Edition 1.3

m2eclipse 0.12

m2eclipse

Maven Integration for Eclipse

Sonatype

Developing with Eclipse and Maven

Tim O'Brien
Jason van Zyl
Benjamin Bentmann
Igor Fedorenko
Brian Fox
Milos Kleint
Peter Lynch
Matthew Piggott
Pascal Rapicault
Rich Seddon
Vlad Tatavu
Dan Yocum

A Sonatype Open Book
Mountain View, CA

Published by:

Sonatype, Inc.
12501 Prosperity Drive
Suite 350
Silver Spring, MD 20904

For online information and ordering of this and other Sonatype books, please visit www.sonatype.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact:
book@sonatype.com
.

ISBN 978-0-9842433-1-0

Editor: Tim O'Brien

Copyright

Online version published by Sonatype, Inc., 800 W. El Camino Real, Suite 400, Mountain View, CA, 94040.

Foreword: 1.3

If you would like to receive updates whenever there is a change to this book, you can subscribe to our Book Announcement mailing list. To subscribe to this mailing list, send an email to: book-notify-subscribe@sonatype.org[1]

We welcome your feedback, please do not hesitate to contact the Sonatype team with any questions or ideas you may have about the product.

If you have any feedback or questions, you are encouraged to post an item in our Get Satisfaction page for this book: http://getsatisfaction.com/sonatype/products/sonatype_developing_with_eclipse_and_maven.

Tim O'Brien, Sonatype

May, 2011

Edition: 1.3

1. Changes in Edition 1.3

The following changes were made in 1.3:

- Update to book site template for the new Sonatype web site.

[1] mailto:book-notify-subscribe@sonatype.org

Chapter 1. Introduction to m2eclipse

1.1. Introduction

The Eclipse IDE is the most widely used IDE for Java development today. Eclipse has a huge amount of plugins (see http://www.eclipseplugincentral.com/) and an innumerable amount of organizations developing their own software on top of it. Quite simply, Eclipse is ubiquitous. The m2eclipse[1] project, provides support for Maven within the Eclipse IDE, and, in this chapter, we will explore the features it provides to help you use Maven with Eclipse.

1.2. m2eclipse

The m2eclipse plugin (http://m2eclipse.sonatype.org/) provides Maven integration for Eclipse. m2eclipse also has hooks into the features of both the Subclipse plugin (http://subclipse.tigris.org/) and the Mylyn plugin (http://www.eclipse.org/mylyn/). The Subclipse plugin provides the m2eclipse plugin with the ability to interact with Subversion repositories, and the Mylyn plugin provides the m2eclipse plugin with the ability to interact with a task-focused interface that can keep track of development context. Just a few of the features m2eclipse provides include:

- Creating and importing Maven projects
- Dependency management and integration with the Eclipse classpath
- Automatic dependency downloads and updates
- Artifact Javadoc and source resolution
- Creating projects with Maven Archetypes
- Browsing and searching remote Maven repositories
- POM management with automatic update to dependency list
- Materializing a project from a Maven POM
- Checking out a Maven project from several SCM repositories
- Adapting nested multi-module Maven projects to the Eclipse IDE
- Integration with Web Tools Project (WTP)
- Integration with Subclipse
- Integration with Mylyn
- Form-based POM Editor
- Graphical Display of Dependency Graph
- GUI Presentation of Dependency Tree and Resolved Dependencies

There are many more features in m2eclipse beyond the list above and this chapter introduces some of the more impressive features that are currently available. Let's get started by installing the m2Eclipse plugin.

[1] http://m2eclipse.codehaus.org/

Chapter 2. Installing m2eclipse

2.1. Installing the Eclipse IDE

Sonatype recommends installing m2eclipse on Eclipse 3.5 or Eclipse 3.6, and to make use of m2eclipse you will need to install the JDK. To download an Eclipse IDE distribution:

1. Go to http://www.eclipse.org/downloads/ in a web browser.

2. Download an Eclipse distribution. Note: If you want to use the m2eclipse WTP integration, download the Eclipse IDE for JavaEE Developers distribution.

3. Install Eclipse - for more information about installing the Eclipse IDE, see the Eclipse Wiki[1].

2.2. Installing m2eclipse in Eclipse 3.6 (Helios) with the Eclipse Marketplace

The Eclipse Marketplace makes it easy to install m2eclipse and m2eclipse (Extras) in Eclipse 3.6 (Helios). To open the Eclipse Marketplace, go to the Eclipse Help menu and select Eclipse Marketplace... as shown in Figure 2.1, “Opening the Eclipse Marketplace”.

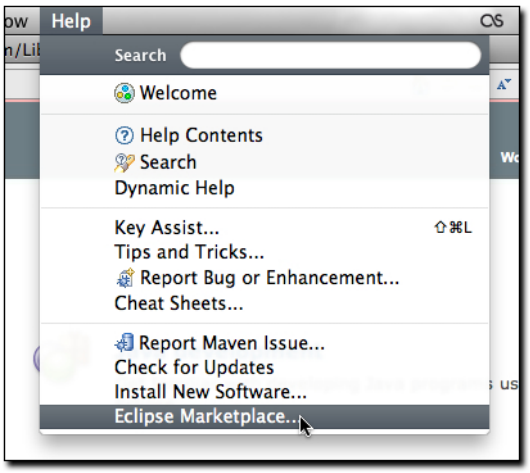

Figure 2.1. Opening the Eclipse Marketplace

The first time you open the Eclipse Marketplace, you will be asked to select a Marketplace. Select the Eclipse Marketplace as shown in Figure 2.2, “Selecting the Eclipse Marketplace”.

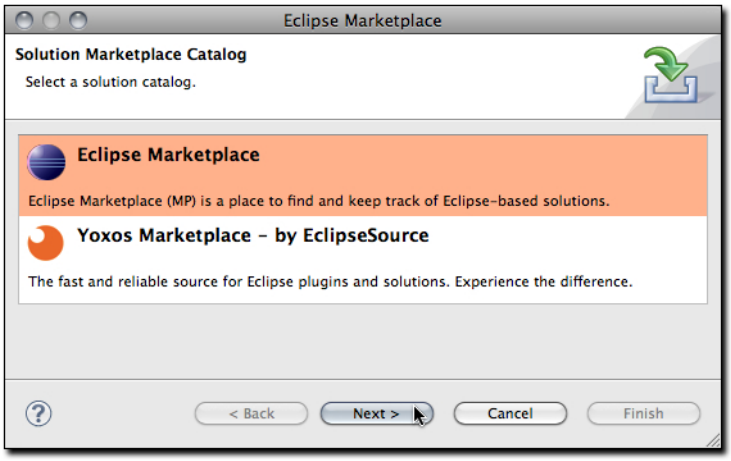

Figure 2.2. Selecting the Eclipse Marketplace

2.2.1. Installing Maven Integration for Eclipse (Core)

Maven Integration for Eclipse is separated into two components: the Core of Maven Integration for Eclipse and an optional package of extra, unsupported components. To install the core component of Maven Integration for Eclipse, open the Eclipse Marketplace, select the Search tab, and search for "Maven Integration" as shown in Figure 2.3, “Selecting Maven Integration for Eclipse from Eclipse Marketplace”. Click on the Install button to the right of the second item listed in Figure 2.3, “Selecting Maven Integration for Eclipse from Eclipse Marketplace”.

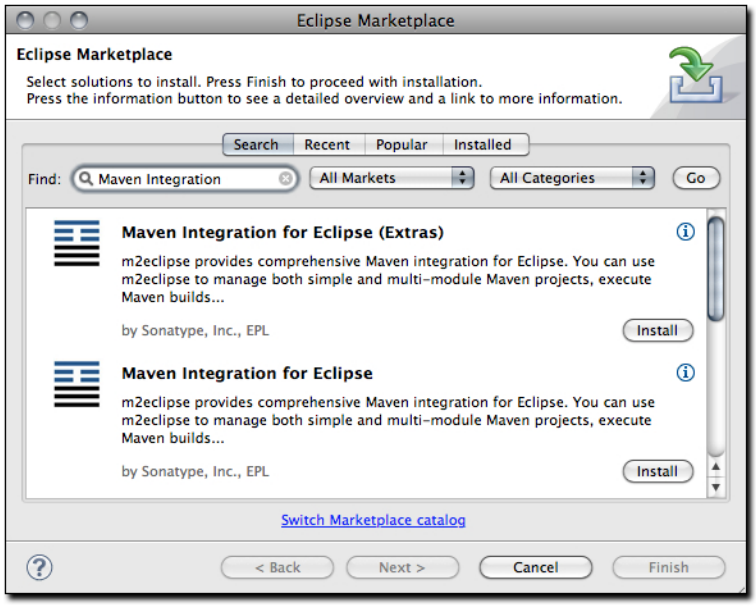

Figure 2.3. Selecting Maven Integration for Eclipse from Eclipse Marketplace

Once you click on Install, Eclipse will download a list of available components from the remote update site and present you with a list of available features in the Maven Integration for Eclipse plugin as shown in Figure 2.4, “Selecting the Core m2eclipse Feature for Installation”. Select the single, required component named "Maven Integration for Eclipse (Required)", and click on the Finish button.

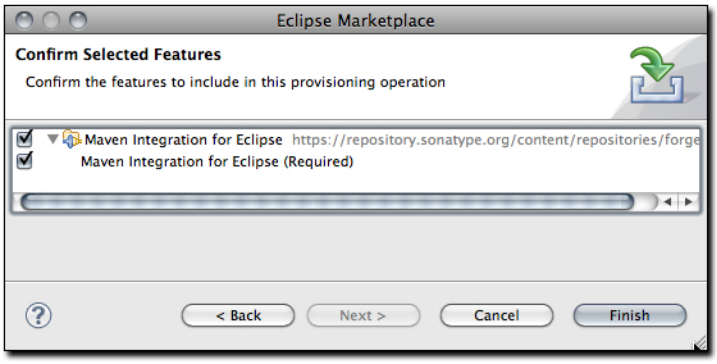

Figure 2.4. Selecting the Core m2eclipse Feature for Installation

Eclipse will then ask you to agree to the licenses for Maven Integration for Eclipse in the Review Licenses step shown in Figure 2.5, “Agreeing to Software License During m2eclipse Installation”. Maven Integration for Eclipse is distributed under the Eclipse Public License version 1.0. If you agree with the conditions of this license, select "I accept the terms of the license agreement" and click on the Finish button.

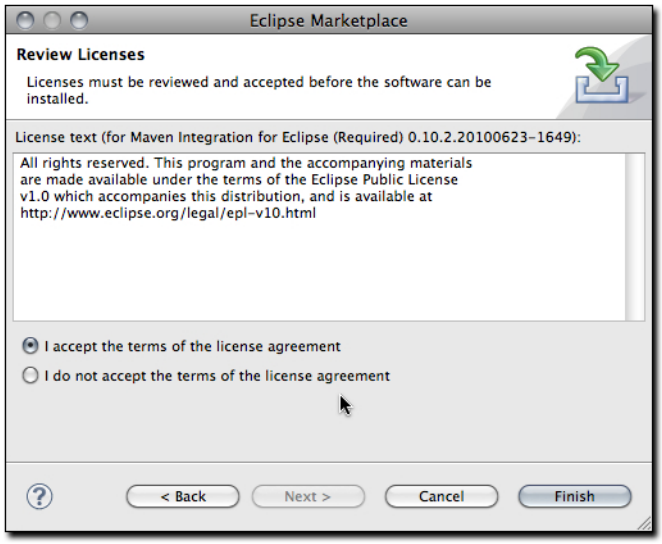

Figure 2.5. Agreeing to Software License During m2eclipse Installation

During the installation process, Eclipse may warn you that the software you are installing contains "unsigned content". If you see the dialog shown in Figure 2.6, “Ignoring Warning During m2eclipse Installation”, click on OK to continue the installation process.

Figure 2.6. Ignoring Warning During m2eclipse Installation

Once m2eclipse has been installed, Eclipse will prompt you to either restart or apply changes to a running Eclipse. At this stage of the installation, you should click on Restart Now to restart your Eclipse instance. After a successful restart, Maven Integration for Eclipse will be installed.

Figure 2.7. Restarting Eclipse after m2eclipse Installation

2.2.2. Installing Maven Integration for Eclipse (Extras) Prerequisites

The Maven Integration for Eclipse extra components provide support for additional tools like the Web Tools Project (WTP), Subversion integration via Subclipse, and integration with Mylyn. The following sections provide guidance for users installing some of the prerequisites for m2eclipse extra components.

2.2.2.1. Installing Subclipse in the Eclipse Marketplace

To install Subclipse in the Eclipse Marketplace, open the Marketplace, select the Search tab, and search for "Subclipse" to see the dialog shown in Figure 2.8, “Selecting Subclipse from the Eclipse Marketplace”. Click on Install next to the Subclipse search result item.

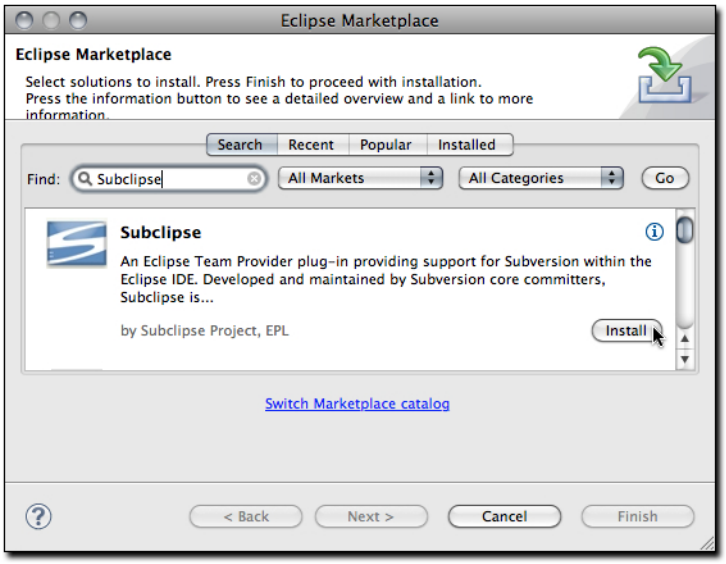

Figure 2.8. Selecting Subclipse from the Eclipse Marketplace

Select the appropriate Subclipse components, agree to the software license for Subclipse, and restart your Eclipse installation after the installation process is completed.

2.2.2.2. Installing Mylyn in the Eclipse Marketplace

To install Mylyn in the Eclipse Marketplace, open the Marketplace, select the Search tab, and search for "Mylyn" to see the dialog shown in Figure 2.9, “Selecting Mylyn from the Eclipse Marketplace”. Click on Install next to the Mylyn search result item.

