An overview and introduction to OSGi BDT

Renato Brunella, brunella ltd

MANIFEST FIRST OSGI DEVELOPMENT

 renato_brunella@acm.org http://sourceforge.net/projects/osgi-bdt

A few words about myself...

- **Renato Brunella**
  - Senior IT consultant and CTO of brunella ltd, UK
  - Project Administrator and Development Lead of the open source projects **OSGi BDT** and **Query Object Factory** hosted on SourceForge
  - 10 years of Java development experience
- **Areas of interest**
  - Software design and architecture
  - OSGi Technology and OSGi Tooling
  - Agile software development

Overview

- Wish list for OSGi build tools
- Motivation and approach of OSGi BDT
- OSGi BDT Repository
- Building bundles using OSGi BDT Ant tasks
- Testing bundles using OSGi BDT test runners
- OSGi BDT Workflow

Wish list for OSGi build and test tool

- Manifest defines all dependencies and should be used by tooling i.e. no other dependency definitions should be needed
- Dependencies should be validated by the compiler i.e. fail during build and not at runtime
- Build dependency between bundles and order of build should be resolved by the tool
- Ant, JUnit, FitNesse and command line support
- Supports unit, black box, integration and acceptance testing
- Support for Equinox, Felix, Knopflerfish
- IDE integration (Eclipse)
- Automated testing against different platforms, versions of bundles
- Repository based
- Hierarchy of repositories including automated propagation of bundles to higher repositories (Local to team, team to validation, validation to production, etc.)
- OBR support for remote repositories
- RESTful interface to repositories for web access and integration

Motivation for developing OSGi BDT

- Lack of OSGi tooling to support build process
- Available tooling has large footprint, is invasive, has many dependencies and is focused on a certain technology stacks (Eclipse PDE, Spring/Maven)
- No manifest first/fail early tool
- No easy to use and free tools with Ant, JUnit and FitNesse support

OSGi BDT Approach

- Based on one or more local repositories
- Tooling for the creation of repositories, deployment and un-deployment of bundles
- Automatically analyses build dependencies of bundles and creation of build classpath
- Support for automated testing
- Small footprint (~350kb for the tool)
- Ant, JUnit, FitNesse, command line support
- Eclipse plugin

Legend

J2SE-1.5

BDT Repository
For a execution environment (EE)

A
a.a [1.0.0]
b.a [1.5.0,2.0.0)

Bundle A
Black: Exported packages with version
White: Imported packages with version range

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: X
Bundle-Version: 1.0.0

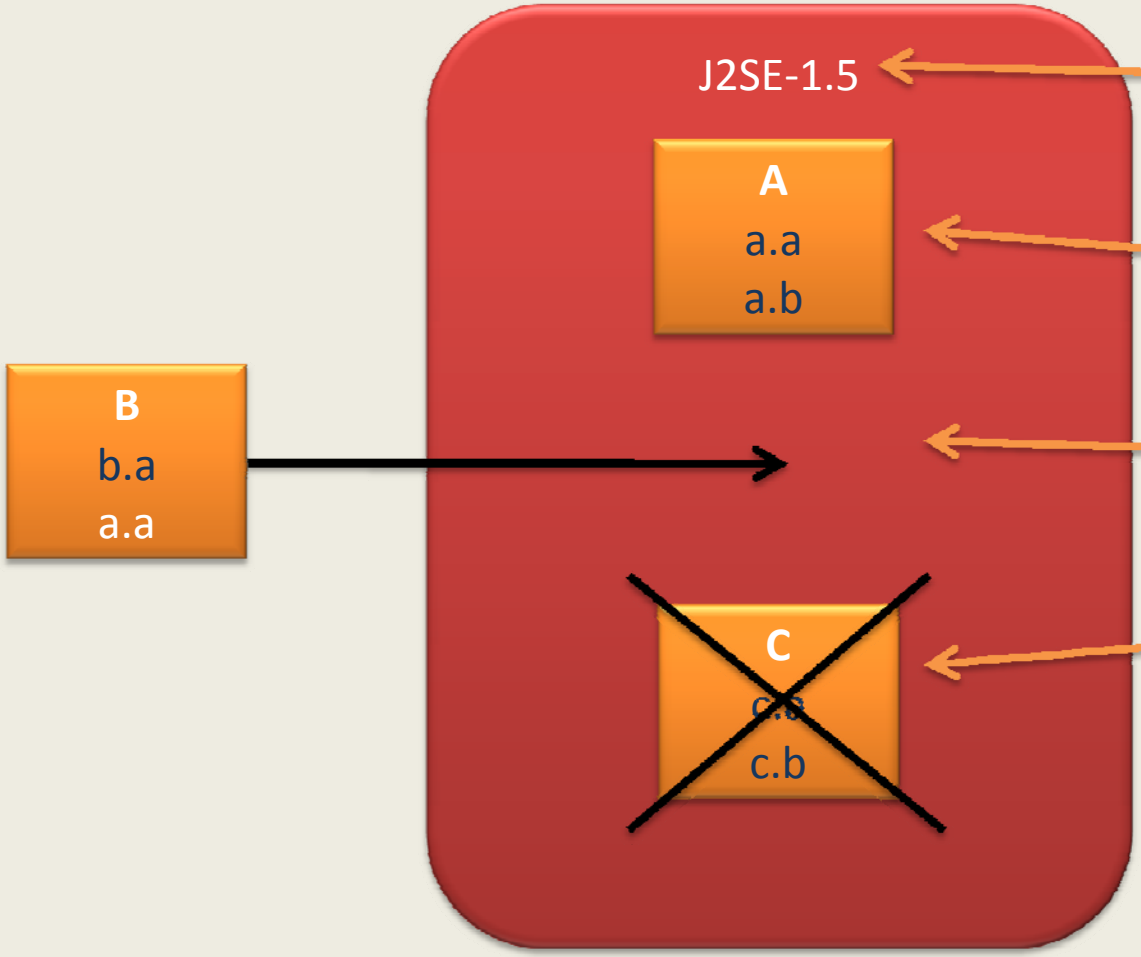
Manifest
Define bundle name and version and dependencies to other bundles

