,
                            org.apache.geronimo.deployment.xbeans*,
                            org.apache.geronimo.deployment.javabean.xbeans*,
                            schemaorg_apache_xmlbeans*,
                            org.apache.geronimo.deployment.dconfigbean,
                            org.apache.geronimo.deployment.service*
                        </Export-Package>
                    </instructions>
```

On the importing side, all of these classes need to be imported as well. The schemaorg_apache_xmlbeans directory is a bit problematic because the bnd tool cannot handle wild-cards on the import unless the packages are explicitly referenced in the code. The schemaorg packages are not referenced in the code and generate lot of cryptically named directories that are difficult to transfer over to the importing side. For the schemaorg directories, DynamicImport-Package can be used. This example is from the geronimo-connector-builder-1_6 plugin:

```
                    <instructions>
                        <!--packages containing "impl" or "internal" are excluded by default -->
                        <Import-Package>org.apache.xmlbeans.impl.schema;version="2.4",org.apache.geronimo.
deployment.xbeans.impl,org.apache.geronimo.deployment.javabean.xbeans.impl,*</Import-Package>
                        <DynamicImport-Package>schemaorg_apache_xmlbeans.*</DynamicImport-Package>
                        <!--<_versionpolicy>[$(version;==;$(@)),$(version;+;$(@)))</_versionpolicy>-->
                    </instructions>
```

If the xmlbeans compilation is to be used by an xmlbeans compilation in another bundle (for example, the "environment" element defined in the geronimo-service-builder compilation), extra steps are needed to make these available. In the defining bundle, all of the generated classes AND all of the packages under the schemaorg_apache_xmlbeans directory need to be exported. Because the bnd tool omits impl directories by default, these must be explicitly listed. Here's a sample from the geronimo-service-builder:

```
                    <instructions>
                        <!--<_versionpolicy>[$(version;==;$(@)),$(version;+;$(@)))</_versionpolicy>-->
                        <Import-Package>org.apache.xmlbeans.impl.schema,*</Import-Package>
                        <Export-Package>
                            org.apache.geronimo.deployment.xbeans.impl,
                            org.apache.geronimo.deployment.javabean.xbeans.impl,
                            org.apache.geronimo.deployment.xbeans*,
                            org.apache.geronimo.deployment.javabean.xbeans*,
                            schemaorg_apache_xmlbeans*,
                            org.apache.geronimo.deployment.dconfigbean,
                            org.apache.geronimo.deployment.service*
                        </Export-Package>
                    </instructions>
```

On the importing side, all of these classes need to be imported as well. The schemaorg_apache_xmlbeans directory is a bit problematic because the bnd tool cannot handle wild-cards on the import unless the packages are explicitly referenced in the code. The schemaorg packages are not referenced in the code and generate lot of cryptically named directories that are difficult to transfer over to the importing side. For the schemaorg directories, DynamicImport-Package can be used. This example is from the geronimo-connector-builder-1_6 plugin:

```
                    <instructions>
                        <!--packages containing "impl" or "internal" are excluded by default -->
                        <Import-Package>org.apache.xmlbeans.impl.schema;version="2.4",org.apache.geronimo.
deployment.xbeans.impl,org.apache.geronimo.deployment.javabean.xbeans.impl,*</Import-Package>
                        <DynamicImport-Package>schemaorg_apache_xmlbeans.*</DynamicImport-Package>
                        <!--<_versionpolicy>[$(version;==;$(@)),$(version;+;$(@)))</_versionpolicy>-->
                    </instructions>
```

Non-bundle interference

Many problems with building plugins are caused by non-bundle dependencies getting installed in felix rather than bundleized equivalents. Unfortuneately it looks like felix only says "non-framework bundles cannot be started" without telling us the location of the non-bundle. Running