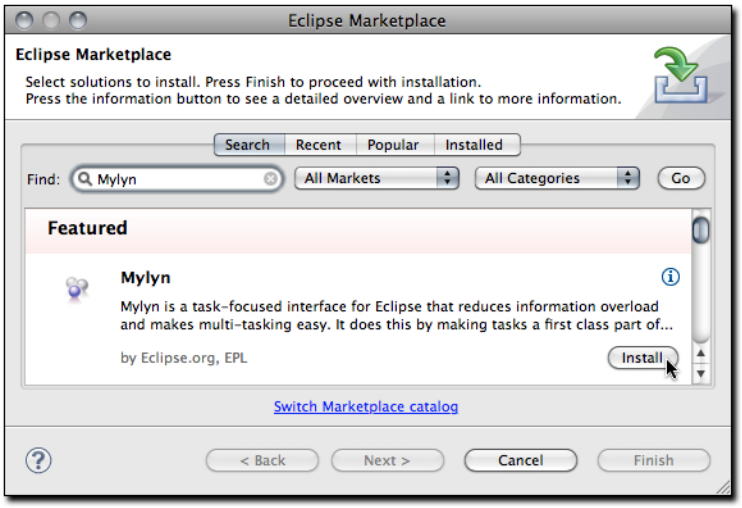

Figure 2.9. Selecting Mylyn from the Eclipse Marketplace

Select the appropriate SMylyn components, agree to the software license for Mylyn, and restart you Eclipse installation after the installation process is completed.

2.2.3. Installing Maven Integration for Eclipse (Extras)

To install the extra components for Maven Integration for Eclipse, open the Eclipse Marketplace, select the Search tab, and search for Maven Integration to see the search results shown in Figure 2.10, “Selecting Maven Integration for Eclipse (Extras) from the Eclipse Marketplace”. CLick on Install next to the "Maven Integration for Eclipse (Extras)" search result item.

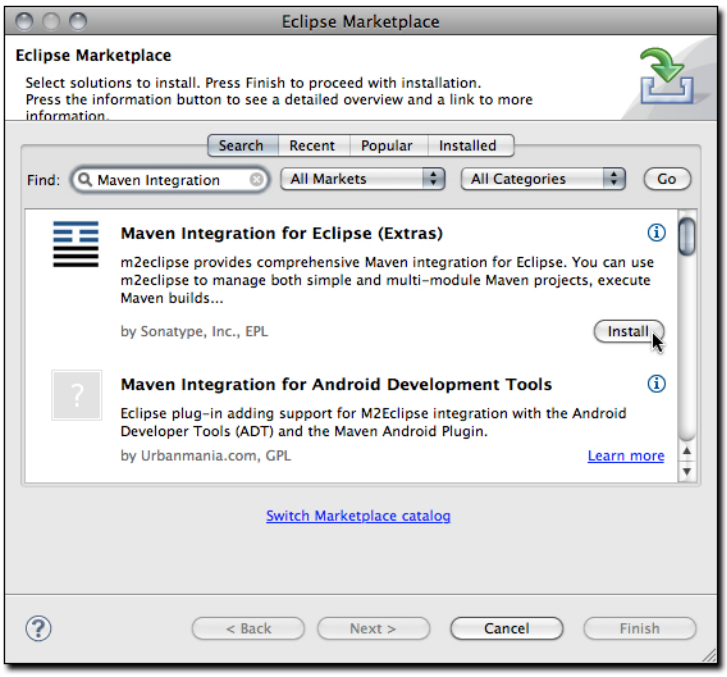

Figure 2.10. Selecting Maven Integration for Eclipse (Extras) from the Eclipse Marketplace

Once you have clicked on Install, Eclipse will download the list of available plugin components from the remote update site. The list of available components will then be displayed in the Confirm Select Features step as shown in Figure 2.11, “Selecting m2eclipse (Extras) Components to Install”. The components available in the Maven Integration for Eclipse (Extras) site are:

M2Eclipse Extensions Development Suppport (Optional)
: Install this component if you want to develop custom pages in the POM Editor, create custom actions in the Maven popup menu, provide custom templates in the POM XML editor, or make other customizations to m2eclipse.

Maven Integration for WTP (Optional)
: If you are developing applications using the Eclipse Web Tools Project, this component will adapt the Maven classpath container and other Maven resources to the WTP standards.

Maven issue tracking configurator for Mylyn 3.x (Optional)
: Maven can be configured to work with Mylyn a comprehensive issue and time tracking plugin which currently ships with the Eclipse IDE.

Maven SCM handler for Subclipse (Optional)
: Subclipse is a popular Subversion plugin for Eclipse hosted by the Tigris community. If you want to use Maven with Subclipse, this component will provide the necessary integration between the two plugins.

Maven SCM handler for Team/CVS (Optional)
: This component provides integration between the m2eclipse plugin and the CVS support built into Eclipse.

Maven SCM Integration (Optional)
: This component is required if you installed one of the previous plugins (Subclipse or Team/CVS)

Project configurators for commonly used maven plugins (temporary)
: This is a temporary project which contains project configurators for commonly used Maven Plugins.

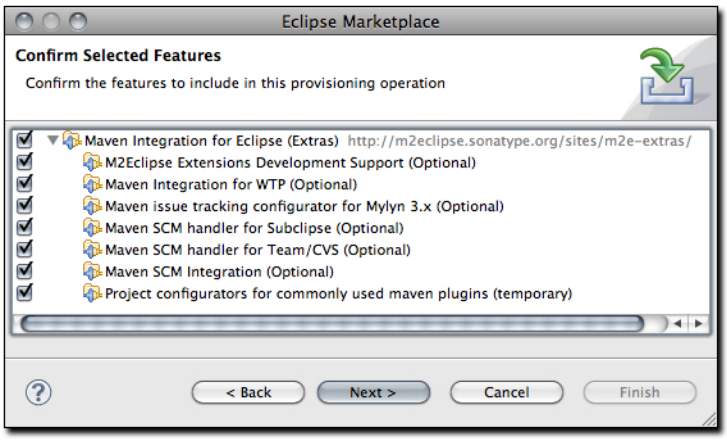

Figure 2.11. Selecting m2eclipse (Extras) Components to Install

Once you have selected the features you wish to install, click Next. Clicking Next will bring you to the Review Licenses step as shown in Figure 2.12, “Agreeing to Software License During m2eclipse (Extras) Installation”. The m2eclipse Extras is distributed under the Eclipse Public License version 1.0. If you agree to this open source license, select "I accept the terms of the license agreement" and click on "Finish".

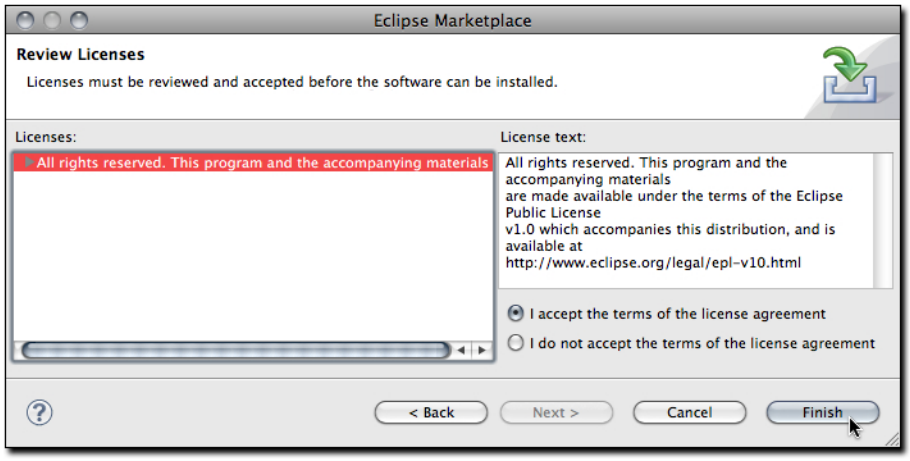

Figure 2.12. Agreeing to Software License During m2eclipse (Extras) Installation

During the installation process for the m2eclipse Extras, you may receive some warning that "you are installing software that contains unsigned content". If you see the dialog shown in Figure 2.13, “Ignoring Warning During m2eclipse (Extras) Installation”, click OK to continue the installation process.

Figure 2.13. Ignoring Warning During m2eclipse (Extras) Installation

Once the installation process is finished, Eclipse will prompt you to apply the changes or restart your Eclipse instance with a dialog shown in Figure 2.14, “Restarting Eclipse after m2eclipse (Extras) Installation”. To complete the installation of m2eclipse (Extras), restart your Eclipse instance.

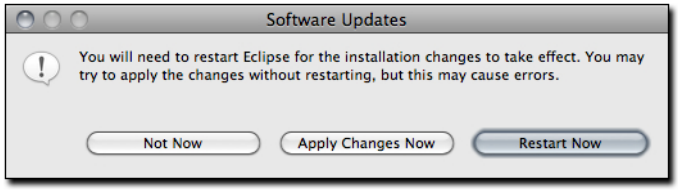

Figure 2.14. Restarting Eclipse after m2eclipse (Extras) Installation

2.3. Installing m2eclipse in Eclipse 3.5 (Gallileo)

2.3.1. Installing m2eclipse Core Components

To install m2eclipse, use the following Eclipse update site to install the core of the m2eclipse plugin. This Core update site contains a single component: "Maven Integration for Eclipse (Required)". When you install this component you will be installing all of the core Wizards, the POM Editor, Maven Repository integration, and Maven integration:

- m2eclipse Plugin: http://m2eclipse.sonatype.org/sites/m2e

To install this plugin in the Eclipse IDE:

1. Select Help > Install New Software. This should display the "Install" dialog.

2. Paste the Update Site URL into the field named "Work with:" and press Enter. Pressing Enter should cause Eclipse to update list of available plugins and components.

3. Choose the component listed under m2eclipse: "Maven Integration for Eclipse (Required)".

4. Click Next. Eclipse will then check to see if there are any issues which would prevent a successful installation.

5. Click Next and agree to the terms of the Eclipse Public License v1.0.

6. Click Finish to begin the installation process.

Eclipse will then download and install the necessary components. Once the installation process is finished, Eclipse will ask you if you want to restart the IDE. Sonatype strongly recommends that you restart your IDE after installing m2eclipse.

> ### Warning
>
> You cannot upgrade from m2eclipse 0.9 to m2eclipse 0.10. If you are running m2eclipse 0.9.8 or 0.9.9 you must either uninstall m2eclipse from your Eclipse installation or start with a fresh installation of Eclipse.

If you've installed the plugin successfully, you should see a Maven option in the list of preference categories when you go to Window, Preferences....

2.3.2. Installing m2eclipse Extras

In addition to the core m2eclipse components, the following optional components are available from a separate update site. If you plan to use m2eclipse to materialize projects from Subversion or CVS, integrate Maven with the Eclipse Web Tools Project (WTP), or use the m2eclipse Mylyn integration you will need to install the following, extra components:

- Maven SCM Integration
- Maven SCM handler for Team/CVS
- Maven SCM handler for Subclipse
- Maven issue tracking configurator for Mylyn 3.x
- Maven Integration for WTP
- M2Eclipse Extensions Development Support

To install optional m2eclipse components, use the m2eclipse Extras update site. This update site contains the following m2eclipse components:

- m2eclipse Extras Update Site: http://m2eclipse.sonatype.org/sites/m2e-extras

2.3.3. Installing Optional Prerequisites

Several of the extra components listed in the previous section require third-party plugins to be installed prior to installation. You can install these prerequisites when you install m2eclipse, just add a new remote update site to Eclipse for each of the prerequisite components.

To install these prerequisites, select Help, Install New Software... which will load the "Software Updates and Add-ons" dialog. In this dialog, choose the Available Software panel and click on Add Site... which will load the simple "Add Site" dialog. Enter the URL of the update site you wish to add and click OK. In the "Software Updates and Add-ons" dialog, the available plugins from an update site will appear as soon as the site is added. You can then select the modules you want to install and click the Install... button. Eclipse will then resolve all the dependencies for the selected plugins, and ask you to agree to the plugin license. After Eclipse installs new plugins, you should restart the IDE.

2.3.3.1. Installing Subclipse

When you install Subclipse, you will need to make a decision about Subversion compatibility. If you are using Subversion 1.5.x client features, you will need to install Subclipse version 1.4. If you are using Subversion 1.6.x client features, you will need to install Subclipse version 1.6.

To install Subclipse, use one of the Eclipse plugin update sites listed below.

- Subclipse 1.4 (for Subversion 1.5 compatibility): http://subclipse.tigris.org/update_1.4.x
- Subclipse 1.6 (for Subversion 1.6 compatibility): http://subclipse.tigris.org/update_1.6.x

For other versions of Subclipse, and for more information about the Subclipse plugin, please see the Subclipse project's web site at http://subclipse.tigris.org/.

2.3.3.2. Installing Mylyn

To install JIRA or Trac integration with Mylyn, add the Mylyn extras Eclipse update URL, you'll want to do this if your organization uses Atlassian's JIRA[2] for issue tracking. To install Mylyn use the following update sites:

- Mylyn (Eclipse 3.4, 3.5, and 3.6M4): http://download.eclipse.org/tools/mylyn/update/e3.4
- Mylyn Extras (JIRA and Trac Support): http://download.eclipse.org/tools/mylyn/update/extras

For more information about the Mylyn project, see the Mylyn project's web site at http://www.eclipse.org/mylyn/.

2.3.3.3. Installing the Web Tools Platform (WTP)

To install the Web Tools Platform (WTP). Install the "Eclipse IDE for Java EE Developers" from http://www.eclipse.org/downloads/, or download a WTP release from http://download.eclipse.org/webtools/downloads/.

For more information about the Web Tools Platform, see the Web Tools Platform project's web site at http://www.eclipse.org/webtools/.

2.4. Uninstalling m2eclipse from Eclipse 3.6 (Helios) with the Eclipse Marketplace

To uninstall m2eclipse and m2eclipse (Extras) from Eclipse 3.6 (Helios), open up the Eclipse Marketplace by selecting Eclipse Marketplace from the Eclipse Help menu. Once you have the Eclipse Marketplace dialog open, select the Installed tab as shown in Figure 2.15, “Selecting Maven Integration for Eclipse Components to Uninstall”. To uninstall either "Maven Integration for Eclipse" or "Maven Integration for Eclipse (Extras)", click on the "Uninstall" button next to each item.