Equinox 3.4.0

OSGi Environment
Equinox, Felix or Knopflerfish

OSGi BDT Repository

Create, deploy, un-deploy, resolve



Create new local OSGi BDT repository for J2SE-1.5 EE

Bundle A is already deployed to the repository

Deploy Bundle B into the repository

Un-deploy Bundle C from the repository

Repository can resolve bundles and packages by name and version range

OSGi BDT Eclipse Plugin

Control BDT Repositories within Eclipse

- TODO: Add screenshots

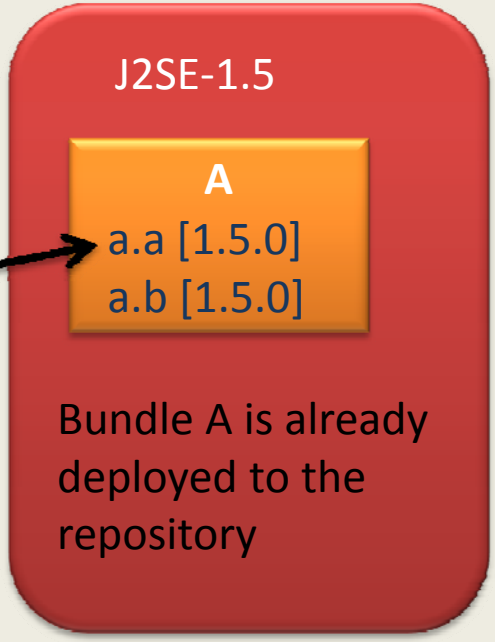
OSGi BDT Ant Tasks

Building a bundle

Directory structure:

./bundle-b/
  ./bin
  ./src/b/a/ServiceB.java
  ./META-INF/MANIFEST.MF
  ./build.xml

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: B
Bundle-Version: 1.0.0
Export-Package: b.a;version="1.0.0"
Import-Package: a.a;version="1.5.0"



Ant build file:

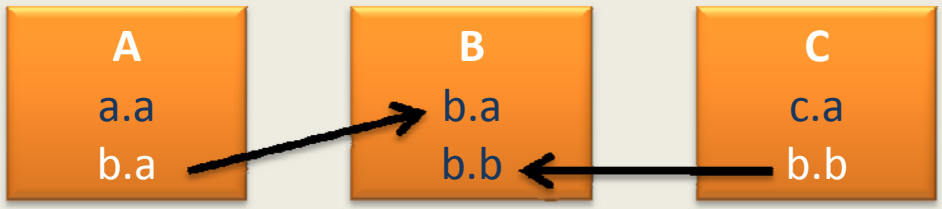
```
<osgi-path id="buildclasspath"
      repository="${repository.dir}"
      manifest="${base.dir}/META-INF/MANIFEST.MF">
</osgi-path>

 <javac destdir="${bin.dir}">
       <src path="${src.dir}" />
       <classpath refid="buildclasspath" />
 </javac>
```

