Rational® Application Development and M2Eclipse

Maven in the development workbench for RAD & RSA

Chuck Bridgham (cbridgha@us.ibm.com)
Roberto Sanchez Herrera (rsanchez@mx1.ibm.com)
August 2011 – Updated June, 2012

Contents

M2Eclipse and Rational Java Application Development 3

Setup and Prerequisites 3

M2Eclipse features 6

How does m2eclipse work? What should I expect? 10

Maven Archiver and MANIFEST generation in Rational Application Developer 11

Recommended Preferences 12

M2Eclipse preferences 12

Java EE preferences from Rational Application Developer 13

Sample Scenarios 15

Creating Maven web project 15

“Mavenize” existing Rational Application Developer projects 18

WebSphere Application Server deployment 22

Full Java EE Maven sample 22

Known issues 27

What’s next? 27

References 27

M2Eclipse and Rational Java Application Development

This paper explores real-world scenarios using either Rational® Application Developer or Rational® Software Architect and m2eclipse together. Today, there are more options and better integration available due to improvements to the m2eclipse features in Web Tools Platform based on Eclipse technology. I provide recommendation on several tips that help prevent problems that many users commonly encounter. This paper is a follow-up to my overview published in 2010 <link> detailing the world of Maven in a Rational Application Developer environment, and a more detailed explanation of the Maven build system. The previous article mentioned many instances where Maven and Rational Application Developer conflicts, or represent similar properties by separate mechanisms. This is still true, but by following the best practices outlined in this paper, and help from m2eclipse *translation* of the internal metadata, the differences are hidden to the user.

The m2eclipse open-source project originally was hosted by the company that founded Maven, Sonatype Inc. In 2010, Sonatype announced that this project would be donated and incubated at Eclipse with a new name m2eclipse. The project successfully graduated with their Indigo release (2011), and is on track to deliver again with the Juno (2012) release in June. This project has many features that allow Maven developers to take advantage of Eclipse and Rational Application Developer features, while maintaining Maven project metadata. Maven projects can either be created or imported from existing environments. Maven dependency mechanisms and module structures are integrated into Rational Application Developer Java class path resolution framework. Maven repositories can be searched, and auto-update projects based on any updates to the repository. Form-based editing of the Maven `pom.xml` file is also supported. The Eclipse Java builder is responsible for compiling, building, assembling, and indexing all Java artifacts. The m2eclipse project configurator handles all non-java artifacts to be assembled by the Maven builder, and aggregates the results according to the Maven project model (`pom.xml`). The latest Indigo release of m2eclipse is backward compatible with the Helios release, making it compatible with Rational Application Developer V8.0.3 or later.

Maven, the m2eclipse feature, and its extended connectors (including m2e-wtp) are not supported by IBM®, but have a very active mailing list and problem reporting systems through Eclipse bugzilla and Sonotype that are monitored closely.

Setup and Prerequisites

This paper targets several releases of Rational Application Developer including V8.0.4, V8.0.4.1, V8.5.0, Websphere Development Toolkit V8.5, and Rational Software Architect V8.5. All recommendations and assumptions are based on these product-levels.

We are going to use the new Eclipse m2eclipse release 1.0.100 as part of the Indigo release. This supports the Helios release that Rational Application Developer requires. In addition, it supports Indigo for WDT installations. We are also going to add the latest m2e-WTP *add-on* support that is compatible with m2e and Helios. A separate Maven installation in not required, because m2e configures an embedded version of Maven.

1. In a new or existing Rational Application Developer or WDT workspace, open the menu **Help** → **Install New Software**

2. In the **Available Software** page of the **Install** wizard, click the **Add** button next to the **Work with** field and add this new location:

http://download.jboss.org/jbosstools/updates/m2eclipse-wtp/

3. Click on **Available Software Sites** link and verify the check boxes for the following sites are selected. Then click **OK**.

   For RAD/RSA:

   - m2eclipse-wtp updates
     http://download.jboss.org/jbosstools/updates/m2eclipse-wtp/

   - “**Eclipse Project Test Site**”
     http://download.eclipse.org/eclipse/updates/3.6

   - **The Eclipse Web Tools Platform (WTP) software repository**
     http://download.eclipse.org/webtools/repository/helios

   For WDT on Java EE EPP Package for Indigo:

   - **m2eclipse-wtp updates**
     http://download.jboss.org/jbosstools/updates/m2eclipse-wtp/

   - “**Eclipse Project Test Site**”
     http://download.eclipse.org/eclipse/updates/3.7

   - **The Eclipse Web Tools Platform (WTP) software repository**
     http://download.eclipse.org/webtools/repository/indigo

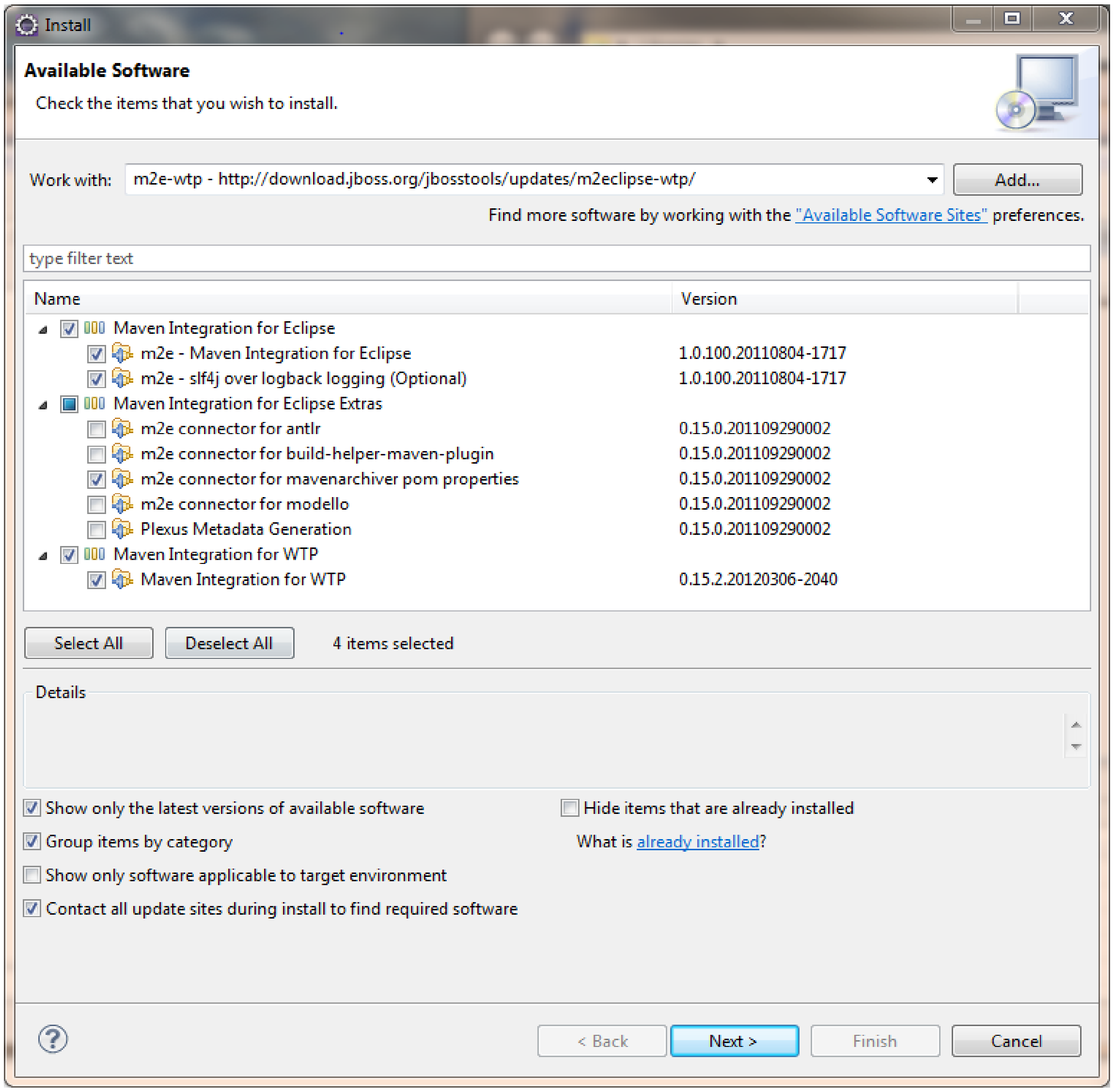
4. In the **Work with** field of the **Available Software** page, select: **m2eclipse-wtp updates**

5. Under the **Name** column, select the following check boxes:

   - **Maven Integration for Eclipse**

   - **Maven Integration for WTP**

   - Expand **Maven Integration for Eclipse Extras** folder and select only **m2e connector for mavenarchiver pom properties**



Work with: m2e-wtp - http://download.jboss.org/jbosstools/updates/m2eclipse-wtp/

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
[✓] Maven Integration for Eclipse	
[✓] m2e - Maven Integration for Eclipse	1.0.100.20110804-1717
[✓] m2e - slf4j over logback logging (Optional)	1.0.100.20110804-1717
[■] Maven Integration for Eclipse Extras	
[ ] m2e connector for antlr	0.15.0.201109290002
[ ] m2e connector for build-helper-maven-plugin	0.15.0.201109290002
[✓] m2e connector for mavenarchiver pom properties	0.15.0.201109290002
[ ] m2e connector for modello	0.15.0.201109290002
[ ] Plexus Metadata Generation	0.15.0.201109290002
[✓] Maven Integration for WTP	
[✓] Maven Integration for WTP	0.15.2.20120306-2040

4 items selected

[✓] Show only the latest versions of available software
[ ] Hide items that are already installed
[✓] Group items by category
What is already installed?
[ ] Show only software applicable to target environment
[✓] Contact all update sites during install to find required software