[2] http://www.atlassian.com/software/jira/

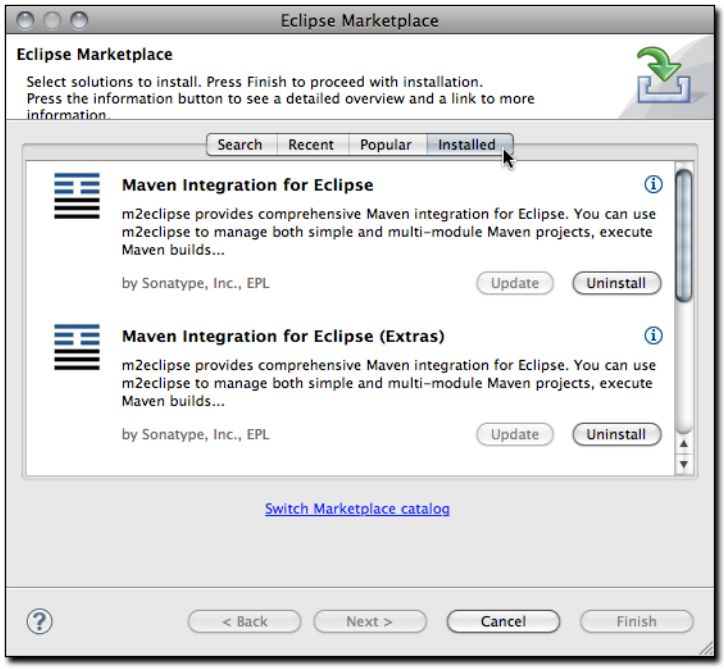

Figure 2.15. Selecting Maven Integration for Eclipse Components to Uninstall

If you are uninstalling "Maven Integration for Eclipse", the Eclipse IDE will prompt you to select the feature you wish to uninstall as shown in ???. Select the feature to uninstall, and click on Next to continue.

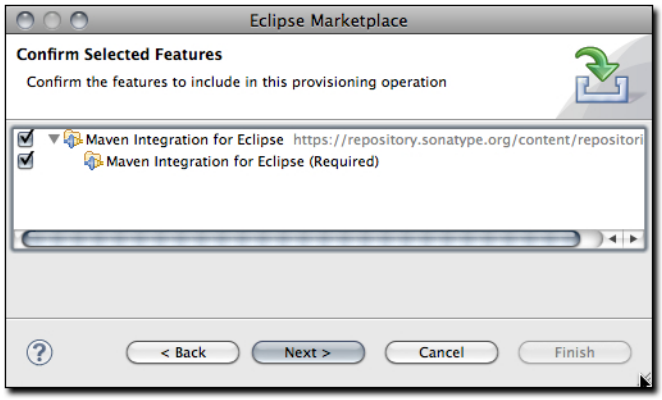

Figure 2.16. Selecting Components to Install for Maven Integration for Eclipse Installation

If you are uninstalling "Maven Integration for Eclipse (Extras)", the Eclipse IDE will prompt you to select the features you wish to uninstall as shown in Figure 2.17, “Selecting Maven Integration for Eclipse (Extras) Features to Uninstall”. Select the features to uninstall, and click on "Finish" or "Next".

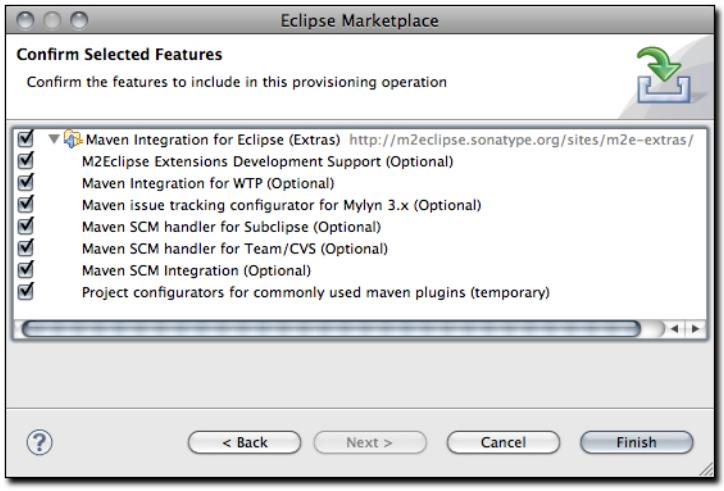

Figure 2.17. Selecting Maven Integration for Eclipse (Extras) Features to Uninstall

Once the uninstallation has successfully completed, Eclipse will prompt you to either apply changes or restart. To complete the uninstallation process, click on "Restart Now".

Figure 2.18. Restarting Eclipse after Uinstallation

Chapter 3. Creating and Importing Projects

3.1. Creating a Maven Project

When using Maven, project creation takes place through the use of a Maven archetype. In Eclipse, project creation takes place via the new project wizard. The new project wizard inside of Eclipse offers a plethora of templates for creating new projects. The m2eclipse plugin improves upon this wizard to provide the following additional capabilities:

- Checking out a Maven project from a SCM repository
- Creating a Maven project using a Maven archetype
- Creating a Maven POM file

As shown in Figure 3.1, “Creating a New Project with m2eclipse Wizards”, all three of these options are important to developers using Maven. Let’s take a look at each one.

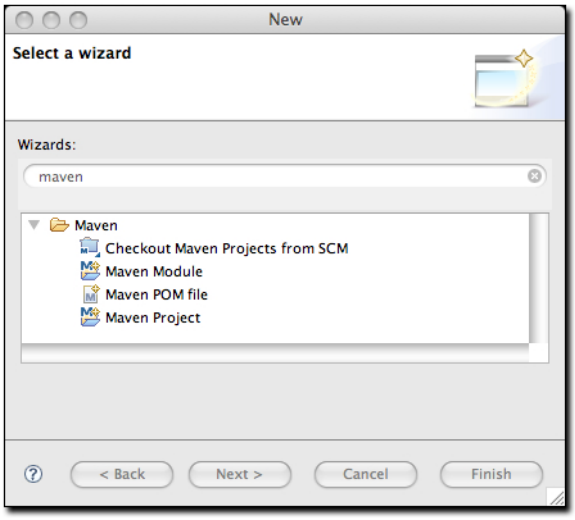

Figure 3.1. Creating a New Project with m2eclipse Wizards

3.1.1. Checking Out a Maven Project from SCM

m2eclipse provides the ability to check out a project directly from a SCM repository. Simply enter the SCM information for a project and it will check it out for you to a location of your choice as shown in Figure 3.2, “Checkout a New Project from Subversion”:

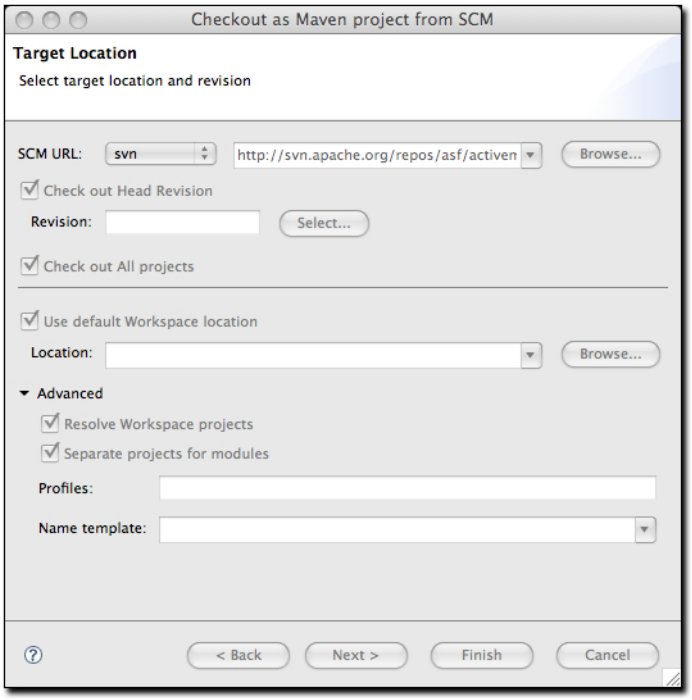

Figure 3.2. Checkout a New Project from Subversion

There are additional options in this dialog for specifying a particular revision by browsing the revisions in a Subversion repository or by simply entering the revision number manually. These features reuse of some of the features in the Subclipse plugin to interact with the Subversion repository. The m2eclipse plugin supports the following SCM providers:

- Bazaar
- Clearcase
- CVS
- git
- hg
- Perforce
- Starteam
- Subversion
- Synergy
- Visual SourceSafe

3.1.2. Creating a Maven Project from a Maven Archetype

m2eclipse offers the ability to create a Maven project using a Maven Archetype. There are many Maven Archetypes provided in the list that comes with m2eclipse as shown in Figure 3.3, “Creating a New Project with a Maven Archetype”.

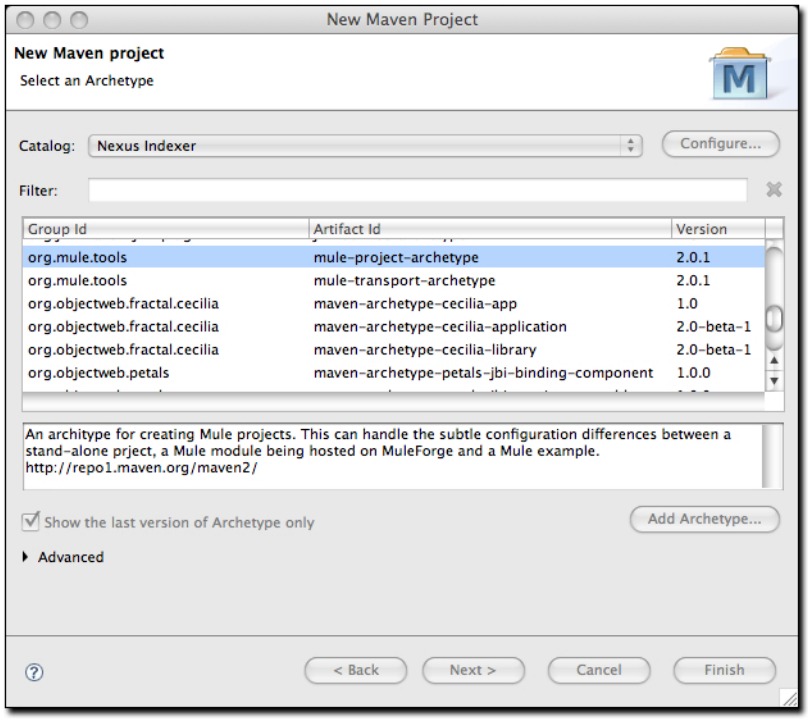

Figure 3.3. Creating a New Project with a Maven Archetype

The list of archetypes in Figure 3.3, “Creating a New Project with a Maven Archetype” is a list generated by something called the Nexus Indexer. Nexus is a repository manager which is introduced in "Repository Management with Nexus", a free book available from Sonatype which can be read online here: http://www.sonatype.com/books/nexus-book/reference/[1]. The Nexus indexer is a file which contains an index of the entire Maven repository, and m2eclipse uses it to list all of the available archetypes in the entire Maven repository. When this chapter was last updated, m2eclipse had approximately ninety archetypes in this Archetype dialog. Highlights of this list include:

- Standard Maven Archetypes to create
  - Maven Plugins
  - Simple Web Applications
  - Simple Projects
  - New Maven Archetypes
- Databinder[2] Archetypes (data-driven Wicket Applications) under `net.databinder`
- Apache Cocoon[3] Archetypes under `org.apache.cocoon`
- Apache Directory Server[4] Archetypes under `org.apache.directory.server`
- Apache Geronimo[5] Archetypes under `org.apache.geronimo.buildsupport`
- Apache MyFaces[6] Archetypes under `org.apache.myfaces.buildtools`
- Apache Tapestry[7] Archetypes under `org.apache.tapestry`
- Apache Wicket[8] Archetypes under `org.apache.wicket`

[1] ???

- AppFuse[9] Archetypes under `org.appfuse.archetypes`
- Codehaus Cargo[10] Archetypes under `org.codehaus.cargo`
- Codehaus Castor[11] Archetypes under `org.codehaus.castor`
- Groovy-based Maven Plugin[12] Archetypes (deprecated)[19] under `org.codehaus.mojo.groovy`
- Jini Archetypes
- Mule[13] Archetypes under `org.mule.tools`
- Objectweb Fractal[14] Archetypes under `org.objectweb.fractal`
- Objectweb Petals[15] Archetypes under `org.objectweb.petals`
- ops4j Archetypes under `org.ops4j`
- Parancoe[16] under `org.parancoe`
- slf4j Archetypes under `org.slf4j`
- Springframework[17] OSGI and Web Services Archetypes under `org.springframework`
- Trails Framework[18] Archetypes under `org.trailsframework`

[19]And these were just the archetypes that were listed under the Nexus Indexer Catalog, if you switch Catalogs you'll see other archetypes. While your results may vary, the following additional archetypes were available in the Internal Catalog:

- Atlassian Confluence[20] Plugin Archetype under `com.atlassian.maven.archetypes`
- Apache Struts[21] Archetypes under `org.apache.struts`
- Apache Shale Archetypes under `org.apache.shale`

A catalog is simply a reference to a repository index. You can manage the set of catalogs that the m2eclipse plugin knows about by clicking on the Configure... button next to the catalog drop down. If you have your own archetypes to add to this list, you can click on Add Archetype....

Once you choose an archetype, Maven will retrieve the appropriate artifact from the Maven repository and create a new Eclipse project with the selected archetype.

3.1.3. Creating a Maven Module

m2eclipse provides the ability to create a Maven module. Creating a Maven module is almost identical to creating a Maven project as it also creates a new Maven project using a Maven archetype. However, a Maven module is a subproject of another Maven project typically known as a parent project.

[19]Don't use the Groovy Maven Plugin in Codehaus' Mojo project. Jason Dillon has moved the Groovy Maven integration to the Groovy project in codehaus. For more information see http://groovy.codehaus.org/GMaven.