BDT resolves the classpath to the specified package in the repository or fails the build if no matching package can be found

javac **-cp c:/repository/packages/A/1.0.0/a.a/1.5.0/** -d ./bundle-b/bin/ -sourcepath ./bundle-b/src/

OSGi BDT Ant Tasks

Building a series of bundles



Build order: First B then A and C

The manifests of the bundles that need to be built define the dependencies between the bundles and therefore the build order.

```
<osgi-build manifestfile="./META-INF/MANIFEST.MF"
      buildfile="build.xml"
      buildtarget="build-and-deploy"
      repository="${repository.dir}“
      fullrebuild="true">
   <dirset dir="${basedir}/..">
      <include name=“uk.co.brunella.bundles.*"/>
      <exclude name=" uk.co.brunella.bundles.test.*" />
   </dirset>
</osgi-build>
```

The osgi-build Ant task is used to build multiple bundles. The bundles’ directory structure is all the same with a build.xml file in the root and a manifest file in ./META-INF. These build files have a target build-and-deploy that will be called to build the bundle. A dirset element controls which bundles to include. fullrebuild controls if all bundles will be built or only the ones that have changes compared to the bundle in the repository.

OSGi BDT Ant Tasks Summary

Ant task name	Description
osgi-create	Creates a new OSGi BDT Repository
osgi-deploy	Deploys a bundle to a repository
osgi-undeploy	Un-deploys/removes a bundle from a repository
osgi-list	Lists all bundles in a repository
osgi-manifest	Reads the bundle symbolic name and version into Ant properties
osgi-path	Creates the build classpath using the manifest and resolves dependencies to packages of bundles in the repository
osgi-build	Resolves the build dependencies between bundles that need to be built and determines the build order
osgi-test	Runs an acceptance test

*The OSGi BDT Eclipse plugin contributes these task to the Ant runtime within Eclipse

OSGi BDT Command Line Support

java -jar uk.co.brunella.osgi.bdt-2.1.0.jar

OSGi BDT:

-**help**
-**create** *repository profilename*
-**listprofiles**
-**deploy** *bundle repository*
-**undeploy** *bundlename bundleversion repository*
-**list** *repository*
-**resolve** *packagename versionrange repository*

Automated Testing of Bundles

- Automated testing of bundles is done at different levels
- Levels of testing:
  - Unit testing
  - Black-box or isolation testing
  - Integration testing
  - Acceptance testing

Unit Testing

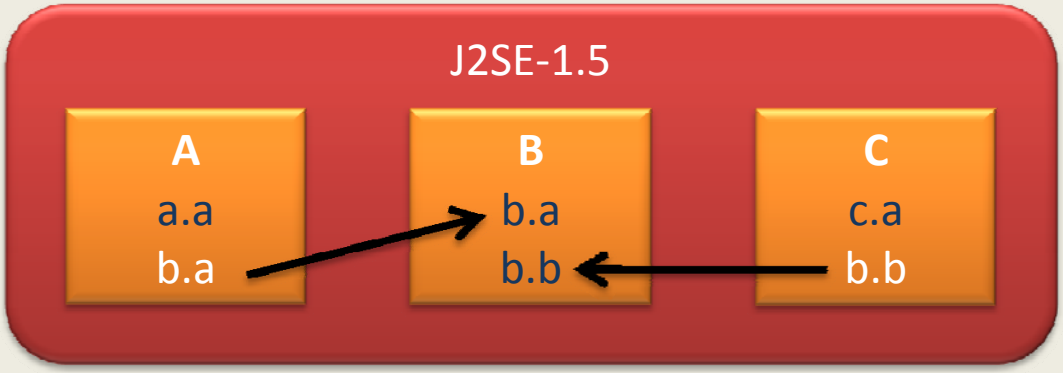
- Unit testing is using unit tests during the build of the bundle **after** compiling the source code and **before** the actual bundle is created
- Does **not** test the bundle in an OSGi environment
- We use JUnit and mock objects when the unit touches the OSGi framework i.e. MockBundle, MockBundleContext etc.
- Tests must run very fast