6. Click **Next** in the **Available Software** page.

7. Click **Next** in the **Install Details** page.

8. In the **Review Licenses** page, accept the Eclipse Public Licenses and click **Finish**.

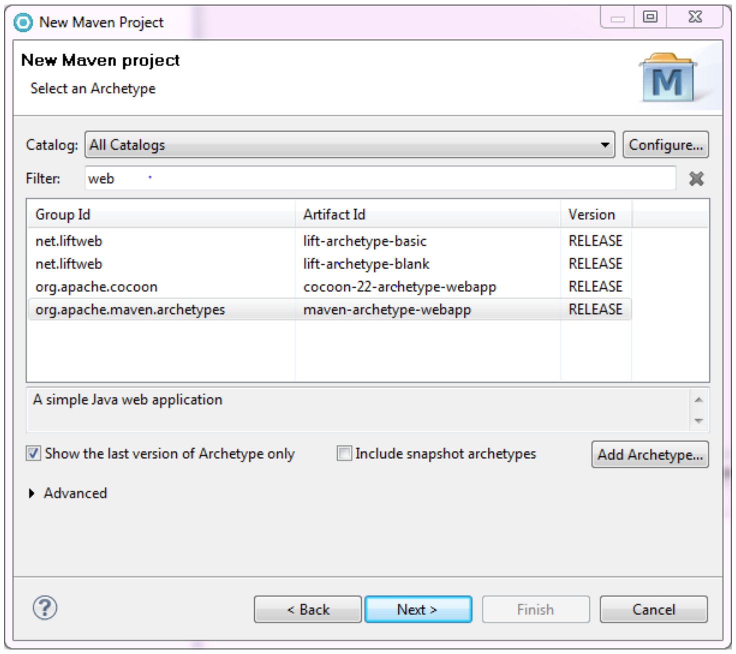
9. In the **Security Warning** window, click **OK** to accept unsigned content.

10. Allow the workspace to restart when finished.

M2Eclipse features

Now that we have the m2eclipse features installed, here are some of the key features and integration with Rational Application Developer workbench:

1. **Maven Project Wizards:** The new Maven Project wizard allows you to create custom Maven projects, or use predefined Maven archetypes that describe the purpose, layout, and required dependencies of a project. For example, a new Java web project can be found by typing `web` in the **type filter text** field. Several archetypes appear and by selecting the popular Maven-archetype-webapp creates a simple web project defaulting to the Servlet 2.3 specification.



Catalog: All Catalogs

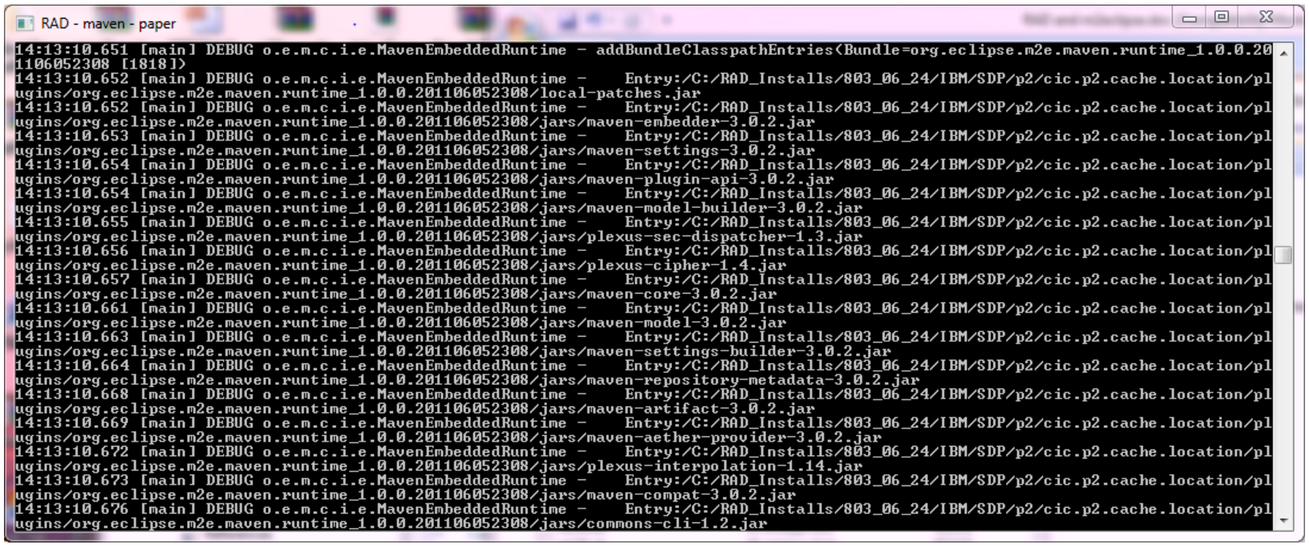
Filter: web

Group Id	Artifact Id	Version
net.liftweb	lift-archetype-basic	RELEASE
net.liftweb	lift-archetype-blank	RELEASE
org.apache.cocoon	cocoon-22-archetype-webapp	RELEASE
org.apache.maven.archetypes	maven-archetype-webapp	RELEASE

A simple Java web application

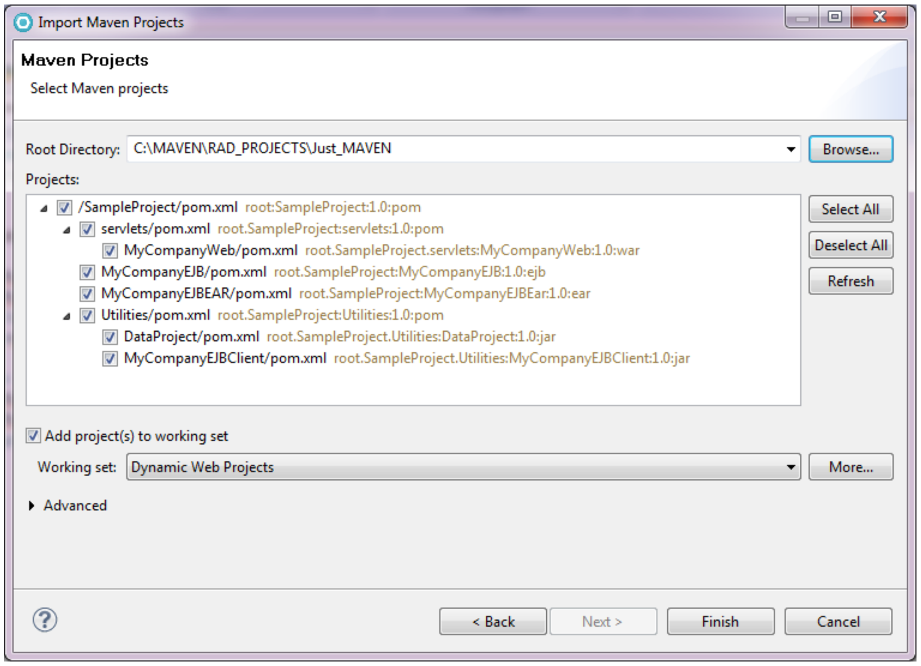
[✓] Show the last version of Archetype only
[ ] Include snapshot archetypes

2. **Maven Repository Management:** Automatic dependency downloads and updates happen behind the scenes. The console logs messages during an initial Maven project creation and build that indicate the dependent libraries required for project creation and build:



```
14:13:10.651 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime - addBundleClasspathEntries(Bundle=org.eclipse.m2e.maven.runtime_1.0.0.20
1106052308 [1818])
14:13:10.652 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/local-patches.jar
14:13:10.652 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-embedder-3.0.2.jar
14:13:10.653 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-settings-3.0.2.jar
14:13:10.654 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-plugin-api-3.0.2.jar
14:13:10.654 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-model-builder-3.0.2.jar
14:13:10.655 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/plexus-sec-dispatcher-1.3.jar
14:13:10.656 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/plexus-cipher-1.4.jar
14:13:10.657 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-core-3.0.2.jar
14:13:10.661 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-model-3.0.2.jar
14:13:10.663 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-settings-builder-3.0.2.jar
14:13:10.664 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-repository-metadata-3.0.2.jar
14:13:10.668 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-artifact-3.0.2.jar
14:13:10.669 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-aether-provider-3.0.2.jar
14:13:10.672 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/plexus-interpolation-1.14.jar
14:13:10.673 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/maven-compat-3.0.2.jar
14:13:10.676 [main] DEBUG o.e.m.c.i.e.MavenEmbeddedRuntime -    Entry:/C:/RAD_Installs/803_06_24/IBM/SDP/p2/cic.p2.cache.location/pl
ugins/org.eclipse.m2e.maven.runtime_1.0.0.201106052308/jars/commons-cli-1.2.jar
```

3. **Maven Import Wizards:** Importing existing Maven projects is useful for easily creating Rational Application Developer projects for each Maven `pom.xml` file. For instance, if the root directory is pointing to a previous set of Maven example projects, each of these projects are imported, class path is configured, and the appropriate *face*ts are applied. Any directory that contains a valid `pom.xml` file appears in this dialog. The example shown below is based on the samples from the previous Rational Application Developer and Maven paper, and is mentioned below in the *Sample Scenarios* section.