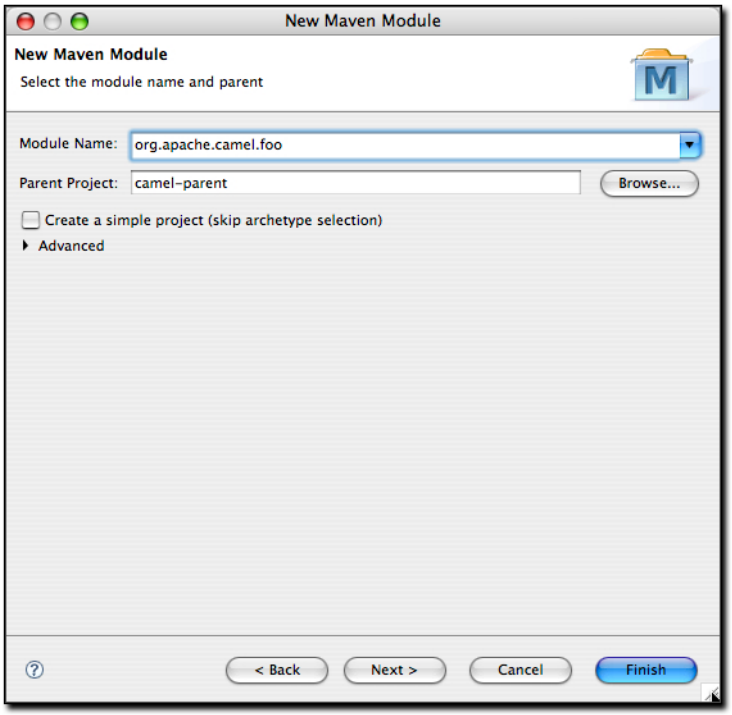

Figure 3.4. Creating a New Maven Module

When creating a new Maven module you must select a parent project that already exists inside of Eclipse. Clicking the browse button displays a list of projects that already exist as shown in Figure 3.5, “Selecting a Parent Project for a New Maven Module”:

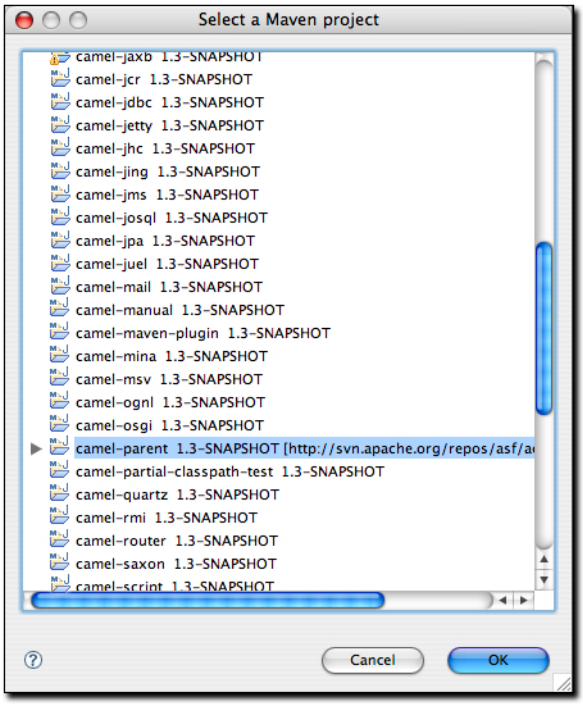

Figure 3.5. Selecting a Parent Project for a New Maven Module

After selecting a parent project from the list, you are returned to the New Maven Module window and the Parent Project field is populated as shown in Figure 3.4, “Creating a New Maven Module”. Clicking Next will then display the standard list of archetypes

from Section 3.1.2, “Creating a Maven Project from a Maven Archetype” so you can choose which one should be used to create the Maven module.

3.2. Create a Maven POM File

Another important feature m2eclipse offers is the ability to create a new Maven POM file. m2eclipse provides a wizard to easily create a new POM file inside of a project that is already in Eclipse. This POM creation wizard is shown in Figure 3.6, “Creating a New POM”:

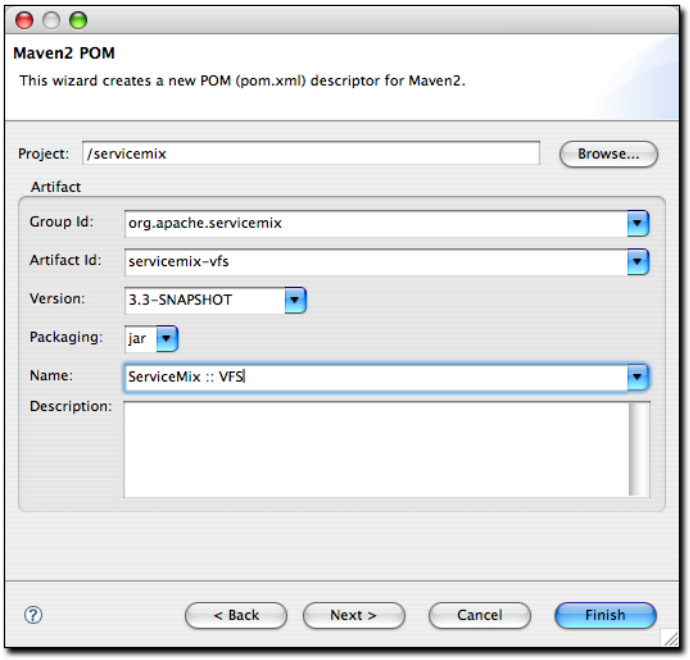

Figure 3.6. Creating a New POM

Creating a new Maven POM is just a matter of selecting a project, entering the Group Id, Artifact Id, Version, choosing the Packaging type, and providing a Name into the fields provided and m2eclipse. Click the Next button to start adding dependencies.

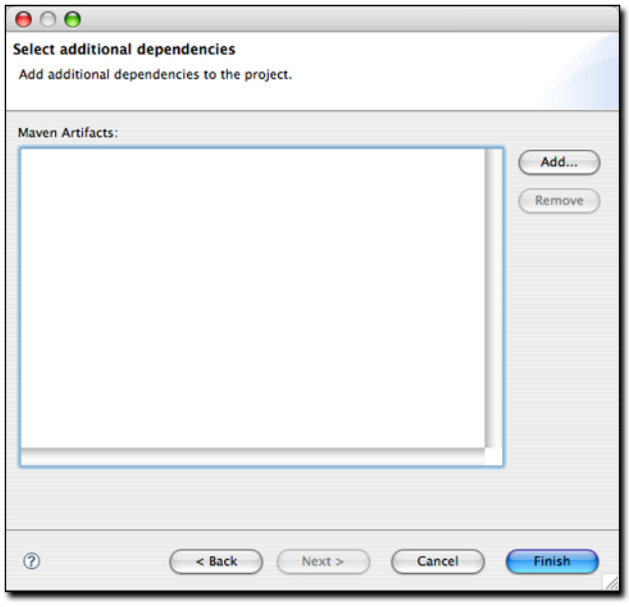

Figure 3.7. Adding Dependencies to a New POM

As you can see in Figure 3.7, “Adding Dependencies to a New POM” here are no dependencies in the POM yet. Just click the Add button to query the central Maven repository for dependencies as shown next in Figure 3.8, “Querying the Central Repository for Dependencies”:

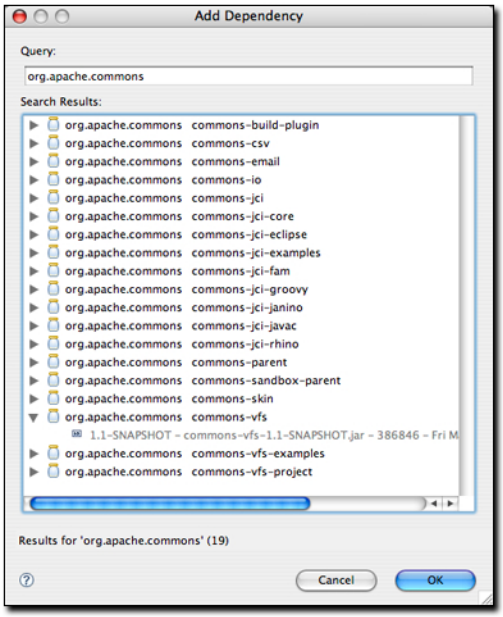

Figure 3.8. Querying the Central Repository for Dependencies

Querying for dependencies is as easy as entering the `groupId` for the artifact you need. Figure 3.8, “Querying the Central Repository for Dependencies” shows a query for `org.apache.commons` with `commons-vfs` expanded to see which versions are available. Highlighting the `1.1-SNAPSHOT` version of `commons-vfs` and clicking OK takes you back to the dependency selection where you can either query for more artifacts or just click finish to create the POM. When you search for dependencies, m2eclipse is making use of the same Nexus repository index that is used in the Nexus Repository Manager, a repository manager introduced in "Repository Management with Nexus" (http://www.sonatype.com/books/nexus-book/reference/).

Now that the you’ve seen the m2eclipse features for creating a new project, let’s look at a similar set of features for importing projects into Eclipse.

3.3. Importing Maven Projects

m2eclipse provides three options for importing a Maven project into Eclipse including:

- Import an existing Maven project
- Check out a Maven project from SCM
- Materialize a Maven project

Figure 3.9, “Importing a Maven Project” shows the wizard for importing projects with the options for Maven provided by m2eclipse:

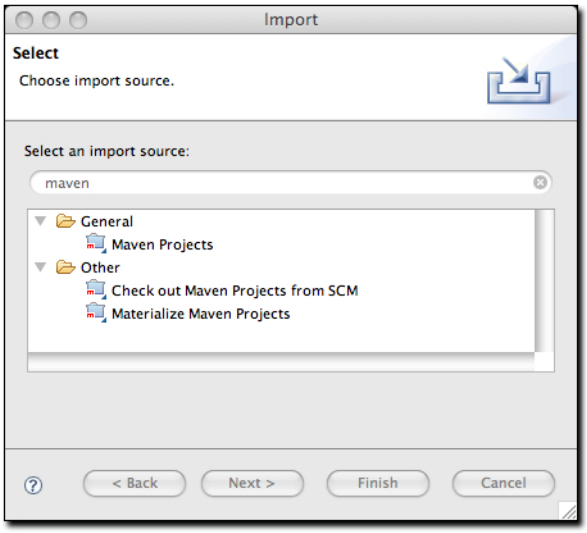

Figure 3.9. Importing a Maven Project

The dialog in Figure 3.9, “Importing a Maven Project” is displayed by using the File, Import command in Eclipse and then filtering the options by entering the word maven in the filter field. As noted above, there are three options available for importing a Maven project into Eclipse including: Maven Projects, Check out Maven Project from Subversion, and Materialize Maven Projects.

Importing a Maven project from Subversion is identical to the creation of a Maven project from Subversion as discussed in the previous section so discussion of it would be redundant. Let’s move on now to review the other two options for importing a Maven project into Eclipse.

3.3.1. Importing a Maven Project

m2eclipse can import a Maven project with an existing `pom.xml`. By pointing at the directory where a Maven project is located, m2eclipse detects all the Maven POMs in the project and provides a hierarchical list of them as shown in Figure 3.10, “Importing a Multi-module Maven Project”.

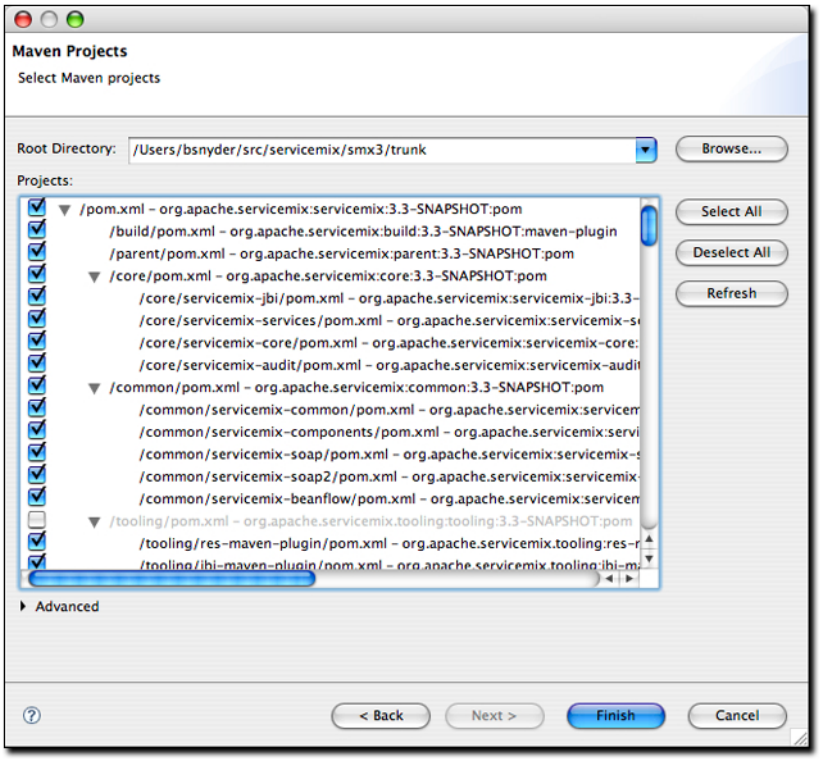

Figure 3.10. Importing a Multi-module Maven Project

Figure 3.10, “Importing a Multi-module Maven Project” displays the view of the project being imported. Notice that all the POMs from the project are listed in a hierarchy. This allows you to easily select which POMs (and therefore which projects) that you want to be imported into Eclipse. Once you select the project you would like to import, m2eclipse will import and build the project(s) using Maven.

3.3.2. Materializing a Maven Project

m2eclipse also offers the ability to "materialize" a Maven project. Materialization is similar to the process of checking out a Maven project from Subversion, but instead of manually entering the URL to the project’s Subversion repository, the Subversion URL is discovered from the project’s root POM file. You can use this feature to "materialize" projects from nothing more than a POM file if the POM file has the appropriate elements to specify the location of a source repository. Using this feature, you can browse the central Maven repository for projects, and materialize them into Eclipse projects. This comes in handy if your project depends on a third-party open source library, and you need to get your hands on the source code. Instead of tracking down the project web site and figuring out how to check it out of Subversion, just use the m2eclipse project to magically "materialize" the Eclipse project.

Figure 3.11, “Materializing a Maven Project” shows the wizard after choosing to materialize Maven projects:

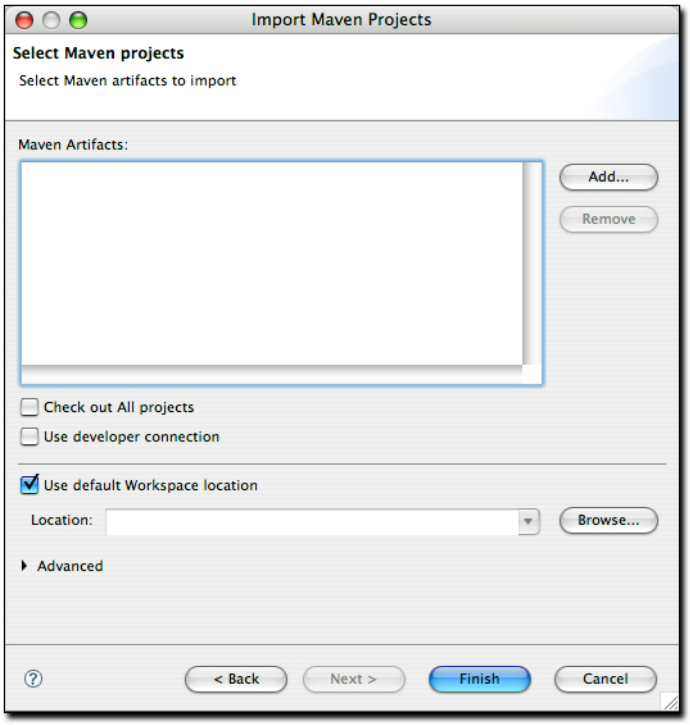

Figure 3.11. Materializing a Maven Project

Notice that the dialog box for Maven artifacts in Figure 3.11, “Materializing a Maven Project” is empty. This is because no projects have been added yet. In order to add a project, you must click the Add button on the right side and select a dependency to add from the central Maven repository. Figure 3.12, “Selecting Artifact to Materialize” shows how to add a project:

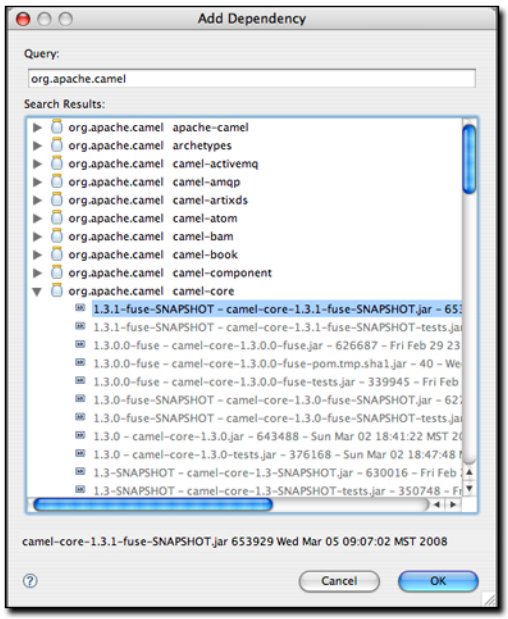

Figure 3.12. Selecting Artifact to Materialize

Upon entering a query, candidate dependencies will be located in the local Maven repository. After a few seconds of indexing the local Maven repository, the list of candidate dependencies appears. Select the dependency to add and click OK so that they are added to the list as shown in Figure 3.13, “Materializing Apache Camel”.