Black-box or Isolation Testing

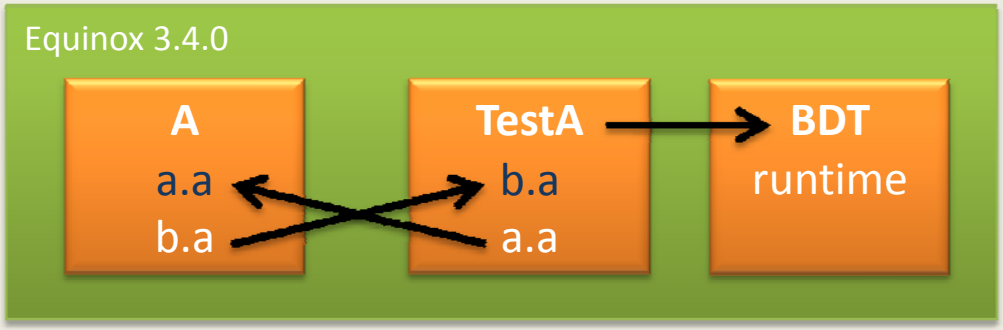
- Test the bundle in an OSGi environment
- Test the bundle using test bundle(s)
- Test the bundle in isolation with minimal dependencies on “real” bundles i.e. all dependencies should be provided by the test bundle
- Use the JUnit framework to write tests
- OSGi BDT provides support for this!

Black-box or Isolation Testing

OSGi BDT Style



Three bundles in the repository after build, unit testing and deployment.



- The test bundle **TestA** and the bundle under test **A** are installed and started in an OSGi environment
- Bundle **TestA** imports package **a.a** from **A** and exports package **b.a** i.e. it injects a test package the dependency that bundle **A** has
- Bundle **A** can be tested in isolation

Black-box or Isolation Testing

OSGi BDT Style (cont.)

JUnit test class:

```
@RunWith(OSGiBDTJUnitRunner.class)
@OSGiBDTTest(
    baseDir = ".",
    repositories = "${OSGI_REPOSITORY}",
    manifest = "META-INF/MANIFEST.MF",
    buildIncludes = { @Include(source = "bin", dest = "") },
    framework = Framework.EQUINOX,
    frameworkStartPolicy = StartPolicy.ONCE_PER_TEST_CLASS,
    requiredBundles = { "A" }
)
public class BundleTest {
  @OSGiBundleContext
  private BundleContext bundleContext;

  @OSGiService(serviceName = "a.a.Service")
  private ServiceA service;

  @Test
  public void testServiceA() {
    ...
  }
}
```

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: TestA
Bundle-Version: 1.0.0
Export-Package: b.a;version="1.0.0"
Import-Package: a.a;version="1.5.0",
org.osgi.framework;version="1.4.0",
org.junit,org.junit.runner,
uk.co.brunella.osgi.bdt.junit.annotation,
uk.co.brunella.osgi.bdt.junit.runner

The OSGiBDTJUnitRunner automatically creates the test bundle **TestA**, starts the Equinox framework, installs its runtime, bundle **A** and the test bundle **TestA**, starts all bundles and runs the tests. After that the framework is stopped.
The test runner runtime has only a dependency to **org.osgi.framework.**

Black-box or Isolation Testing

OSGi BDT Style (cont.)

```
@RunWith(OSGiBDTJUnitRunner.class)

@OSGiBDTTest(

  baseDir = ".",

  repositories = "${OSGI_REPOSITORY}",

  manifest = "META-INF/MANIFEST.MF",

  buildIncludes =
      { @Include(source = "bin", dest = "") },

  framework = Framework.EQUINOX,

  frameworkStartPolicy =
      StartPolicy.ONCE_PER_TEST_CLASS,

  requiredBundles = { "A" }

)
```

- Tell JUnit to use the OSGi BDT runner
- This is a OSGi BDT test case
- The base directory of the test bundle project
- Where to find the repository
- Where to find the manifest
- Include anything in the bin directory in the test bundle jar
- Run against the Equinox framework
- Run all test methods in the same OSGi environment
- All the required bundles

Black-box or Isolation Testing

OSGi BDT Style (cont.)

```
@RunWith(OSGiBDTJUnitRunner.class)
@OSGiBDTTest(
   ...
)
public class BundleTest {

 @OSGiBundleContext
 private BundleContext bundleContext;

 @OSGiService(serviceName = "a.a.Service")
 private ServiceA service;

 @Test
 public void testServiceA() {
   ...
 }
}
```