Root Directory: C:\MAVEN\RAD_PROJECTS\Just_MAVEN

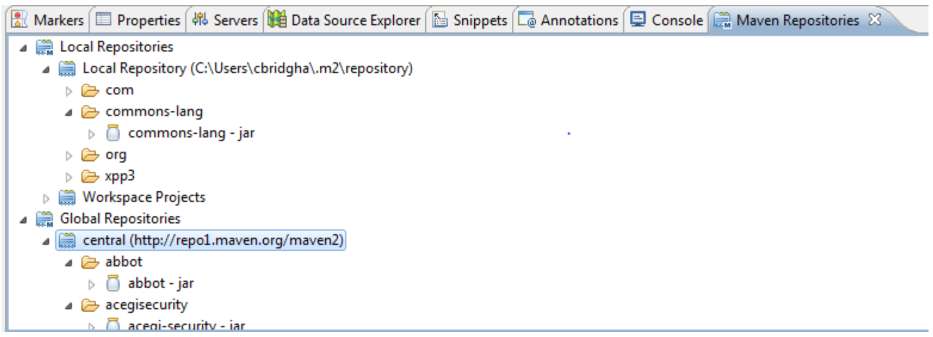
Projects:

- [✓] /SampleProject/pom.xml root:SampleProject:1.0:pom
  - [✓] servlets/pom.xml root.SampleProject:servlets:1.0:pom
    - [✓] MyCompanyWeb/pom.xml root.SampleProject.servlets:MyCompanyWeb:1.0:war
  - [✓] MyCompanyEJB/pom.xml root.SampleProject:MyCompanyEJB:1.0:ejb
  - [✓] MyCompanyEJBEAR/pom.xml root.SampleProject:MyCompanyEJBEar:1.0:ear
  - [✓] Utilities/pom.xml root.SampleProject:Utilities:1.0:pom
    - [✓] DataProject/pom.xml root.SampleProject.Utilities:DataProject:1.0:jar
    - [✓] MyCompanyEJBClient/pom.xml root.SampleProject.Utilities:MyCompanyEJBClient:1.0:jar

[✓] Add project(s) to working set

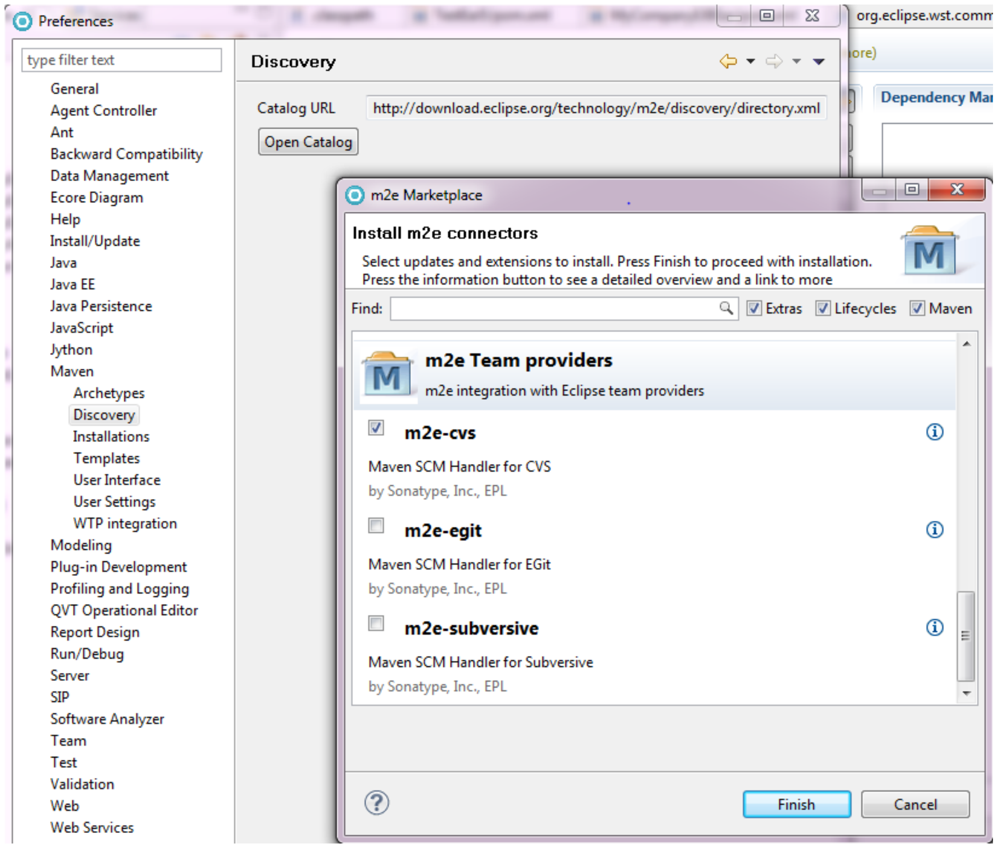
Working set: Dynamic Web Projects

4. **Maven Repository Browser:** Browsing and searching remote Maven repositories using the **Maven Repositories** view, available by selecting from the toolbar **Window → Show view → Other → Maven → Maven Repositories**



Local Repositories
Local Repository (C:\Users\cbridgha\.m2\repository)
com
commons-lang
commons-lang - jar
org
xpp3
Workspace Projects
Global Repositories
central (http://repo1.maven.org/maven2)
abbot
abbot - jar
acegisecurity
acegi-security - jar

5. **M2E Connectors:** Ability to connect to various software configuration management (SCM) systems and other extended features. To use the connectors from RAD, you must add the **Equinox p2 discovery** feature by selecting **Help → Install New Software**. The list of available connectors or extended features built on top of m2eclipse 1.0.0 appears by pressing the "Open Catalog" button from the Maven -> Discovery preference page. Extensions for Android, AspectJ, Groovy, eGit, and others are currently available.



Catalog URL http://download.eclipse.org/technology/m2e/discovery/directory.xml

Install m2e connectors

Select updates and extensions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more

[✓] Extras [✓] Lifecycles [✓] Maven

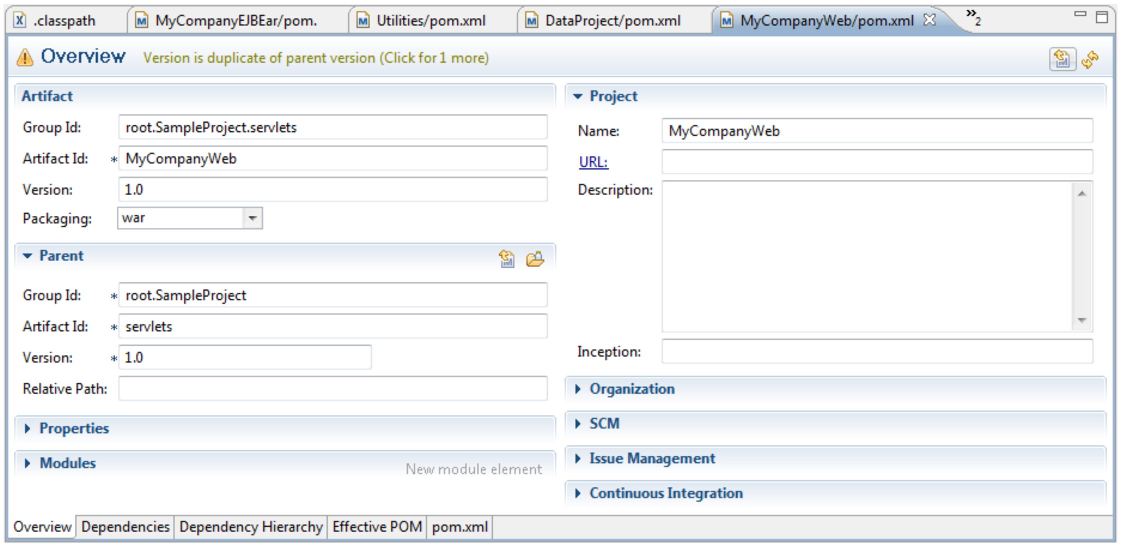
m2e Team providers
m2e integration with Eclipse team providers

[✓] m2e-cvs
Maven SCM Handler for CVS
by Sonatype, Inc., EPL

[ ] m2e-egit
Maven SCM Handler for EGit
by Sonatype, Inc., EPL

[ ] m2e-subversive
Maven SCM Handler for Subversive
by Sonatype, Inc., EPL

6. **Form-based POM.XML Editor:** The overview and dependencies pages allow form-based editing of the vast properties available for the Maven dependency model. The effective `pom.xml` is a read-only XML source view showing a fully populated model including all the defaults, and at last the `pom.xml` file itself using the source editor with semantic assistance from the Maven schema.



.classpath | MyCompanyEJBEar/pom. | Utilities/pom.xml | DataProject/pom.xml | MyCompanyWeb/pom.xml

Overview — Version is duplicate of parent version (Click for 1 more)

Artifact

Group Id:	root.SampleProject.servlets
Artifact Id: *	MyCompanyWeb
Version:	1.0
Packaging:	war

Parent

Group Id: *	root.SampleProject
Artifact Id: *	servlets
Version: *	1.0
Relative Path:	

Properties

Modules — New module element

Project

Name:	MyCompanyWeb
URL:	
Description:	
Inception:	

Organization

SCM

Issue Management

Continuous Integration

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

How does m2eclipse work? What should I expect?

Maven captures many of the semantic properties of a given project in the `pom.xml` file. This is the driving force behind the m2eclipse feature. Rational Application Developer traditionally stores project information in a variety of locations (`.classpath`, `.settings`, `MANIFEST.MF` and other files) *The most important tip to remember while developing with m2eclipse with the Web Tools Platform connector is to always manipulate the `pom.xml` file, and all other Rational Application Developer metadata gets updated automatically*. Making changes to any of the various Eclipse or Rational Application Developer metadata files create problems if the semantic equivalent is not captured in the `pom.xml` file. Properties and editors to avoid in Rational Application Developer include the **Java Build Path** and **Deployment Assembly** project property sheets, and the **MANIFEST.MF** editors. M2eclipse captures project dependencies and structure defined in the POM, and generates appropriate Rational Application Developer properties from the **Project Configurator** that runs incrementally or can be forced to run using the project pop-up menu **Maven –> Update project configuration**. As a convenience, Maven archetypes are a popular method for creating module projects initialized with relevant `pom.xml` settings. The `m2e-wtp` connector recognizes basic `pom` settings that represent module types. For example, any archetype that has the `<packaging>war</packaging>` fragment automatically results in the Java web facet being installed. The `m2e-wtp` connector is only intended to work on Java EE type projects, such as WAR, EAR, EJB, App Client,Connector projects, and including plain Java projects. Other archetypes supporting other programming models, such as Service Component Architecture (SCA) can not be recognized. Automatic Rational Application Developer configurations that setup specific Rational Application Developer facets does not occur, but manual configuration is an option in these cases.