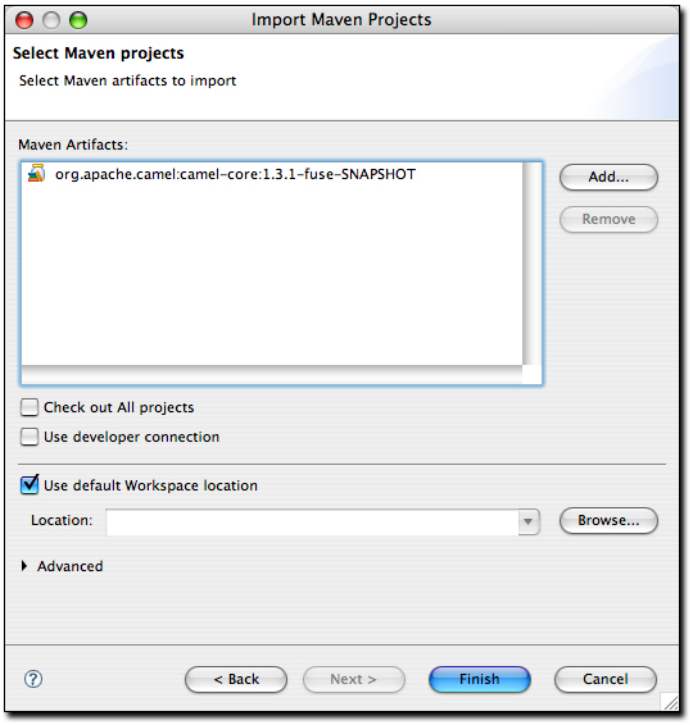

Figure 3.13. Materializing Apache Camel

Upon adding a dependency, you have the option of telling the m2eclipse plugin to check out all projects for the artifact.

Chapter 4. Running Maven Builds

4.1. Enabling the Maven Console

Before we begin to examine the features of m2eclipse, let's first enable the Maven console. Open the Console View by going to Window, Show View, Console. Then click on the little arrow on the right-hand side of the Open Console icon and select Maven Console as shown below:

Figure 4.1. Enabling the Maven Console in Eclipse

Maven Console shows the Maven output that normally appears on the console when running Maven from the command line. It is useful to be able to see what Maven is doing and to work with Maven debug output to diagnose issues.

4.2. Running Maven Builds

m2eclipse modified the Run As... and Debug As... menus to allow you to run a Maven build within Eclipse. Figure 4.2, "Running an Eclipse build with Run As.." shows the Run As... menu for an m2eclipse project. From this menu you can run one of the more common lifecycle phases like clean, install, or package. You can also load up the Run configuration dialog window and configure a Maven build with parameters and more options.

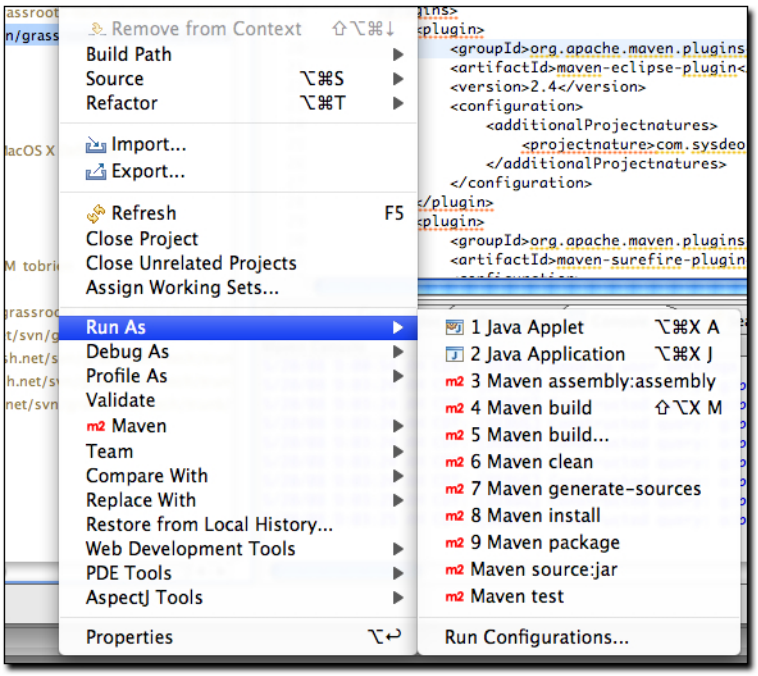

Figure 4.2. Running an Eclipse build with Run As..

If you need to configure a Maven build with more options, you can choose Run Configurations... and create a new Maven build. Figure 4.3, "Configuring a Maven Build as a Run Configuration" shows the Run dialog for configuring a Maven build.

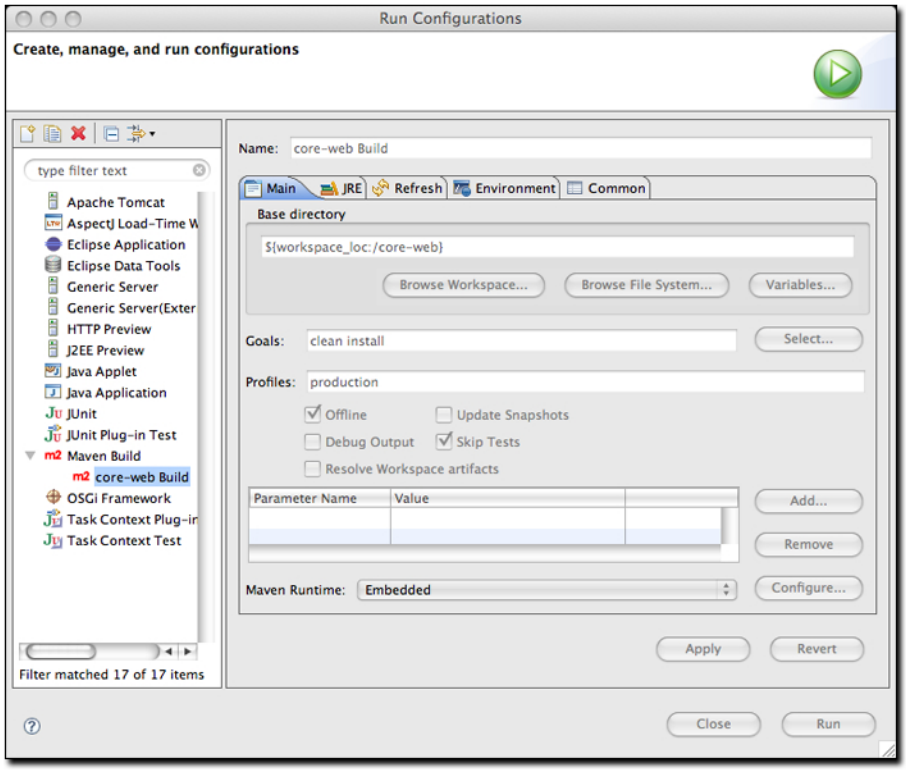

Figure 4.3. Configuring a Maven Build as a Run Configuration

The Run configuration dialog allows you to specify multiple goals and profiles, it exposes options like "skip tests" and "update snapshots", and allows you to customize everything from the project to the JRE to the environment variable. You can use this dialog to support any custom Maven build that you wish to launch with m2eclipse.

Chapter 5. m2eclipse Preferences

5.1. Maven Preferences

The ability to adjust the Maven preferences and some Maven options is an important aspect of developing with Maven and m2eclipse offers the ability to tweak these items via the Maven preferences page inside of Eclipse. Typically when using Maven on the command line, such preferences and options are available from files in your `~/.m2` directory and as command line options. m2eclipse provides access to some of the most important preferences and options from the Eclipse IDE. Figure 5.1, “Maven Preferences for Eclipse” shows the Maven preferences page in Eclipse:

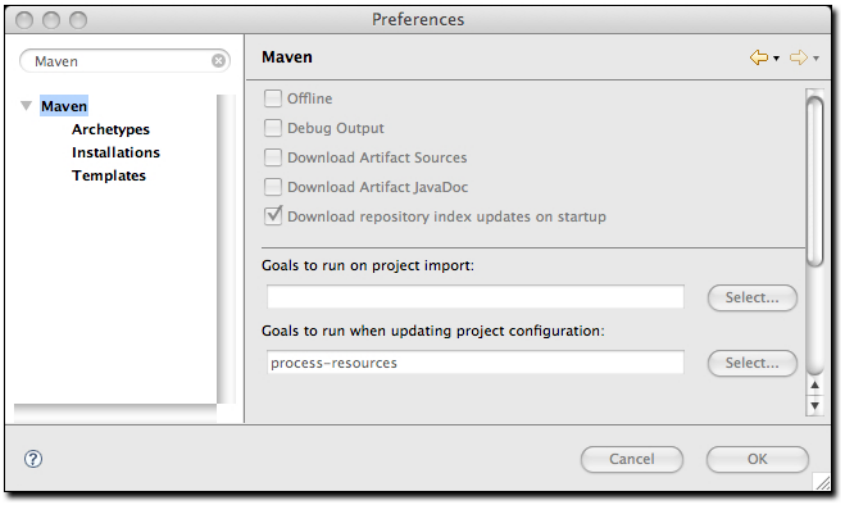

Figure 5.1. Maven Preferences for Eclipse

The check boxes in the top section provide the ability to:

- Run Maven in Offline mode, disabling any downloads from remote repositories
- Enable Debug output in the Maven Console
- Download Source jars for artifacts from remote Maven repositories
- Download JavaDoc jars for artifacts from remote Maven repositories
- Download and Update local indexes for remote repositories on startup

The next section offers a pop-up menu to select which goal you’d like to be executed when a project is imported and when the source folders for a given project are updated. The default goal is named `process-resources` which copies and process the resources for the project into the destination directory to make the project ready for packaging. Customizing this list of goals can come in handy if you need to run any custom goals which process resources or generate supporting configuration.

If you need help selecting a goal, click the Select... button to see the "Goals" dialog. The dialog on the left-hand side ofFigure 5.2, “Maven Goal Dialogs” shows the Goals dialog with a list of all the phases in the default Maven lifecycle.

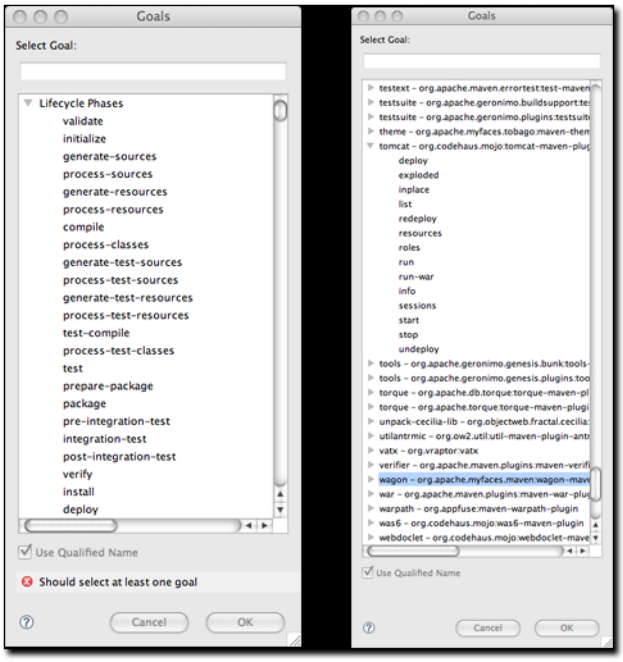

Figure 5.2. Maven Goal Dialogs

When you see the Goals dialog for the first time, there's a chance you might be overwhelmed by the number of goals it lists. There are literally hundreds of Maven plugins for everything from generating a database, to running integration tests, to performing static analysis, to generating web services with XFire. There are over two hundred plugins with selectable goals in the Goals dialog, the dialog on the right-hand side ofFigure 5.2, “Maven Goal Dialogs” shows the "Goals" dialog with the Tomcat Maven plugin's goals highlighted. You can always narrow the list of goals shown in this dialog by typing in some text to the search dialog, as you type in text, m2eclipse is going to narrow the list of available goals to goals which contain the text in the search field.

Another Maven preference page is the Maven Installations configuration page shown in Figure 5.3, “Maven Installations Preference Page”:

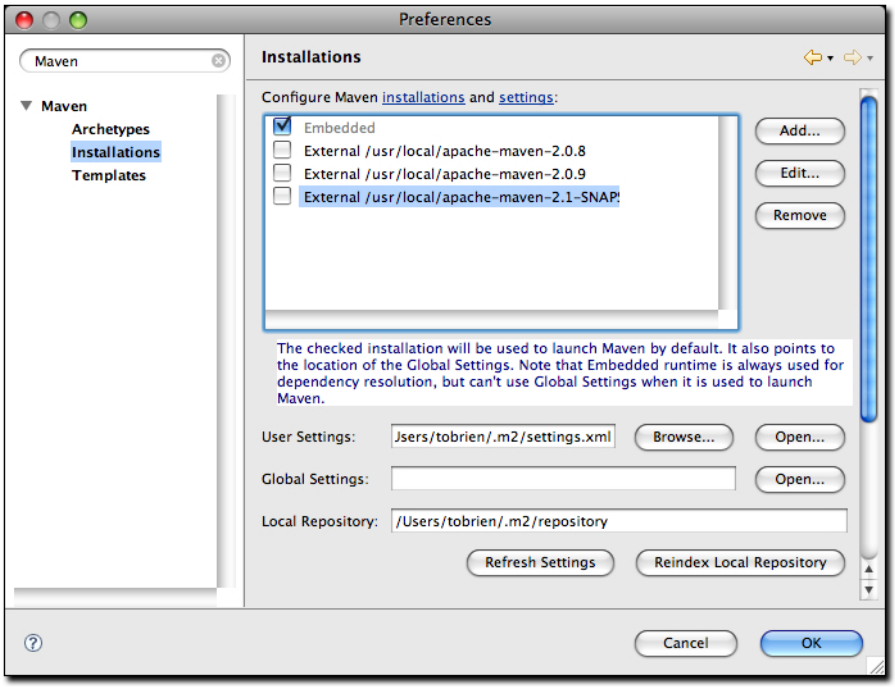

Figure 5.3. Maven Installations Preference Page

This page allows you to add other Maven installations to the Eclipse environment. If you want to use a different version of Maven with the m2eclipse plugin you can configure multiple installations of Maven from this configuration page, this is very similar to the ability to add more than one Java Virtual Machine to be run inside of Eclipse. An embedded version of the Maven known as the Maven Embedder is already specified. This is what is used to execute Maven inside of Eclipse. If you have another installation of Maven which you would like to use instead of the Maven Embedder, you can add another Maven runtime by clicking on the Add.. button. Figure 5.3, "Maven Installations Preference Page" shows a configuration page that lists the Maven Embedder, Maven 2.0.9, and an installation of Maven 2.1-SNAPSHOT.