```
mvn dependency:tree
```

helps to find the bad dependencies. On a related note, generally you have to exclude original jars from the dependencyManagement dependency entry of a bundleized repackaging. This seems like a serious defect in maven-bundle-plugin.

Assemble a server to make sure each set of plugins starts.

I think it's a good idea to assemble a server for each set of plugins to make sure they at least start. Here's how:

1. run

   ```
   mvn archetype:create \
      -DarchetypeGroupId=org.apache.geronimo.buildsupport \
      -DarchetypeArtifactId=geronimo-assembly-archetype \
      -DarchetypeVersion=3.0-SNAPSHOT \
      -DgroupId=org.apache.geronimo.plugins \
      -DartifactId=geronimo-<foo>-server
   ```

   where <foo> is your set of plugins
2. add it to svn
3. add the plugins you want to test as depedendencies
4. add this to start the server when run with -Pit
   in plugin management:

```
                <plugin>
                    <groupId>org.apache.geronimo.buildsupport</groupId>
                    <artifactId>geronimo-maven-plugin</artifactId>
                    <version>${version}</version>

                    <configuration>
                       <assemblyArchive>${project.build.directory}/${pom.artifactId}-${pom.version}-bin.
zip</assemblyArchive>
                        <optionSets>
                            <optionSet>
                                <id>morememory</id>
                                <options>
                                    <option>-Xmx512m</option>
                                    <option>-XX:MaxPermSize=128m</option>
                                </options>
                            </optionSet>

                            <optionSet>
                                <id>debug</id>
                                <options>
                                    <option>-Xdebug</option>
                                    <option>-Xrunjdwp:transport=dt_socket,address=8000,server=y,
suspend=n</option>
                                </options>
                            </optionSet>
                        </optionSets>
                    </configuration>
                    <executions>
                         <execution>
                             <id>start</id>
                             <phase>pre-integration-test</phase>
                             <goals>
                                 <goal>start-server</goal>
                             </goals>
                             <configuration>
                                 <assemblyId>${it-server}</assemblyId>
                                 <logOutput>true</logOutput>
                                 <background>true</background>
                                 <verifyTimeout>300</verifyTimeout>
                                 <refresh>true</refresh>
                                 <optionSets>
                                     <optionSet>
                                         <id>default</id>
                                         <options>
                                            <option>-XX:MaxPermSize=128m</option>
                                         </options>
                                     </optionSet>

                                     <optionSet>
                                          <id>morememory</id>
                                          <options>
                                              <option>-Xmx512m</option>
                                              <option>-XX:MaxPermSize=128m</option>
                                          </options>
                                      </optionSet>

                                      <optionSet>
                                          <id>debug</id>
                                          <options>
                                              <option>-Xdebug</option>
                                              <option>-Xrunjdwp:transport=dt_socket,address=8000,
server=y,suspend=n</option>
                                          </options>
                                      </optionSet>
                                 </optionSets>
                             </configuration>
                         </execution>
                         <execution>
                             <id>stop</id>
                             <phase>install</phase>
```

```
                            <goals>
                                <goal>stop-server</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
```

and a profile

```
    <profiles>
        <profile>
            <id>it</id>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.apache.geronimo.buildsupport</groupId>
                        <artifactId>geronimo-maven-plugin</artifactId>
                    </plugin>
                </plugins>
            </build>
        </profile>
    </profiles>
```

Note that at the moment the -Pit doesn't actually work, you have to try to start the server by hand

Classloading problems

Frequently, you'll see build problems with attempting to load a class when starting a configuration. Most of the time, this is caused by a bundle resolution problem that occurs earlier, but the information has been swallowed. If you use the -X option on the build, the resolution error will be given earlier in the build and you can generally figure out what's missing from that.

Occasionally, the resolution error will be a very generic "Constraint violation" without much information on what actually failed. Turning on debug logging in Felix will give some information, but I've found that building Felix with the following patch applied helps diagnose the problem more quickly:

```
Index: src/main/java/org/apache/felix/framework/searchpolicy/Resolver.java
===================================================================
--- src/main/java/org/apache/felix/framework/searchpolicy/Resolver.java        (revision 831350)
+++ src/main/java/org/apache/felix/framework/searchpolicy/Resolver.java        (working copy)
@@ -332,8 +332,12 @@
                     }
                     else
                     {
+                        System.out.println(
+                            "Constraint violation for " + importer
+                            + " detected; module can see "
+                            + rp + " and " + rpUses);
                         m_logger.log(
-                            Logger.LOG_DEBUG,
+                            Logger.LOG_ERROR,
                             "Constraint violation for " + importer
                             + " detected; module can see "
                             + rp + " and " + rpUses);
@@ -705,7 +709,7 @@
         catch (ResolveException ex)
         {
             m_logger.log(
-                Logger.LOG_DEBUG,
+                Logger.LOG_ERROR,
                 "Constraint violation for " + targetModule + " detected.",
                 ex);
             return false;
@@ -789,7 +793,7 @@
         catch (ResolveException ex)
         {
             m_logger.log(
-                Logger.LOG_DEBUG,
+                Logger.LOG_ERROR,
```

```
                "Constraint violation for " + targetModule + " detected.",
                ex);
            return false;
@@ -824,7 +828,7 @@
         catch (ResolveException ex)
         {
             m_logger.log(
-                Logger.LOG_DEBUG,
+                Logger.LOG_ERROR,
                 "Constraint violation for " + targetModule + " detected.",
                 ex);
             return false;
@@ -875,7 +879,7 @@
                     else
                     {
                         m_logger.log(
-                            Logger.LOG_DEBUG,
+                            Logger.LOG_ERROR,
                             "Constraint violation for " + targetModule
                             + " detected; module can see "
                             + rp + " and " + rpUses);
Index: src/main/java/org/apache/felix/framework/ModuleImpl.java
===================================================================
--- src/main/java/org/apache/felix/framework/ModuleImpl.java        (revision 831350)
+++ src/main/java/org/apache/felix/framework/ModuleImpl.java        (working copy)
@@ -1295,7 +1295,7 @@

     public String toString()
     {
-        return m_id;
+        return getSymbolicName() + " (" + m_id + ")";
     }

     private synchronized ModuleClassLoader getClassLoader()
@@ -1590,12 +1590,12 @@

             try
             {
-                dexFileClassLoadDex = dexFileClass.getMethod("loadDex",
+                dexFileClassLoadDex = dexFileClass.getMethod("loadDex",
                     new Class[]{String.class, String.class, Integer.TYPE});
             }
             catch (Exception ex)
             {
-                // Nothing we need to do
+                // Nothing we need to do
             }
             dexFileClassConstructor = dexFileClass.getConstructor(
                 new Class[] { java.io.File.class });
@@ -1717,7 +1717,7 @@

                         if (clazz == null)
                         {
-                            int activationPolicy =
+                            int activationPolicy =
                                 ((BundleImpl) getBundle()).isDeclaredActivationPolicyUsed()
                                 ? ((BundleImpl) getBundle()).getCurrentModule().getDeclaredActivationPolicy()
                                 : IModule.EAGER_ACTIVATION;
@@ -1873,8 +1873,8 @@
                 {
                     if (m_dexFileClassLoadDex != null)
                     {
-                        dexFile = m_dexFileClassLoadDex.invoke(null,
-                            new Object[]{content.getFile().getAbsolutePath(),
+                        dexFile = m_dexFileClassLoadDex.invoke(null,
+                            new Object[]{content.getFile().getAbsolutePath(),
                                 content.getFile().getAbsolutePath() + ".dex", new Integer(0)});
                     }
                     else
```

Logging conversion

When converting plugins make sure to avoid the following dependencies: `log4j`, `jcl-over-slf4j`, `jul-to-slf4j`, `slf4j-api`, or `slf4j-log4j12`. Instead use `pax-logging-api`. This bundle exports all of these logging API.

```
<dependency>
    <groupId>org.ops4j.pax.logging</groupId>
    <artifactId>pax-logging-api</artifactId>
</dependency>
```

The `pax-logging-api` together with `pax-logging-service` provides a logging service in the OSGi environment that works with all commonly used logging API.

General OSGi tips

- The Bundle-ClassPath can refer to a directory within the bundle. Make sure there is an entry just for that directory in the JAR file of the bundle. Otherwise, classes might not be loaded from that directory.
- Always specify some wild mark in the filePattern (the second argument) in the bundle.findEntries() call to ensure the same set of entries is returned on different OSGi frameworks.