Here is a mapping of project configurator actions based on maven project types according to m2e-wtp developers:

- WAR projects : Adds the Java and Dynamic Web Facets, based on maven-war-plugin configuration
- EJB projects : Adds the Java and EJB Facets, based on maven-ejb-plugin configuration
- EAR projects : Adds the EAR Facet, based on maven-ear-plugin configuration
- RAR projects : Adds the Java and Connector Facets, based on maven-rar-plugin configuration
- JAR projects packaged with JavaEE projects : Adds the Java and Utility facets

Let's take some example Maven `pom.xml` fragments, and show the resulting Rational Application Developer and Java EE metadata:

POM Fragment	Change in Rational Application Developer

`<packaging>war</packaging>`	Natures added to `.project`:
	`<nature>org.eclipse.wst.common.modulecore.ModuleCoreNature</nature>`
	`<nature>org.eclipse.wst.common.project.facet.core.nature</nature>`
	
	Java and Web facet added to `.settings/org.eclipse.wst.common.project.facet.core.xml`

`<build>` `<finalName>TestWar2</finalName>`	Changes `.settings/org.eclipse.wst.common.component` `<property name="context-root" value="TestWar2"/>`
`<build>` `<outputDirectory>${project.basedir}/anotherTarget/classes</outputDirectory>` `<testOutputDirectory>${project.basedir}/anotherTarget/classes</testOutputDirectory>`	Changes to output location of Java source folders
`<plugin>` `<groupId>org.apache.maven.plugins</groupId>` `<artifactId>maven-war-plugin</artifactId>` `<version>2.1.1</version>` `<configuration>` `<archive>` `<manifest>` `<addClasspath>true</addClasspath>` `<classpathPrefix>lib/</classpathPrefix>` `</manifest>` `<manifestEntries>` `<Ignore-Scanning-Packages>org.apache.avalon, org.apache.batik, org.apache.commons</Ignore-Scanning-Packages>` `</manifestEntries>` `</archive>` `</configuration>` `</plugin>`	Generated `MANIFEST.MF` file in the `target/m2e-wtp/web-resources/META-INF` folder `Manifest-Version: 1.0` `Built-By: cbridgha` `Build-Jdk: 1.6.0` `Created-By: Maven Integration for Eclipse` `Ignore-Scanning-Packages: org.apache.avalon, org.apache.batik, org.apa che.commons` `class-path: util.jar`

Maven Archiver and MANIFEST generation in Rational Application Developer

The `m2e + m2e-wtp` plugins depend on generating a `MANIFEST.MF`, and packages this file in the deployed module. If a `MANIFEST.MF` file exists in any source folder, it is required to move any existing `MANIFEST.MF` properties to the `pom.xml` archive sections, so they are generated correctly by the Maven builder. Then delete these files. For JAR, EJB, and Connector projects, the Maven generated `MANIFEST.MF` file is generated under `target/classes` folder. Web projects generate in `target/m2e-wtp/web-resources/` folder and EAR projects, generated in `target/m2e-wtp/ear-resources/folder`. Because this is an additional resource mapping to the projects; `root`, the single root validator shows warnings. The deployed application is copied first to the workspace metadata directory, but then be incrementally changed.

Information for packaging JAR files in EAR files rather than WAR files are located here:

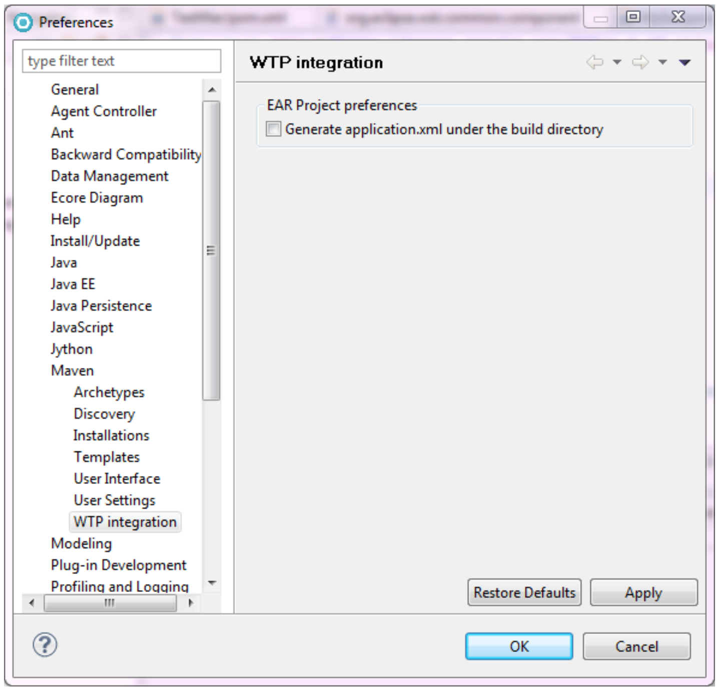
http://maven.apache.org/plugins/maven-war-plugin/examples/war-manifest-guide.html

More general information on regarding the Maven archiver can be found here:
http://maven.apache.org/shared/maven-archiver/index.html

Recommended Preferences

M2Eclipse preferences

M2Eclipse by default is installed with an embedded Maven runtime environment, and sets up a new local repository, but different runtime environments or repositories can be configured by using the various Maven preferences. If doing Java EE 5 development, generating deployment descriptors are optional. The **WTP Integration** preference for generating `application.xml` should be disabled in this case. You can find the **WTP Integration** preference page by going to the toolbar, select **Window → Preferences → Maven - > WTP integration**. Clear the **Generate application.xml under the build directory** check box.



WTP integration

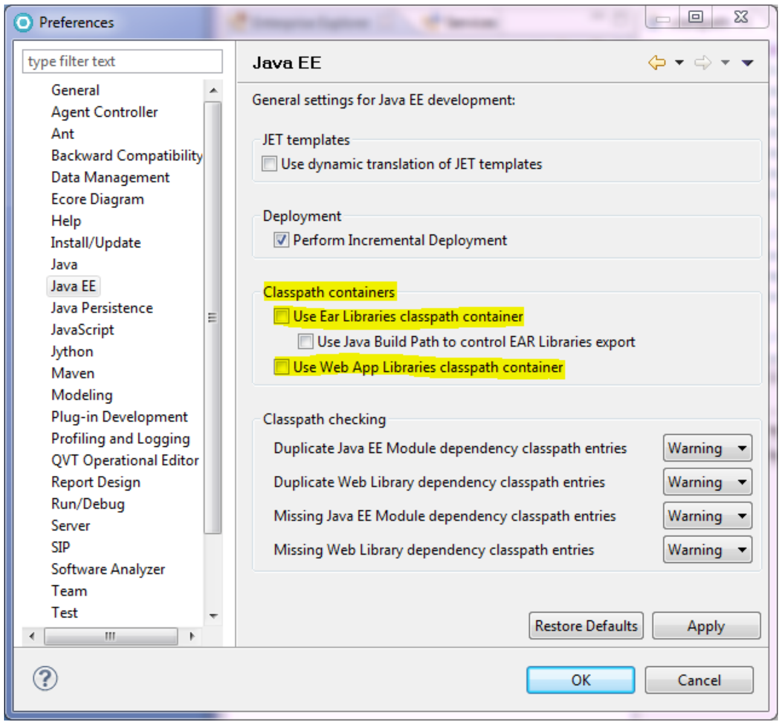
EAR Project preferences

☐ Generate application.xml under the build directory

Java EE preferences from Rational Application Developer

There are a few Java EE preferences that are recommended if using m2eclipse for project development. Many of these preferences affect the entire workspace; a separate workspace is first recommended for all non-maven projects.

1. **Classpath Containers** – In the **Java EE** preferences page (from the toolbar select **Window → Preferences → Java EE**), under the **Classpath containers** section, clear the **Use Ear Libraries classpath container** and **Use Web App Library classpath container check boxes**, such that projects does not use these class path containers, and rely on the Maven container to provide class path entries.



Java EE

General settings for Java EE development:

JET templates
- [ ] Use dynamic translation of JET templates

Deployment
- [x] Perform Incremental Deployment

Classpath containers
- [ ] Use Ear Libraries classpath container
- [ ] Use Java Build Path to control EAR Libraries export
- [ ] Use Web App Libraries classpath container

Classpath checking

Duplicate Java EE Module dependency classpath entries	Warning
Duplicate Web Library dependency classpath entries	Warning
Missing Java EE Module dependency classpath entries	Warning
Missing Web Library dependency classpath entries	Warning

2. **Java EE Project settings** – The default folder structure for creating Java EE projects or adding facets is available here. These settings are used when creating projects using the new project wizards from Rational Application Developer. If you are using the new Maven project wizards, defaults based on the chosen archetype are used. Most Java projects follow the same Maven defaults. There are some differences from the previous recommendations using the standard m2eclipse without Web Tools Platform support because of the additional resource mapping to the `temp` directory used for the `MANIFEST` generation. In the case of web projects, the output folder is left to the Maven default of `/target`, and the deployment is required to *assemble* the application files in the <*workspace*>`\.metadata\.plugins\org.eclipse.wst.server.core` folder. In addition, the setting to create an EAR project when creating a new module project has been cleared, because this configuration should be handled only in the `pom.xml` of the EAR project.