The Installations configuration page also allows you to specify the location of the global Maven settings file. If you do not specify the location of this file on this configuration page, Maven will use the default global settings file found in `conf/settings.xml` of the selected Maven installation. You can also customize the location of your user settings file from the default location of `~/.m2/settings.xml`, and you can customize the location of your local Maven repository from the default location of `~/.m2/repository`.

Also available in the Eclipse preferences is the ability to enable a decorator named the Maven Version Decorator. This preference provides a given project's current version on the Eclipse Package Explorer and is shown in Figure 5.4, "Enabling the Maven Version Decorator".

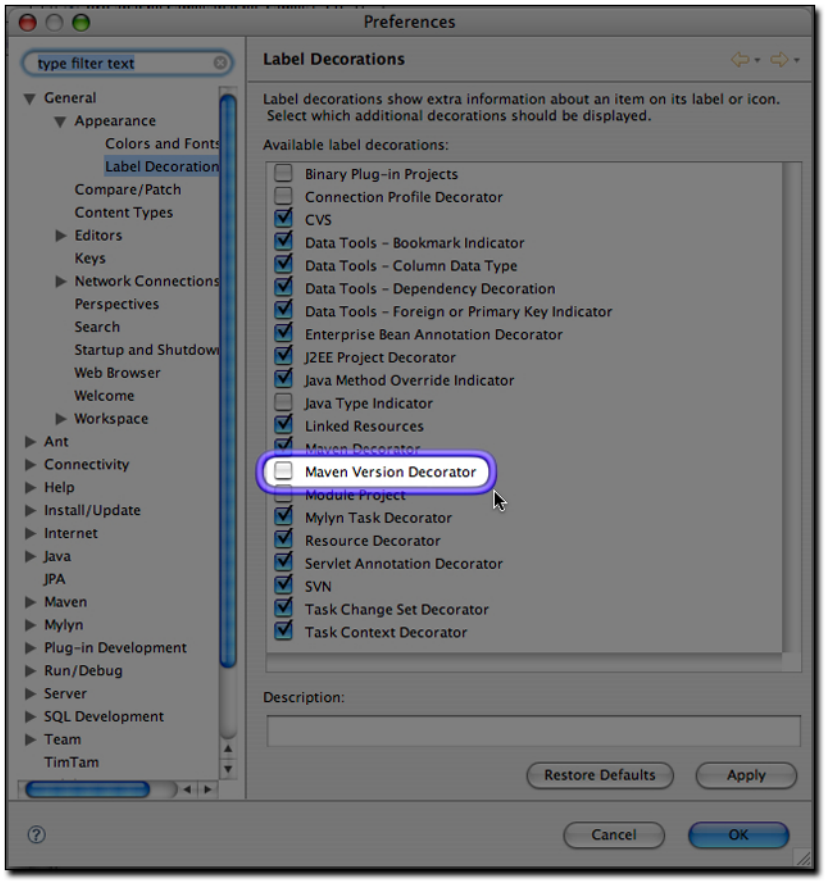

Figure 5.4. Enabling the Maven Version Decorator

To enable this preference, simply check the Maven Version Decorator option that is highlighted in Figure 5.4, "Enabling the Maven Version Decorator". If the Maven Version Decorator is not enabled, a project will only list it's name and relative path in the Package Explorer as shown in Figure 5.5, "Package Explorer without Maven Version Decorator":

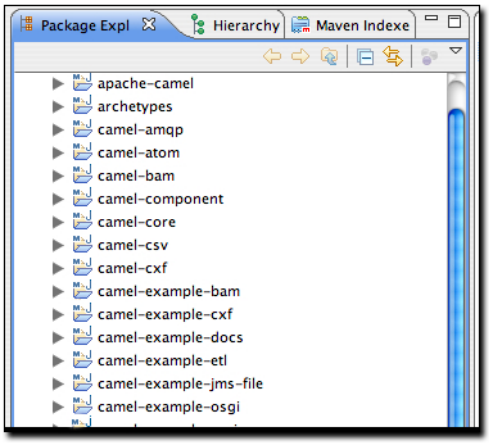

Figure 5.5. Package Explorer without Maven Version Decorator

Upon enabling the Maven Version Decorator, the project name will include the current project version as shown in Figure 5.6, “Package Explorer with Maven Version Decorator Enabled”:

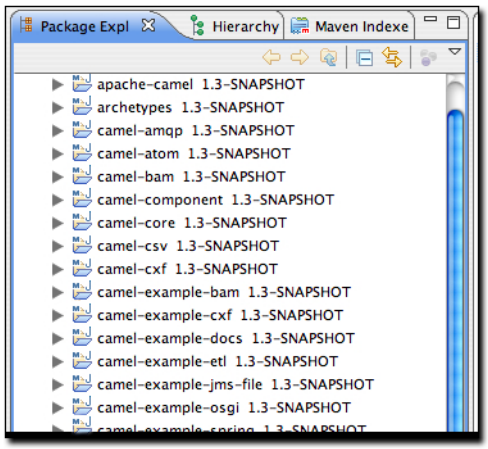

Figure 5.6. Package Explorer with Maven Version Decorator Enabled

This is a helpful feature that provides the project version at a glance instead of being required to open the POM to locate the version element.

Chapter 6. Working with Maven Repositories

6.1. Working with Maven Repositories

m2eclipse also provides some tools to make working with Maven repositories a bit easier. These tools provide functionality for:

- Searching for Artifacts
- Searching for Java classes
- Indexing Maven repositories

6.2. Searching For Maven Artifacts and Java classes

m2eclipse adds a couple of items to the Eclipse Navigation menu that make searching for Maven Artifacts and Java classes easy work. Each option is available by clicking on the Navigate menu as shown in Figure 6.1, “Searching for Artifacts and Classes”:

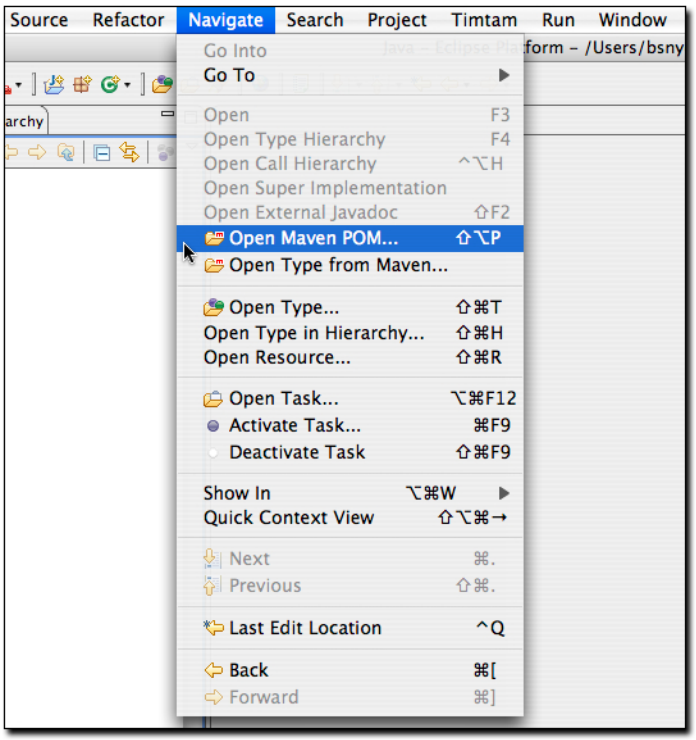

Figure 6.1. Searching for Artifacts and Classes

Notice the available options in Figure 6.1, “Searching for Artifacts and Classes” under the Eclipse Navigate menu named Open Maven POM and Open Type from Maven. The Open Maven POM option allows you to search the Maven repository for a given POM as shown in Figure 6.2, “Searching for a POM”:

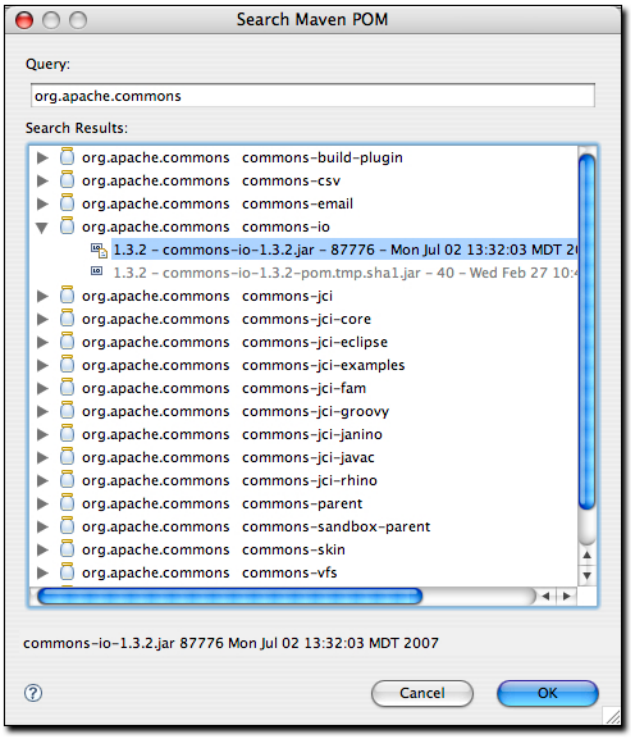

Figure 6.2. Searching for a POM

Upon selecting an artifact and clicking OK, the POM for that artifact is opened in Eclipse for browsing or editing. This is handy when you need to take a quick look at the POM for a given artifact.

The second m2eclipse option in the Navigate menu is named Open Type from Maven. This feature allows you to search for a Java class by name in a remote repository. Upon opening this dialog, simply type ‘factorybean’ and you’ll see many classes with the name `FactoryBean` in them as shown in Figure 6.3, “Searching the Repository for a Class”:

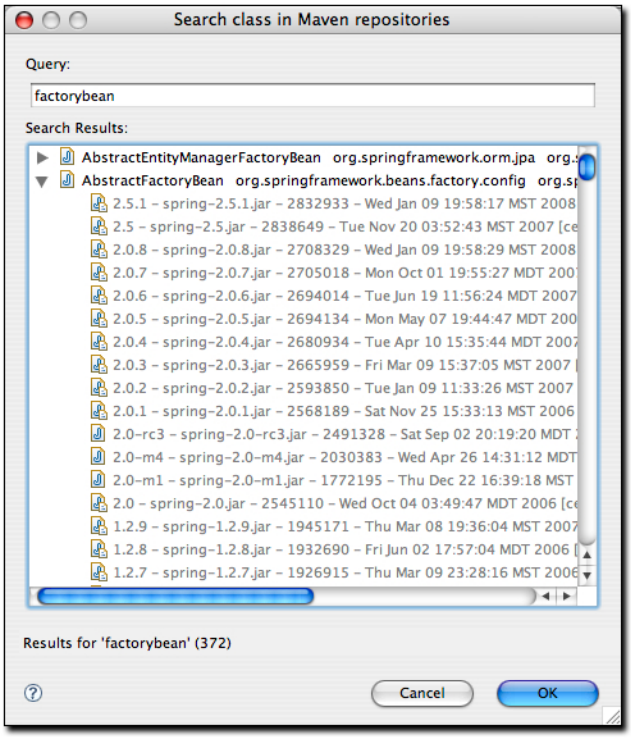

Figure 6.3. Searching the Repository for a Class

This is a big time saving feature because it means that manually searching through artifacts in a Maven repository for a particular class is a thing of the past. If you need to use a specific class, just fire up Eclipse, go to the Navigate menu and search for the class. m2eclipse will show you the list of artifacts in which it appears.

6.3. Indexing Maven Repositories

The Maven Indexes View allows you to manually navigate to POMs in a remote repository and open them in Eclipse. To see this View, go to View, Show View, Other, type the word "maven" into the search box and you should see a view named Maven Indexes as shown in Figure 6.4, “Show Maven Indexes View”:

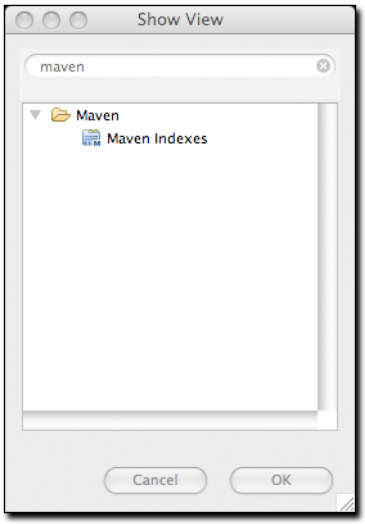

Figure 6.4. Show Maven Indexes View

Select this View and click OK. This will show the Maven Indexes View as shown in Figure 6.5, “Maven Indexes View”:

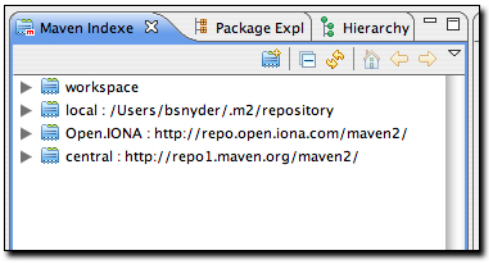

Figure 6.5. Maven Indexes View

Additionally, Figure 6.6, “Locating a POM from the Indexes View” shows the Maven Indexes View after manually navigating to locate a POM:

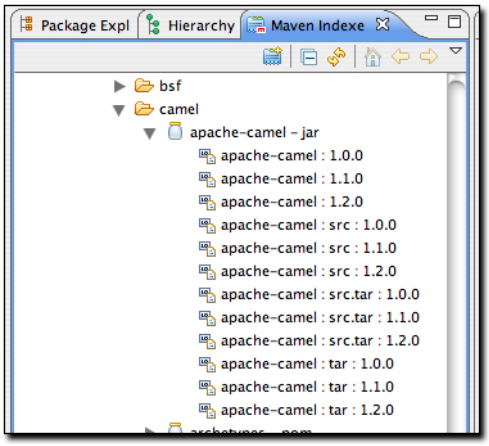

Figure 6.6. Locating a POM from the Indexes View

After finding the `apache-camel` artifact, double-clicking on it will open it up in Eclipse for browsing or editing.