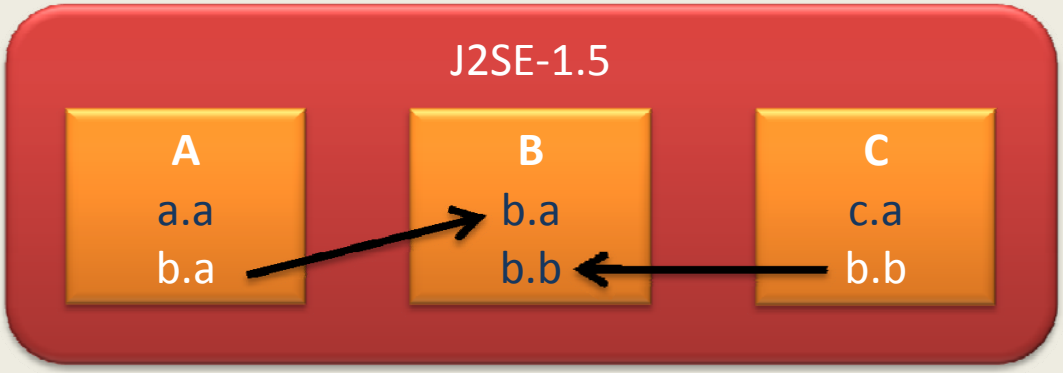
- Tell JUnit to use the OSGi BDT runner
- This is an OSGi BDT test case
- Inject the bundle context of the test bundle into a field in the test class
- Inject an OSGi service into a field in the test class
- Normal JUnit 4 test annotation

Integration Testing

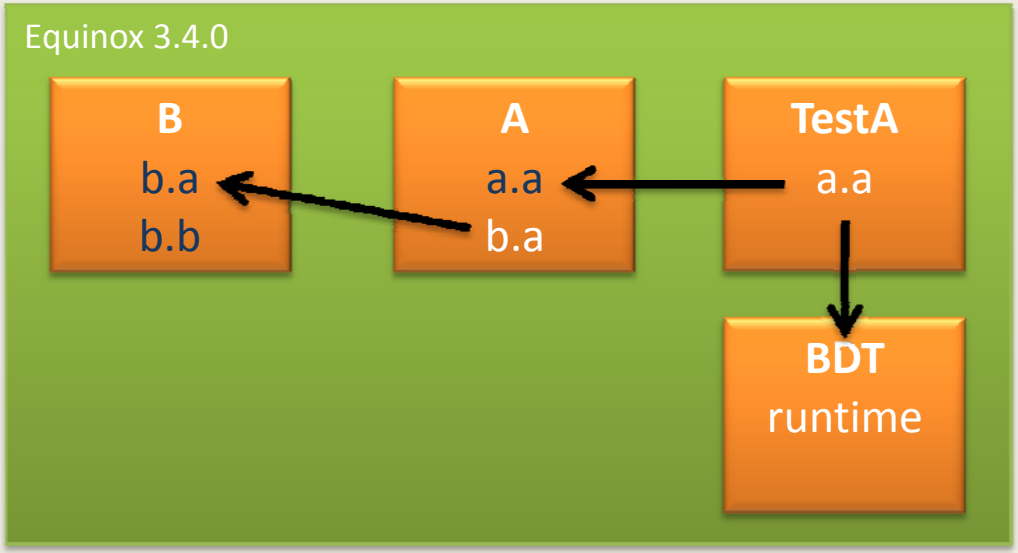
- Test the bundle in an OSGi environment
- Test the bundle using test bundle(s)
- Test the bundle in interaction with the real bundles it depends on
- Use the JUnit framework to write tests
- Again OSGi BDT provides support for this!

Integration Testing

OSGi BDT Style



Three bundles in the repository after build, unit testing and deployment. Bundle **A** has a dependency to bundle **B**



- The test bundle **TestA** and the bundle under test **A** as well as its dependency **B** are installed and started in an OSGi environment
- Bundle **TestA** imports package **a.a** from **A** and bundle B exports package **b.a**
- Bundle **A** can be tested in integration with its real dependency

Acceptance Testing

- FitNesse is a wiki based acceptance framework based on FIT ("Framework for integration testing")
- Acceptance tests are written using fixtures
- OSGi BDT provides a fixture that enables FitNesse to run tests in an OSGi environment and report the test results back to the wiki

Acceptance Testing (cont.)

- TODO: include FitNesse screen shots

OSGi BDT Workflow

- TODO: include workflow:
  - Unit Test & build
  - Deploy
  - Black-box testing
  - Integration testing
  - Acceptance testing
  - Generic Ant build files

Thank you for your attention

- Questions are always welcome
- Would you like to contribute to OSGi BDT?

Renato Brunella

renato_brunella@acm.org