Use these values in the **Window → Preferences →Java EE → Project** preference page:

a. Under **Enterprise Application membership**, clear the **Add project to an EAR** check box.

b. Under **Enterprise Application Project**, in the **Content Directory** field, type `src/main/application`

c. Under **Dynamic Web project,** type the following values for each field:

- **Default Source Folder :** `src/main/resources`
- **Output Folder**: `target/classes`
- **Content Directory:** `src/main/webapp`

d. Under **EJB Project,** type the following values for each field**:**

- **Default Source Folder:** `src/main/resources`
- **Output Folder**: `target/classes`

e. Under **Application Client Project,** type the following values for each field**:**

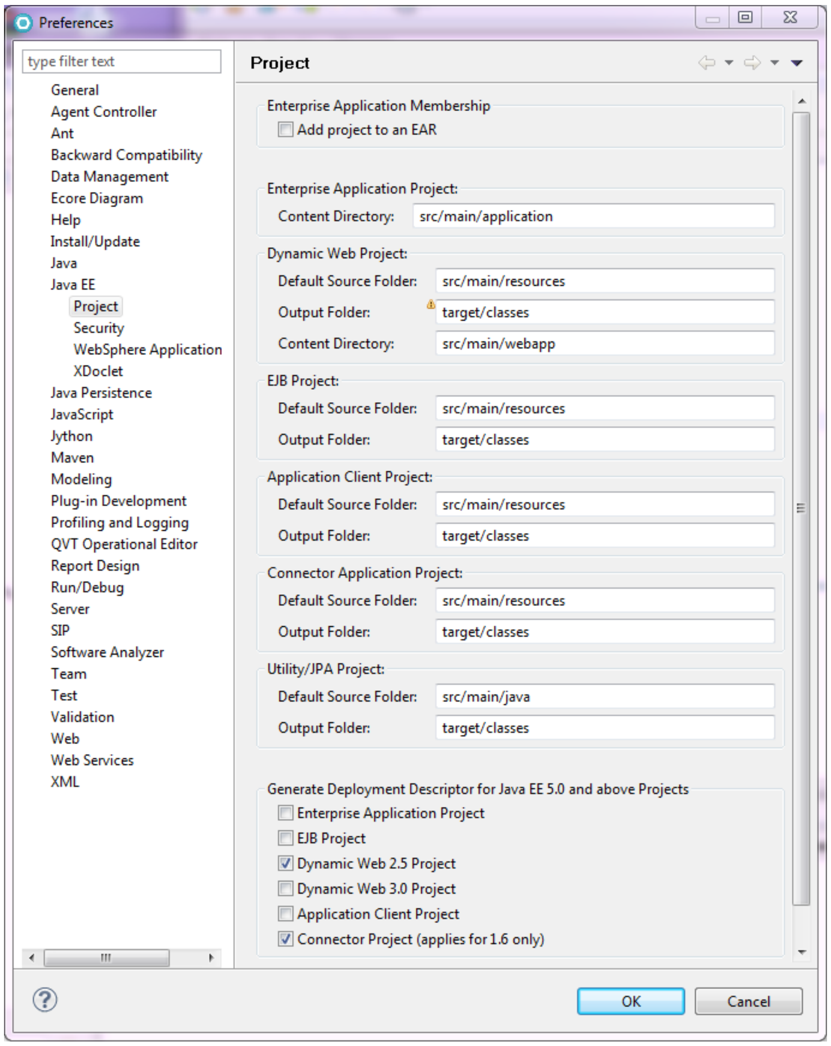
- **Default Source Folder:** `src/main/resources`
- **Output Folder**: `target/classes`

f. Under **Connector Application Project,** type the following values for each field**:**

- **Default Source Folder:** `src/main/resources`
- **Output Folder**: `target/classes`

g. Under **Utility/JPA Project,** type the following values for each field**:**

- **Default Source Folder:** `src/main/java`
- **Output Folder**: `target/classes`



Project

Enterprise Application Membership
- [ ] Add project to an EAR

Enterprise Application Project:
Content Directory: src/main/application

Dynamic Web Project:
Default Source Folder: src/main/resources
Output Folder: target/classes
Content Directory: src/main/webapp

EJB Project:
Default Source Folder: src/main/resources
Output Folder: target/classes

Application Client Project:
Default Source Folder: src/main/resources
Output Folder: target/classes

Connector Application Project:
Default Source Folder: src/main/resources
Output Folder: target/classes

Utility/JPA Project:
Default Source Folder: src/main/java
Output Folder: target/classes

Generate Deployment Descriptor for Java EE 5.0 and above Projects
- [ ] Enterprise Application Project
- [ ] EJB Project
- [x] Dynamic Web 2.5 Project
- [ ] Dynamic Web 3.0 Project
- [ ] Application Client Project
- [x] Connector Project (applies for 1.6 only)

A copy of these settings have been provided in a preference file (*.epf file), and can be imported into your workspace using the **Import Preferences** wizard by selecting from the toolbar **File → Import → General → Preferences**.

Sample Scenarios

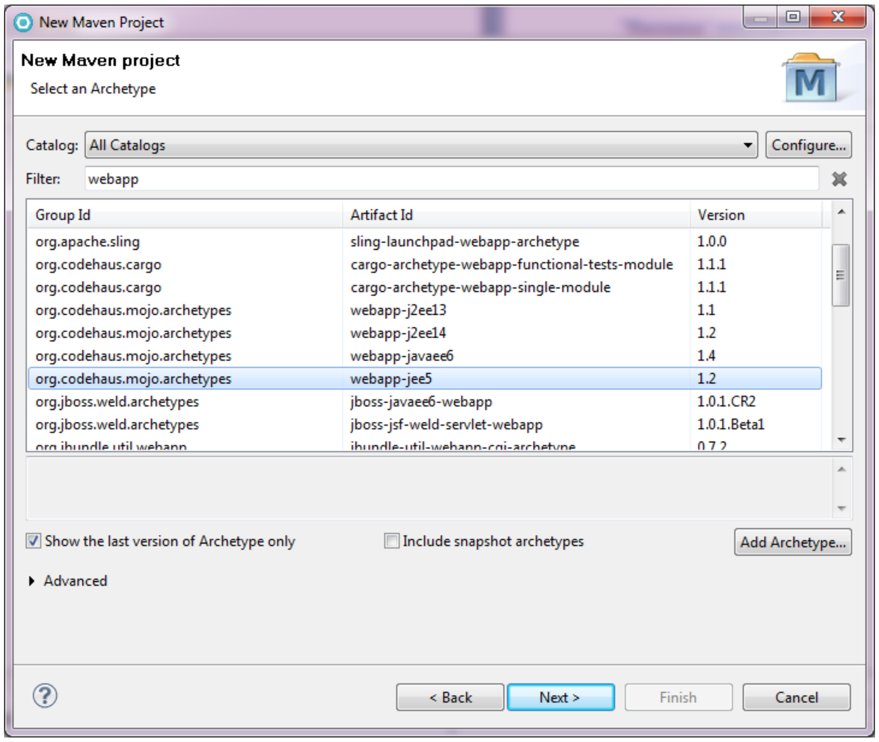
Creating Maven web project

M2eclipse contains enhancements to the enriched set of web development tools from Rational Application Developer which improves the experience of creating a Maven web project. In this example, start by creating a project based on a popular archetype.

1. Select from the toolbar **File → New → Project → Maven → Maven Project**. Click **Next**.

2. In the **New Maven Project** wizard, select **Next**.

3. On the **New Maven project** page, you can use the **Filter** field to search for an archetype by specifying your search texts. Type `webapp` in the **Filter** field, under the **Artifact Id** column find the `webapp-jee5` archetype, and then click **Next**.
Tip: Allow the Maven indexer to finish searching the nexus repository, or these types cannot be found. The indexing process can take about 30 minutes to complete.



Catalog: All Catalogs

Filter: webapp

Group Id	Artifact Id	Version
org.apache.sling	sling-launchpad-webapp-archetype	1.0.0
org.codehaus.cargo	cargo-archetype-webapp-functional-tests-module	1.1.1
org.codehaus.cargo	cargo-archetype-webapp-single-module	1.1.1
org.codehaus.mojo.archetypes	webapp-j2ee13	1.1
org.codehaus.mojo.archetypes	webapp-j2ee14	1.2
org.codehaus.mojo.archetypes	webapp-javaee6	1.4
org.codehaus.mojo.archetypes	webapp-jee5	1.2
org.jboss.weld.archetypes	jboss-javaee6-webapp	1.0.1.CR2
org.jboss.weld.archetypes	jboss-jsf-weld-servlet-webapp	1.0.1.Beta1
org.ibundle.util.webapp	ibundle-util-webapp-cgi-archetype	0.7.2

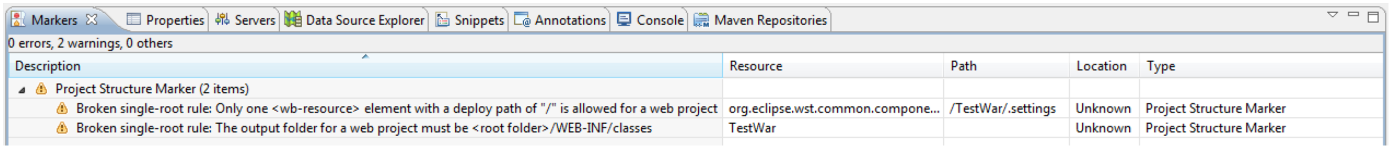
4. In the **Group Id** field, type `mygroup`.

5. In the **Artifact Id** field, type `TestWar`, and click **Finish**.

6. The above setting creates a web project with the default Maven folder settings. However, the project is not yet targeted to a runtime environment.

7. In the **Enterprise Explorer** view, right-click the **TestWar** project and select **Properties → Targeted Runtimes.** Use the **Targeted Runtimes** preference page to specify the server type, such as WebSphere® Application Server V7.0.

8. Open the **Markers** view from the **Java EE** perspective, and see the warnings reported.



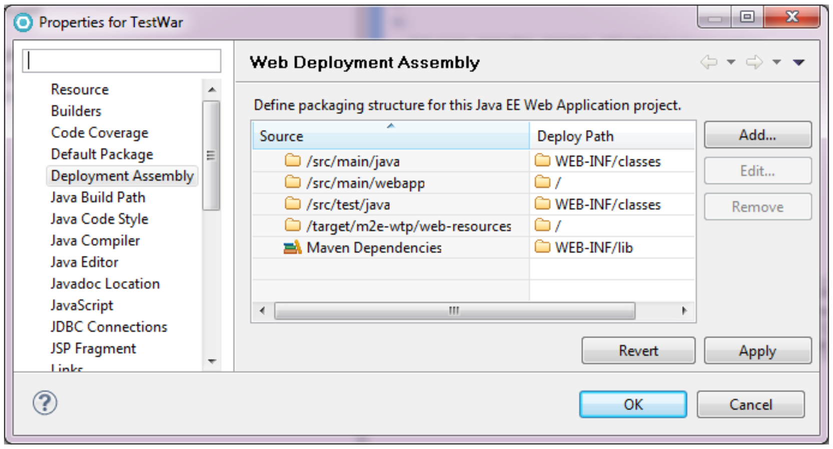
0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Project Structure Marker (2 items)				
Broken single-root rule: Only one <wb-resource> element with a deploy path of "/" is allowed for a web project	org.eclipse.wst.common.compone...	/TestWar/.settings	Unknown	Project Structure Marker
Broken single-root rule: The output folder for a web project must be <root folder>/WEB-INF/classes	TestWar		Unknown	Project Structure Marker

9. These warnings are an indication that deployment of this project cannot be done without *copying* the files to the temporary workspace metadata location, which slows initial deployment performance.