These features make working with remote repositories from inside of Eclipse so much easier and faster. After all the hours you may have spent doing these types of tasks by manually over the last few years - visiting repositories through a web browser, downloading artifacts and grepping through them for classes and POMs - you'll find that m2eclipse is a welcome change for the better.

6.4. Browsing and Manipulating Maven Repositories

The m2eclipse plugin allows you to browse and manipulate repository indexes. Using the Maven Repository view in m2eclipse you can:

- Browse your Local Maven repository
- Browse global repositories such as the Central Maven repository
- Browse a repository which captures artifacts generated by Maven projects in your Eclipse workspace
- Rebuild a Nexus Index from scratch
- Update a Nexus Index with incremental changes

- Modify the scope of repository indexing with a "minimal" or "full" index
- Disable Indexing for a repository
- Materialize a Maven project from information stored in a POM

6.4.1. Opening the Maven Repository View

To browse Maven repositories and to manipulate repository indexes open the Maven Repositories view by selecting Windows, Show View, Other... as shown in Figure 6.7, “Opening a View in Eclipse”.

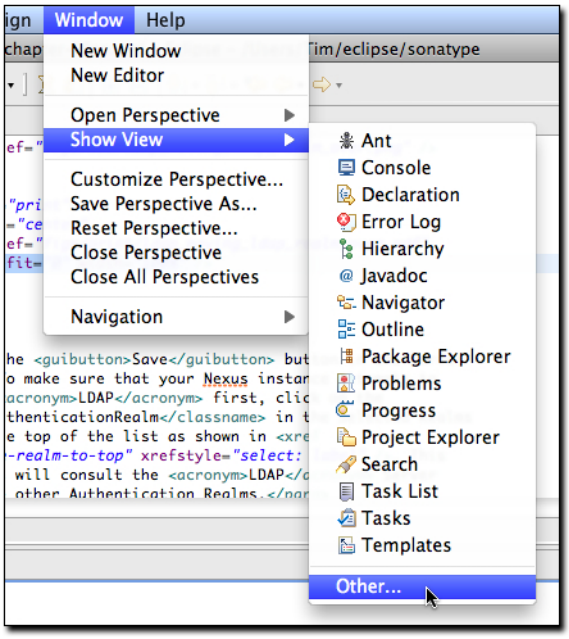

Figure 6.7. Opening a View in Eclipse

Once you select Other... Eclipse will display a dialog containing all available views. Select the Maven Repositories view under the Maven folder in the Show View dialog as shown in Figure 6.8, “Selecting the Maven Repositories View in the Show View Dialog”.

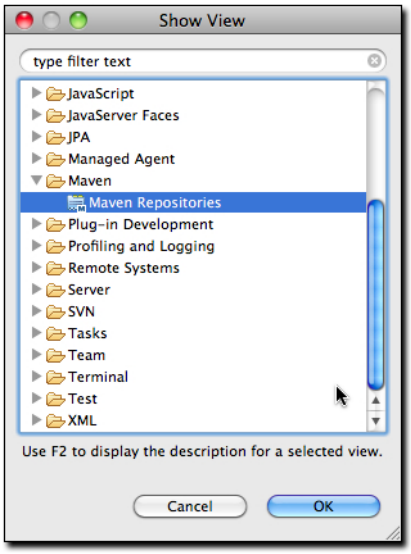

Figure 6.8. Selecting the Maven Repositories View in the Show View Dialog

Once you have selected Maven Repositories and clicked on the OK button, Eclipse will then load the Maven Repositories view. This view contains three folders:

Local Repositories
: This folder contains your local Maven repository which is stored in ~/.m2/repository by default. It also contains a repository that represents the Maven projects contained in your Eclipse workspace.

Global Repositories
: This folder contains any global Maven repositories that are referenced by all Maven projects. This folder contains the Central Maven repository under the repository identifier of "central". It will also contain mirrors that have been configured in your Maven Settings (`~/.m2/settings.xml`).

Project Repositories
: This folder contains repositories which are defined by your projects. These repositories are present either in your project's `pom.xml` file or in an active Maven Profile.

6.4.2. Browsing Global Repositories

If you have been using Maven, you are familiar with the Central Maven repository. This is default repository from which Maven will retrieve dependencies and other artifacts needed during a build. If you expand the central repository, you will be able to browse the contents of the repository and double click on specific artifacts. Double-clicking on one of the artifacts shown in Figure 6.9, “Browsing a Global Repository” will load that artifact's POM in the Form-based POM Editor.

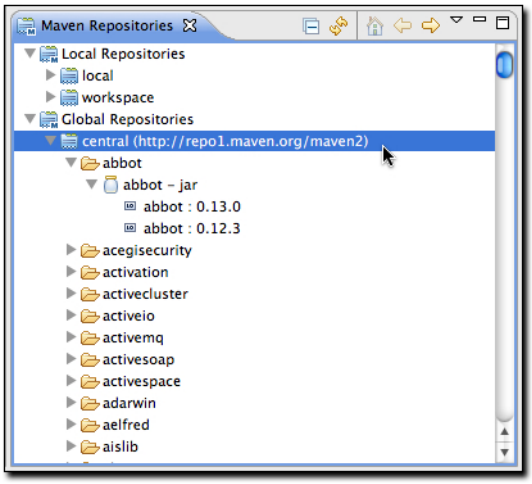

Figure 6.9. Browsing a Global Repository

In addition to loading an artifact's POM in the Form-based POM Editor, you can also right click on a artifact and choose Materialize Project. If the POM for a particular artifact contains valid SCM information, m2eclipse can "materialize" the project from source control into your workspace.

6.4.3. Browsing Your Workspace Repository

m2eclipse maintains an index of artifacts generated by your Eclipse workspace. This "workspace" repository is shown in Figure 6.10, “Browsing the m2eclipse Workspace Repository” under the Local Repository folder. If you expand this folder, you will see artifacts that correspond to your workspace project as shown in Figure 6.10, “Browsing the m2eclipse Workspace Repository”.

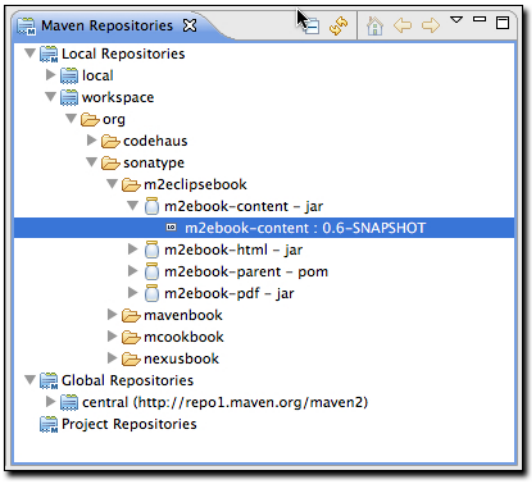

Figure 6.10. Browsing the m2eclipse Workspace Repository

6.4.4. Browsing a Project Repository

The Maven Repositories view is also intelligent enough to keep track of any repositories that have been added to your project via your Maven Settings, an active Maven Profile, or that have been added directly to a project's POM. To demonstrate this feature, add a repository element to a pom.xml, by loading the Form-based POM Editor and clicking on the Repositories tab. Click on the "Create..." button and add a new repository with the following values as shown in Figure 6.11, “Adding a Repository to a Project's POM”.

- Repository Identifier: flexmojos
- Repository Name: Flexmojos Repository
- URL: http://repository.sonatype.org/content/groups/flexgroup/

> **Note**
>
> You will only see the Repositories tab in your Form-based POM Editor, if you have set your Maven preferences to "Show advanced tabs in the POM Editor" under Eclipse, Preferences..., Maven, POM Editor.

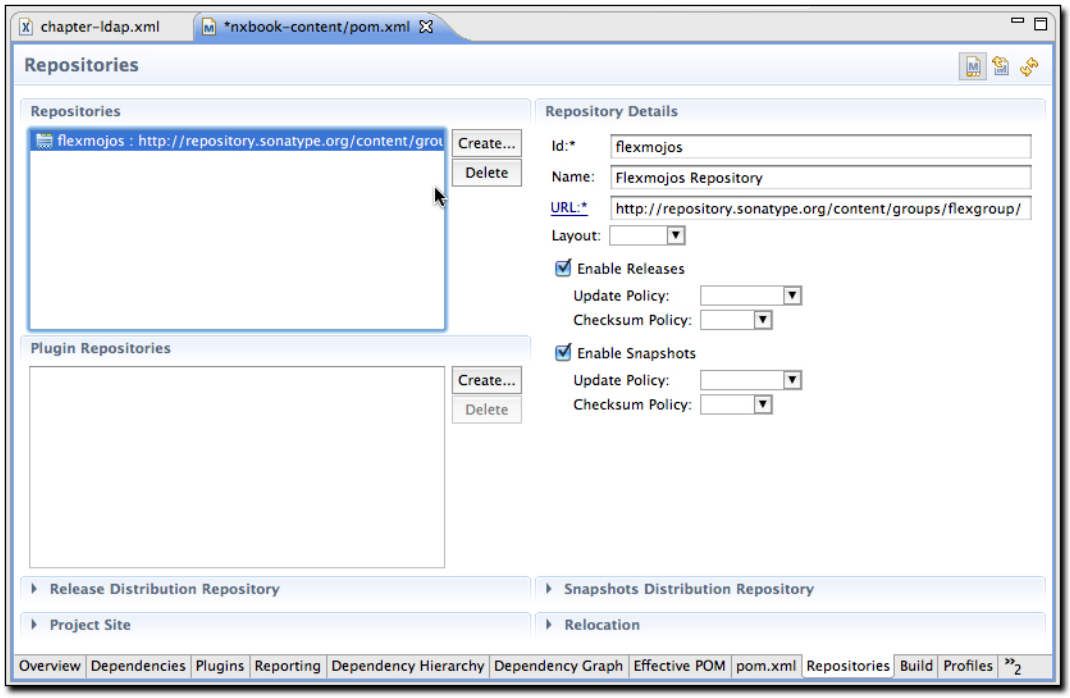

Figure 6.11. Adding a Repository to a Project's POM

Save the POM and open the `pom.xml` tab in the POM Editor. The project's `pom.xml` should contain the repositories element shown in Figure 6.12, “Project POM with a Custom Repository”.

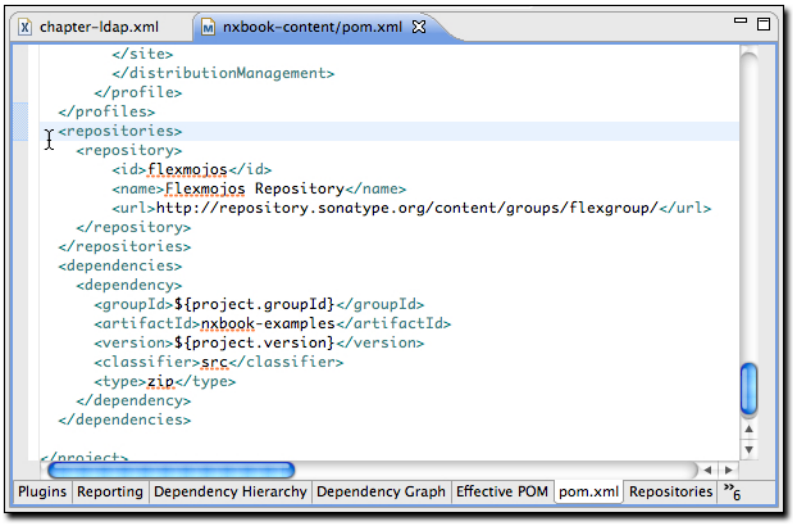

Figure 6.12. Project POM with a Custom Repository

Now that the `pom.xml` contains a custom repository, click on the refresh icon shown in the upper right-hand of Figure 6.13, “Browsing a Project Repository”. The refresh icon looks like two opposing yellow arrows, and clicking this icon will cause the Maven Repositories view to refresh the list of repositories from the selected project and your configured Maven settings.

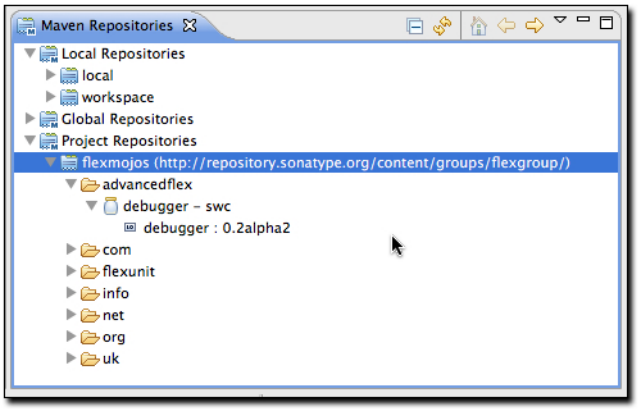

Figure 6.13. Browsing a Project Repository

Once you have added a project repository and clicked on the refresh icon in the Maven Respositories view, you will be able to view the project-specific repository and manipulate the repository index for this project-specific repository.

6.4.5. Browsing Your Local Repository

The Maven Repostitories view allows you to browse and manipulate your local Maven repository index. m2eclipse maintains an index for the contents of your local repository, you can use this interface to browse artifacts that have been loaded into your local repository as shown in Figure 6.14, “Browsing Your Local Maven Repository”.

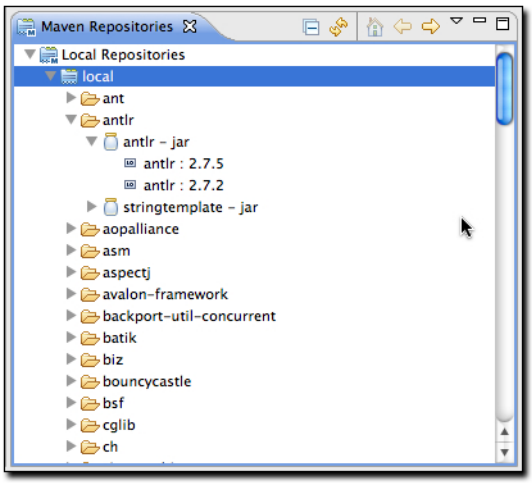

Figure 6.14. Browsing Your Local Maven Repository

6.4.6. Manipulating a Repository Index

Every repository that m2eclipse uses is indexed by the Nexus Indexer. If m2eclipse is using a remote repository, it will download a Nexus index from the remote repository. If m2eclipse is managing a local repository (local or workspace) it will use the open source Nexus indexer to create and maintain a local index. This index is what allows you to quickly search for and locate dependencies by artifactId, groupId, version, or classname. You can manipulate the index that is associated with a Maven repository by right-clicking on a repository in the Maven Repositories view and selecting one of the following actions:

Update Index
: This will update the index by running an incremental update or by downloading and index from a remote repository.