10. In the **Enterprise Explorer** view, right-click the **TestWar** project and select **Properties → Deployment Assembly** to verify the folder mappings. Observe that for web projects, the entire class path container **Maven Dependencies** is mapped to the `WEB-INF/lib` location. The folder `/target/m2e-wtp/web-resources` is used to temporarily generate files such as `MANIFEST.MF`. Generation of the `MANIFEST` based on Maven archiver options is important if you want to develop *skinny* WAR projects, such as packaging dependant JAR files in the EAR rather than in the `web app lib` directory.

11. In the **Web Deployment Assembly** properties page, click **OK**.



Define packaging structure for this Java EE Web Application project.

Source	Deploy Path
/src/main/java	WEB-INF/classes
/src/main/webapp	/
/src/test/java	WEB-INF/classes
/target/m2e-wtp/web-resources	/
Maven Dependencies	WEB-INF/lib

12. Create a Servlet class.

a. From the toolbar select **File → New → Other → Web - >Servlet** and click **Next**.

b. In the **Java package** field for the **Create Servlet** wizard, type `test`

c. In the **Class name** field, type `TestServlet`

d. Accept all other defaults. Click **Finish**.

13. The `TestServlet.java` editor opens, replace the existing `doGet` method with the following code:

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            response.setContentType("text/html");
             PrintWriter pw = response.getWriter();
             pw.println("<html>");
             pw.println("<head><title>Hello World</title></title>");
```

```
        pw.println("<body>");
        pw.println("<h1>Hello World</h1>");
        pw.println("</body></html>");
      }
```

14. There is an error message that the `PrintWriter` cannot be resolved to a type. To fix this error message, right-click on the `TestServlet.java` editor and select **Source → Organize Imports**. The `import java.io.PrintWriter;` gets added to the `TestServlet.java` source. Save the changes by selecting in the toolbar **File → Save.**

15. To test the WAR project on WebSphere Application Server, you must contain the web project in an EAR. The next scenario is going to cover creating an EAR project using traditional Java EE wizards from Rational Application Developer, and convert the application to using Maven.

"Mavenize" existing Rational Application Developer projects

An existing or newly created Java EE project from Rational Application Developer can be configured to use Maven, and must be converted to properly interact within the Maven project dependency model. Here are the instructions to create and configure an EAR project:

1. Create a new EAR project.

a. Open the **New EAR Application Project** wizard by selecting from the toolbar **File → New → Project → Java EE → Enterprise Application Project** and click **Next**.,

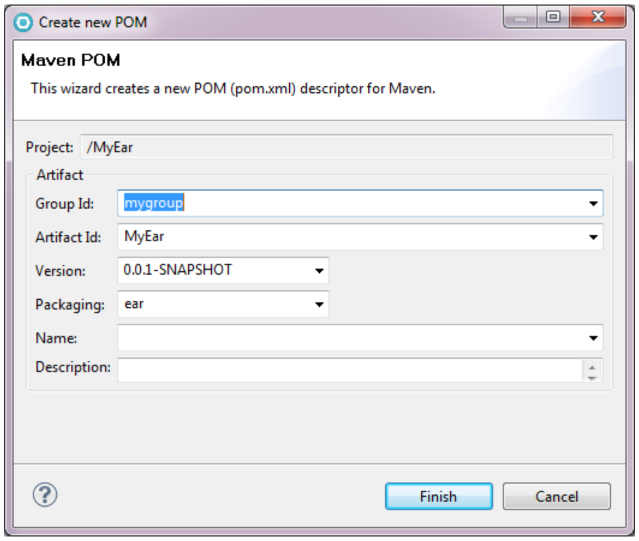
b. In the **Project name** field, type `MyEar`.

c. In the **EAR version** list, select **5.0**.

d. In **Target runtime** list, ensure the project has a valid runtime environment, such as WebSphere Application Server V7.0 or V8.0, and click **Finish**. In the following steps, we are going to add the WAR module to the project through the pom.

2. In the **Enterprise Explorer** view, right-click the **MyEar** project and select **Configure → Convert To Maven Project**.

3. In the **Maven POM** page of the **Create new POM** wizard, type `mygroup` for the **Group Id** field. There is a known limitation with the packaging options in this wizard, you must manually type `ear` for the **Packaging** field. Click **Finish**. A sparsely populated `pom.xml` file is created along with project metadata, such as Maven natures and builders.


Create new POM

Maven POM

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /MyEar

Artifact

Group Id: mygroup

Artifact Id: MyEar

Version: 0.0.1-SNAPSHOT

Packaging: ear

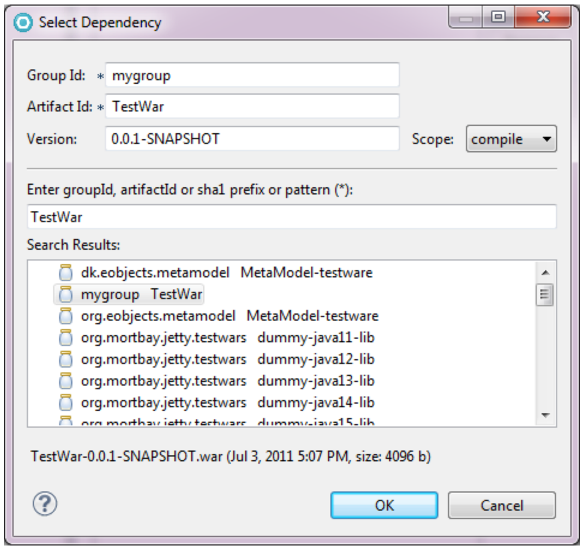
Name:

Description:

4. The pom editor opens, click on the **Dependencies** tab.

5. Under the **Dependencies** section, click the **Add**.

6. In the **Select Dependency** wizard, type `TestWar` in the filter text box. Under the **Search Results** list, select `mygroup TestWar` and click **OK**.


Select Dependency

Group Id: * mygroup

Artifact Id: * TestWar

Version: 0.0.1-SNAPSHOT Scope: compile

Enter groupId, artifactId or sha1 prefix or pattern (*):

TestWar

Search Results:

dk.eobjects.metamodel MetaModel-testware
mygroup TestWar
org.eobjects.metamodel MetaModel-testware
org.mortbay.jetty.testwars dummy-java11-lib
org.mortbay.jetty.testwars dummy-java12-lib
org.mortbay.jetty.testwars dummy-java13-lib
org.mortbay.jetty.testwars dummy-java14-lib
org.mortbay.jetty.testwars dummy-java15-lib

TestWar-0.0.1-SNAPSHOT.war (Jul 3, 2011 5:07 PM, size: 4096 b)

7. In the pom editor, select the **pom.xml** tab to view the source. Add the below fragment of code after this line of code: `<packaging>ear</packaging>`. The fragment of code properly aligns the maven EAR plugin settings with the Rational Application Developer project, such as configuring the following settings: set the EAR version to 5.0, do not generate deployment descriptor, specify the archive name, specify the location of the source folder and other settings.

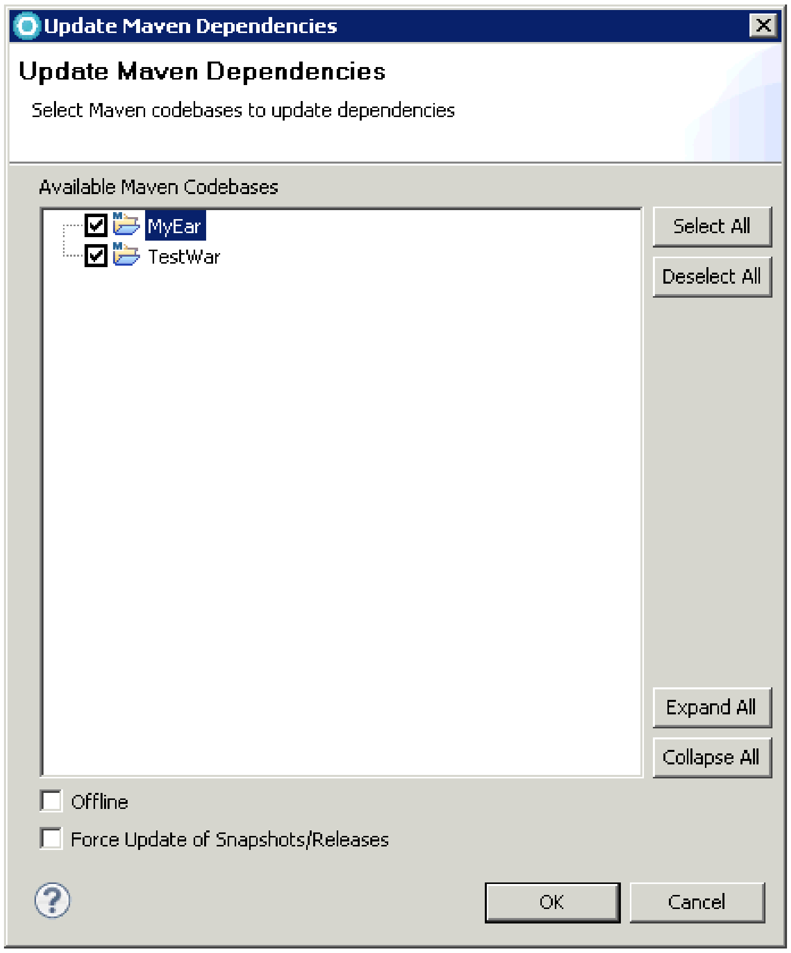
```
<build>
```

```
      <finalName>MyEar</finalName>
      <plugins>
            <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-ear-plugin</artifactId>
                  <version>2.5</version>
                  <configuration>
                        <version>5</version>
                        <modules>
                              <webModule>
                                    <groupId>mygroup</groupId>
                                    <artifactId>TestWar</artifactId>
                                    <bundleFileName>TestWar.war</bundleFileName>
                              </webModule>
                        </modules>
                        <fileNameMapping>no-version</fileNameMapping>

      <generateApplicationXml>false</generateApplicationXml>

      <earSourceDirectory>${basedir}\src\main\application</earSourceDirectory
>
                  </configuration>
            </plugin>
      </plugins>
   </build>
```

8. In the **Enterprise Explorer** view, right-click the MyEar project and select **Maven → Update Project Configuration**. Select both **MyEar** and **TestWar** check boxes and click **OK**.


Update Maven Dependencies

Update Maven Dependencies

Select Maven codebases to update dependencies

Available Maven Codebases

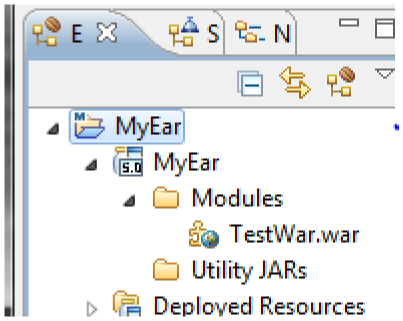
- [✓] MyEar
- [✓] TestWar

[ ] Offline

[ ] Force Update of Snapshots/Releases

9. In **Enterprise Explorer** view, expand **src → main → application→ META-INF** folder and right-click `application.xml` file and select **Delete.** The `application.xml` file was initially generated by Maven before the `pom.xml` changes were applied.

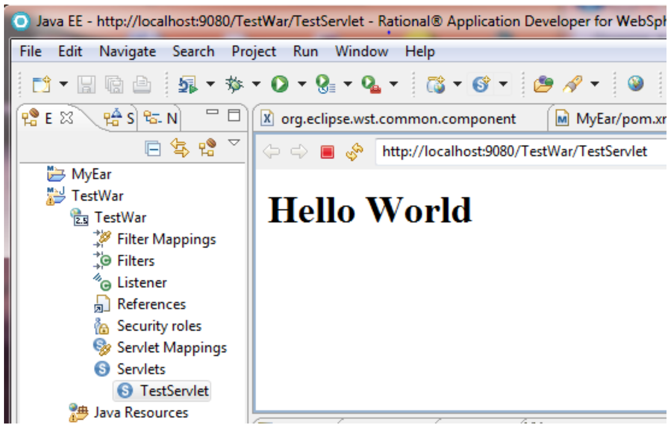
10. In the **Enterprise Explorer** view, verify the WAR module is now part of the EAR.



WebSphere Application Server deployment

1. Test the Maven application by publishing it on the server. In the **Enterprise Explorer** view, expand **TestWar → TestWar → Servlets** and right-click **TestServlets** and select **Run As… → Run on Server.**

2. In the **Run On Server** wizard, under the **Select the server that you want to use** list, select the WebSphere Application Server entry, and click **OK.**


Java EE - http://localhost:9080/TestWar/TestServlet - Rational® Application Developer for WebSph
File Edit Navigate Search Project Run Window Help
org.eclipse.wst.common.component
MyEar/pom.xr
http://localhost:9080/TestWar/TestServlet
MyEar
TestWar
TestWar
Filter Mappings
Filters
Listener
References
Security roles
Servlet Mappings
Servlets
TestServlet
Java Resources

Hello World

Full Java EE Maven sample

In the previous Java EE Development using Rational Application Developer 7.5.5 and Maven paper, an example was used to demonstrate an application with various Maven project types. This sample has been provided again as an example, with changes following the best practices for m2eclipse development. To run the sample complete the following steps:

1. Import the sample.

a) Save a copy of the CompanySample.zip file into a known folder location.

b) In the toolbar select **Import… → General → Existing Projects into Workspace.**

c) On the **Select** page of the **Import** wizard, click **Next**.

d) In the **Import Project** page, select **Select archive file** option, browse to the location of the sample saved on your file system, and then click **Open**.

e) In the **Import** wizard, click **Finish**.

2. The default environment for this sample is set to WebSphere Application Server V7.0. If the **Workspace Migration** wizard display, click **Next**.

1. On the **Workspace projects which need migration** page, click **Next**.

2. On the **Migration Project Resources** page, click Next.

3. In the **Server Runtimes** list on the **Undefined Server Runtime** page, for the `was.base.v7` entry, select **WebSphere Application Server v8.0** or **WebSphere Application Server v7.0** under the **New Server Runtime** column. Click **Next**.

4. On the **Complete Migration Startup** page, click **Finish**.

5. On the **Migration Validation** window, click **OK**.

3. A sample of a Derby database is also provided to use with the sample above.

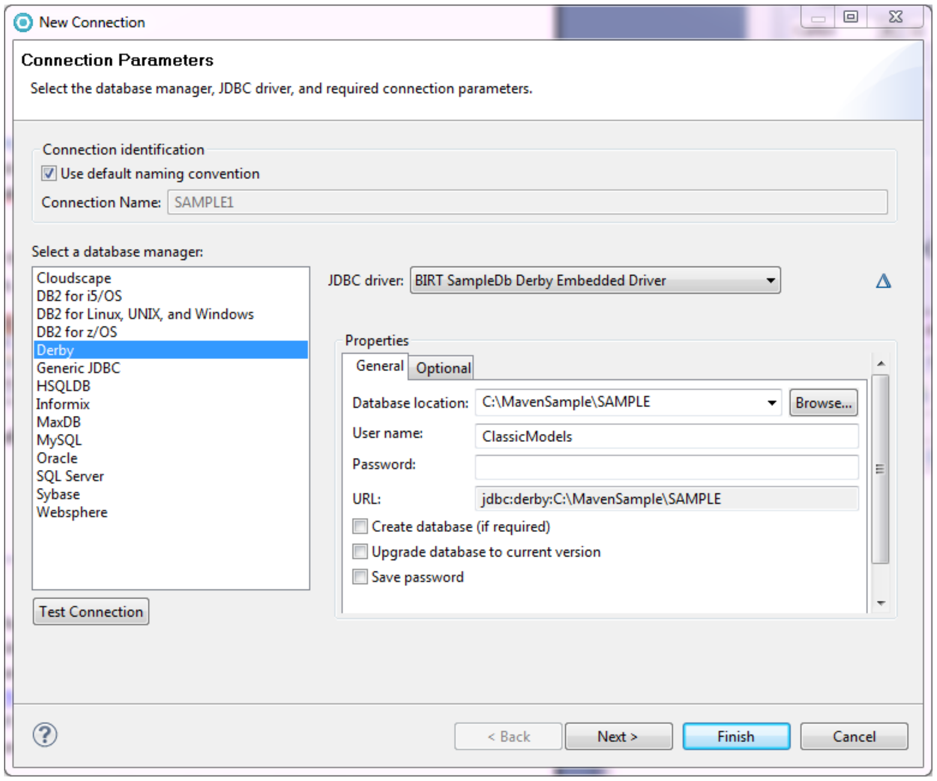
a) Unzipping the database contents to a known folder location.

b) Create a Derby database connection from the **Data Source Explorer** view from Rational Application Developer. In the **Data Source Explorer** view, right-click the **Database Connections** folder and select **New.**

c) Under the **Select a database manager,** select **Derby**. An existing JDBC driver should be found: `BIRT SampleDb Derby Embedded Driver`.

d) In the **Database location** field, browse to the folder where the unzip database sample exists and then click **OK**.

e) In the **New Connection** wizard, click **Finish**.


New Connection

Connection Parameters

Select the database manager, JDBC driver, and required connection parameters.

Connection identification

[✓] Use default naming convention

Connection Name: SAMPLE1

Select a database manager: Derby

JDBC driver: BIRT SampleDb Derby Embedded Driver

Properties — General

Database location: C:\MavenSample\SAMPLE

User name: ClassicModels

Password:

URL: jdbc:derby:C:\MavenSample\SAMPLE

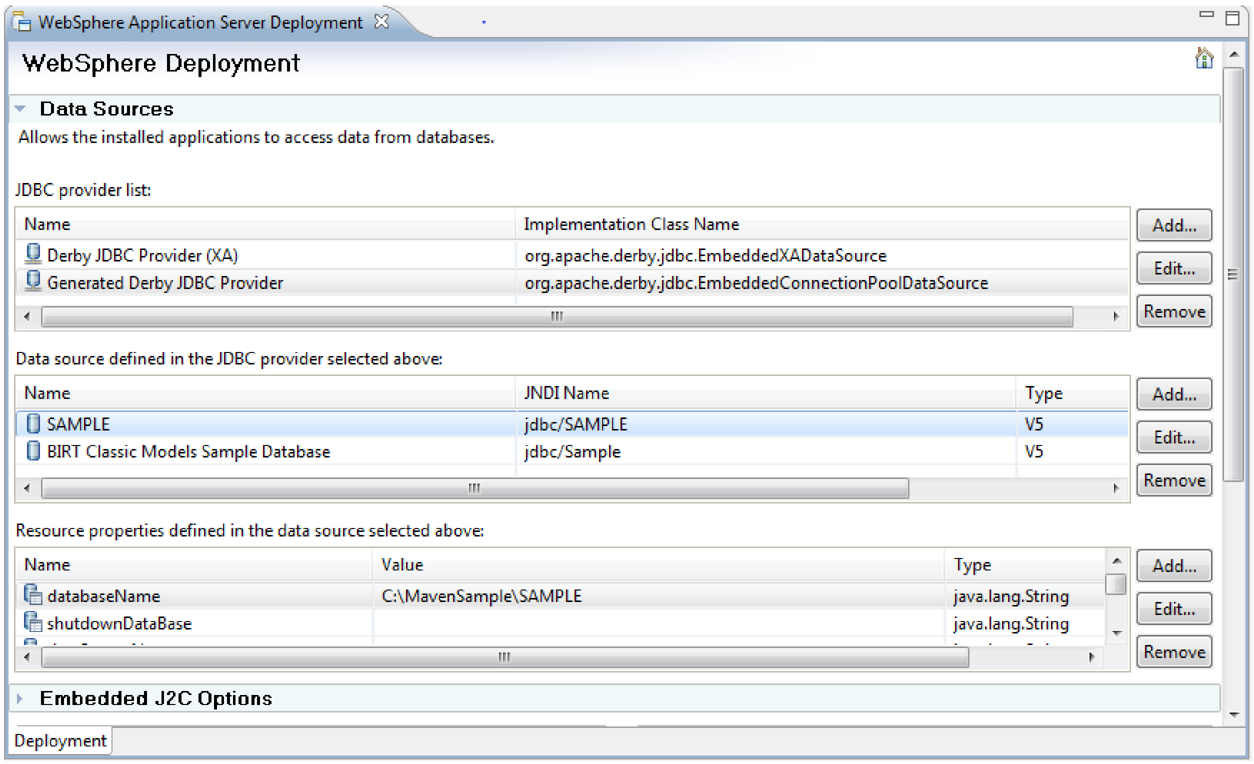
[ ] Create database (if required)

[ ] Upgrade database to current version

[ ] Save password

4. In the **Enterprise Explorer** view, right-click **DataProject** JPA project and select **Properties → JDBC Connections.** Verify the new **SAMPLE** connection is associated with this project.

5. In the **Enterprise Explorer** view, right-click **MyCompanyEJBEar** project and select **Java EE → Open Websphere Application Server Deployment.** Verify the WebSphere Application Server data source binding to `jdbc/SAMPLE` is also using this connection data.



WebSphere Deployment

Data Sources

Allows the installed applications to access data from databases.

JDBC provider list:

Name	Implementation Class Name
Derby JDBC Provider (XA)	org.apache.derby.jdbc.EmbeddedXADataSource
Generated Derby JDBC Provider	org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource

Data source defined in the JDBC provider selected above:

Name	JNDI Name	Type
SAMPLE	jdbc/SAMPLE	V5
BIRT Classic Models Sample Database	jdbc/Sample	V5

Resource properties defined in the data source selected above:

Name	Value	Type
databaseName	C:\MavenSample\SAMPLE	java.lang.String
shutdownDataBase		java.lang.String

6. This sample also refers to several installed JAR files that need to be added to the local repository for a clean Maven build. The JAR files for the WebSphere Application Server runtime environment are located in your local installation of WebSphere Application Server. Installing these JAR files into your local repository can be done in two ways. The first method is if you have a local installation of Maven on your workstation, you need to execute scripts on the command prompt. Here is an example of running the scripts if you are working with a WebSphere Application Server V7.0:

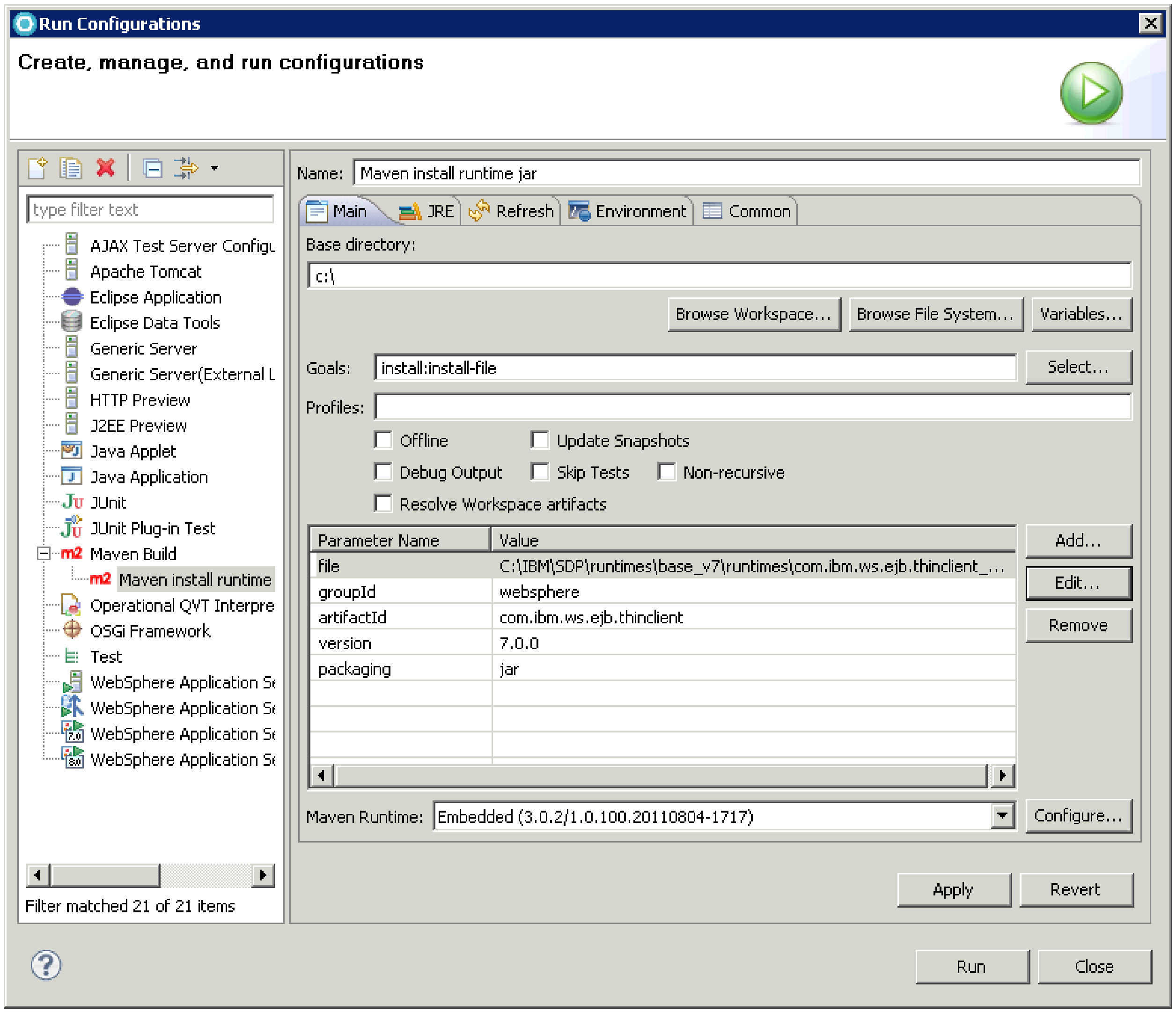
```
mvn install:install-file -
Dfile=C:\IBM\SDP\runtimes\base_v7\runtimes\com.ibm.ws.ejb.thinclient_7
.0.0.jar -
DgroupId=websphere -DartifactId=com.ibm.ws.ejb.thinclient -
Dversion=7.0.0 -
Dpackaging=jar
```

```
mvn install:install-file -
Dfile=C:\IBM\SDP\runtimes\base_v7\runtimes\com.ibm.ws.jpa.thinclient_7
.0.0.jar -
DgroupId=websphere -DartifactId=com.ibm.ws.jpa.thinclient -
Dversion=7.0.0 -
Dpackaging=jar

mvn install:install-file -
Dfile=C:\IBM\SDP\runtimes\base_v7\runtimes\com.ibm.ws.admin.client_7.0
.0.jar -DgroupId=websphere -DartifactId=com.ibm.ws.admin.client -
Dversion=7.0.0 -
Dpackaging=jar

mvn install:install-file -
Dfile=<workspace_dir>\MyCompanyEJBEar\src\main\application\MyCompanyUt
ilities.jar -
DgroupId=root.SampleProject.Utilities -DartifactId=MyCompanyUtilities
-Dversion=1.0 -
Dpackaging=jar
```

The second method is using the embedded Maven runtime environment within Rational Application Developer. Specify a run configuration of type **Maven build** using the goal **install:install-file**. Here is an example screen capture of the **Maven build** run configuration:



Run Configurations

Create, manage, and run configurations

Name: Maven install runtime jar

Base directory: c:\

Goals: install:install-file

Profiles:

Parameter Name	Value
file	C:\IBM\SDP\runtimes\base_v7\runtimes\com.ibm.ws.ejb.thinclient_...
groupId	websphere
artifactId	com.ibm.ws.ejb.thinclient
version	7.0.0
packaging	jar

Maven Runtime: Embedded (3.0.2/1.0.100.20110804-1717)

Filter matched 21 of 21 items

7. To run the sample:

f) In the **Enterprise Explorer** view, expand **MyCompanyWeb → MyCompanyWeb → Servlets** . Right-click **DepartmentSalarySearch** and select **Run As → Run on Server**.

g) Under the **Select the server that you want to use** list, select the WebSphere Application Server entry, and click **OK.**

Known issues

Here is a list of known issues and restrictions

EJB Client projects

If creating EJB projects using the EJB wizards from Rational Application Developer, clear the option to create an EJB Client project because the proper Maven dependencies are not properly created. Create a new Maven project with JAR packaging, and add a dependency to the EJB project pointing to this new project (using the `pom` editor).

What Java EE version module are you developing?

Rational Application Developer and Maven have different mechanisms for tracking what is the intended specification version of the module project, and keeping these mechanisms in sync is important to avoid errors that could occur. If these problems arise, synchronize the **facet** version of the project and the module version specified within the `pom.xml`, usually within the plugins `<configuration>` tag.

What's next?

The Rational's tools development team is continuously improving the environment for various frameworks such as Maven and m2e, and this paper evolves as problems are resolved, and enhancements are offered. The m2eclipse project and various connector extensions continue to improve in quality and function, and collaboration between the m2eclipse and other Eclipse projects continues to strengthen. In March 2012 it was announced that a new incubator project at Eclipse was proposed and accepted for m2e-wtp. This project will enable both m2e and wtp projects to share, and integrate with a common theme and goals. Updates to this paper are forthcoming as significant improvements become available. Questions and feedback can be added using the Rational Application Developer tools forum, or by emailing the authors.

References

Java EE Development using Rational Application Developer 7.5.5 and Maven
http://www.ibm.com/developerworks/wikis/download/attachments/113607155/RAD_755_MAVEN_0601.pdf?version=1

Rational Desktop Tools forum
http://www.ibm.com/developerworks/forums/forum.jspa?forumID=430

Apache Maven website
http://maven.apache.org/

M2Eclipse project site
http://www.eclipse.org/m2e/

M2Eclipse-WTP Wiki
https://github.com/sonatype/m2eclipse-wtp/wiki