Rebiuld Index
: This will rebuild an index for a local repository by iterating through the contents of a repository and recreating a Nexus index from scratch. This can be a useful tool if there is another process outside of Eclipse that is going to be modifying a local Maven repository.

Disable Index
: Choosing this option causes m2eclipse to skip index generation for a repository. This can come in handy if you have a series of repositories which you do not want to include in artifact searches. If your organization maintains a number of specialized, segregated repositories that hold snapshots, you may not want to include these artifacts in simple searches for artifacts that contain a particular identifier or class.

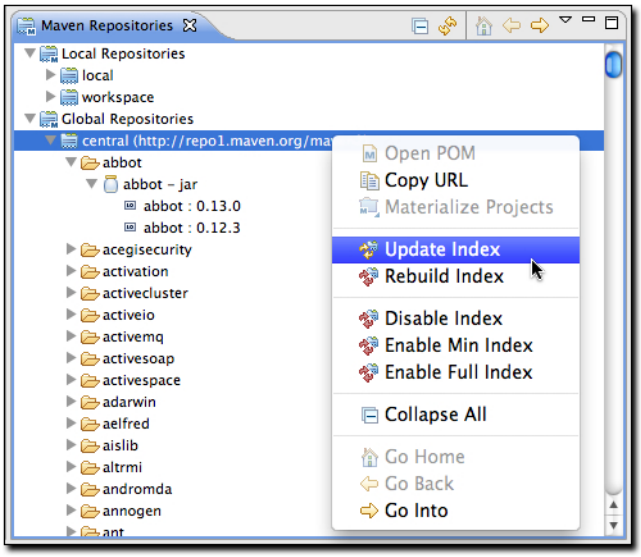

Figure 6.15. Updating a Repository Index

Figure 6.15, “Updating a Repository Index” shows two indexing options that control the scope of a particular Nexus index:

Enable Min Index
: Configures the Nexus Indexer to maintain a minimal index that doesn't contain information about class names.

Enable Full Index
: Configures the Nexus Indexer to maintain a full Index that includes the class names contained within each artifact.

Chapter 7. Using m2eclipse

7.1.1. Adding and Updating Dependencies and Plugins

Let's say we'd like to add a dependency or a plugin to the `camel-core` POM. For the sake of demonstration, we're going to add `commons-lang` as a dependency. (Please note that the functionality for adding a dependency or a plugin is exactly the same so we'll demonstrate it by adding a dependency.)

m2eclipse offers two options for adding dependencies to a project. The first option is by manually editing the POM file to type in the XML to add the dependency. The downside to manually editing the POM file to add a dependency is that you must already know the information about the artifact, or use the features discussed in the next section to manually locate the artifact information in the repository indexes. The upside is that after manually adding the dependency and saving the POM, the project's Maven Dependencies container will be automatically updated to include the new dependency. Figure 7.1, "Manually Adding a Dependency to the Project's POM" shows how I added a dependency for `commons-lang` to the `camel-console` POM and the Maven Dependencies container was automatically updated to included it:

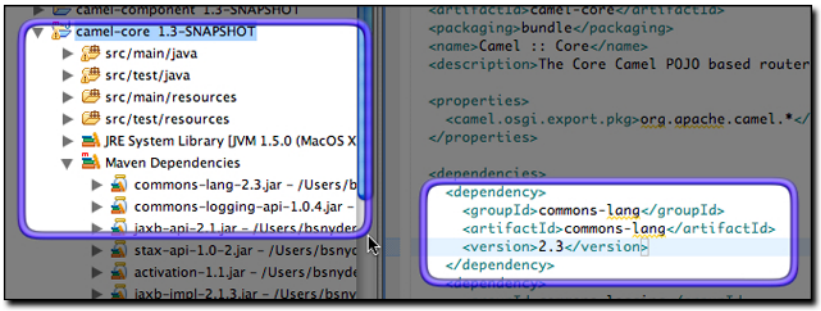

Figure 7.1. Manually Adding a Dependency to the Project's POM

Manually adding a dependency works well but requires more work than the second approach. Upon manually adding the dependency element to the POM, the Eclipse progress in the lower right-hand corner of the Eclipse workbench reflects the action as shown in Figure 7.2, "Updating Maven Dependencies":

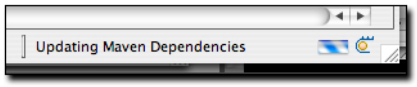

Figure 7.2. Updating Maven Dependencies

The second option for adding a dependency is much easier because you don't have to know any information about the artifact other than its `groupId`. Figure 7.3, "Searching for a Dependency" shows this functionality:

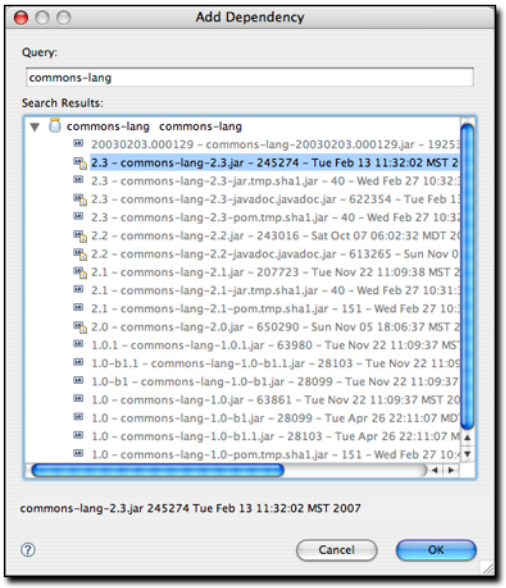

Figure 7.3. Searching for a Dependency

By simply entering a `groupId` into the query field, m2eclipse queries the repository indexes and even shows a version of the artifact that is currently in my local Maven repository. This option is preferred because it is such a tremendous time saver. With m2eclipse, you no longer need to hunt through the central Maven repository for an artifact version.

7.1.2. Downloading Source

If the central Maven repository contains a source artifact for a particular project, you can download the source from the repository and expose it to the Eclipse environment. When you are trying to debug a complex issue in Eclipse, nothing can be easier than being able to right click on a third-party dependency and drill into the code in the Eclipse debugger. Select this option, and m2eclipse will attempt to download the source artifact from the Maven repository. If it is unable to retrieve this source artifact, you should ask the maintainers of the project in question to upload the appropriate Maven source bundle to the central Maven repository.

7.1.3. Opening Project Pages

A Maven POM contains some valuable URLs which a developer may need to consult. These are the project's web page, the URL for the source code repository, a URL for a continuous integration system like Hudson, and a URL for an issue tracker. If these URLs are present in a project's POM, m2eclipse will open these project pages in a browser.

7.1.4. Resolving Dependencies

You can configure a project to resolve dependencies from a workspace. This has the effect of altering the way that Maven locates dependency artifacts. If a project is configured to resolve dependencies from the workspace, these artifacts do not need to be present in your local repository. Assume that project-a and project-b are both in the same Eclipse workspace, and that project-a depends on project-b. If workspace resolution is disabled, the m2eclipse Maven build for `project-a` will only succeed if `project-b`'s artifact is present in the local repository. If workspace resolution is enabled, m2eclipse will resolve the dependency via the Eclipse workspace. In other words, when workspace resolution is enabled, project's don't have to be installed in the local repository to relate to one another.

You can also disable dependency management. This has the effect of telling m2eclipse to stop trying to manage your project's classpath, and it will remove the Maven Dependencies classpath container from your project. If you do this, you are essentially on your own when it comes to managing your project's classpath.

7.2. Analyzing Project Dependencies in m2eclipse

The latest release of m2eclipse contains a POM editor which provides some dependency analysis tools. These tools promise to change the way people maintain and monitor a project's transitive dependencies. One of the main attractions to Maven is the fact that it manages a project's dependencies. If you are writing an application which depends on the Spring Framework's Hibernate3 integration, all you need to do is depend on the `spring-hibernate3` artifact from the Central Maven Repository. Maven then reads this artifact's POM and adds all of the necessary transitive dependencies. While this is a great feature that attracts people to using Maven in the first place, it can often become confusing with a project starts to depend on tens of dependencies, each with tens of transitive dependencies.

Problems start to happen when you depend on a project with a poorly crafted POM which fails to flag dependencies as optional, or when you start encountering conflicts between transitive dependencies. If one of your requirements is to exclude a dependency like `commons-logging` or the `servlet-api`, or if you need to find out why a certain dependency is showing up under a specific scope you will frequently need to invoke the `dependency:tree` and `dependency:resolve` goals from the command-line to track down the offending transitive dependencies.

This is where the POM editor in m2eclipse comes in handy. If you open a project with many dependencies, you can open the Dependency Tree tab and see a two-column display of dependencies as shown in Figure 7.4, “Dependency Tree Tab of the POM Editor”. The left-side of the panel displays a tree of dependencies. The first level of the tree consists of direct dependencies from your project, and each subsequent level lists the dependencies of each dependency. The left-hand side is a great way to figure out how a specific dependency made its way into your project's resolved dependencies. The right-hand side of this panel displays the resolved dependencies. This is the list of effective dependencies after all conflicts and scopes have been applied, and it is the effective list of dependencies that your project will use for compilation, testing, and packaging.

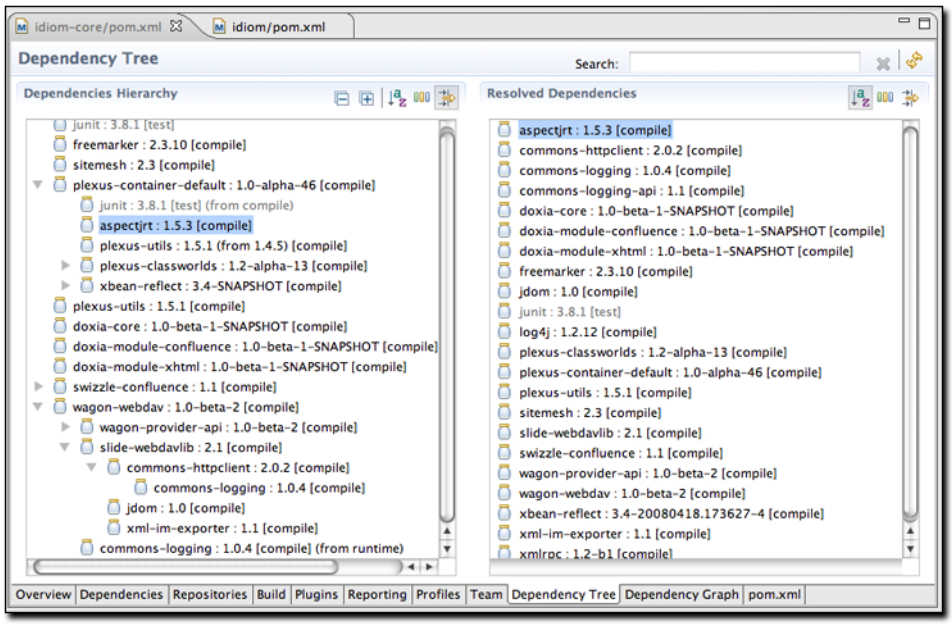

Figure 7.4. Dependency Tree Tab of the POM Editor

The feature which makes the Dependency Tree tab so valuable is that it can be used as an investigative tool to figure out how a specific dependency made it into the list of resolved dependencies. Searching and filtering functionality available in the editor makes it really easy to search and browse trough the project dependencies. You can use “Search” entry field from the editor tool-bar and “Sort” and “Filter” actions from “Dependency Hierarchy” and “Resolved Dependencies” sections to navigate trough dependencies. Figure 7.5, “Locating Dependencies in the Dependency Tree” shows what happens when you click on commons-logging in the "Resolved Dependencies" list. When filtering is enabled in “Dependencies Hierarchy” section, clicking on a resolved dependency filters the hierarchy on the left-hand side of the panel to show all of the node which contributed to the resolved dependency. If you are trying to get rid of a resolved dependency, you can use this tool to find out what dependencies (and what transitive dependencies)

are contributing the artifact to your resolved dependencies. In other words, if you are trying to get rid of something like `commons-logging` from your dependency set, the Dependency Tree tab is the tool you will likely want to use.

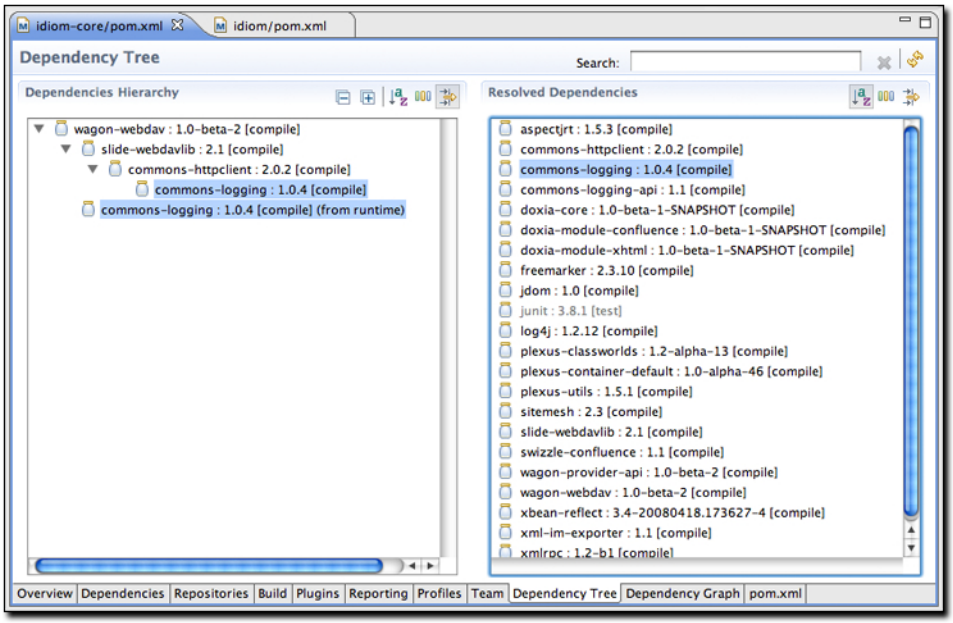

Figure 7.5. Locating Dependencies in the Dependency Tree

m2eclipse also provides you with the ability to view your project's dependencies as a graph. Figure 7.6, “Viewing the Dependencies of a Project as a Graph” shows the dependencies of `idiom-core`. The top-most box is the `idiom-core` project and the other dependencies are shown below it. Direct dependencies are linked from the top box and the transitive dependencies are linked from those. You can select a specific node in the graph to highlight the linked dependencies, or you can use the Search field at the top of the page to find matching nodes.

Note that “open folder” icon on each graph node indicates that the corresponding artifact is present in the Eclipse workspace and “jar” icon indicates that the node's artifact is referenced from the Maven repository.

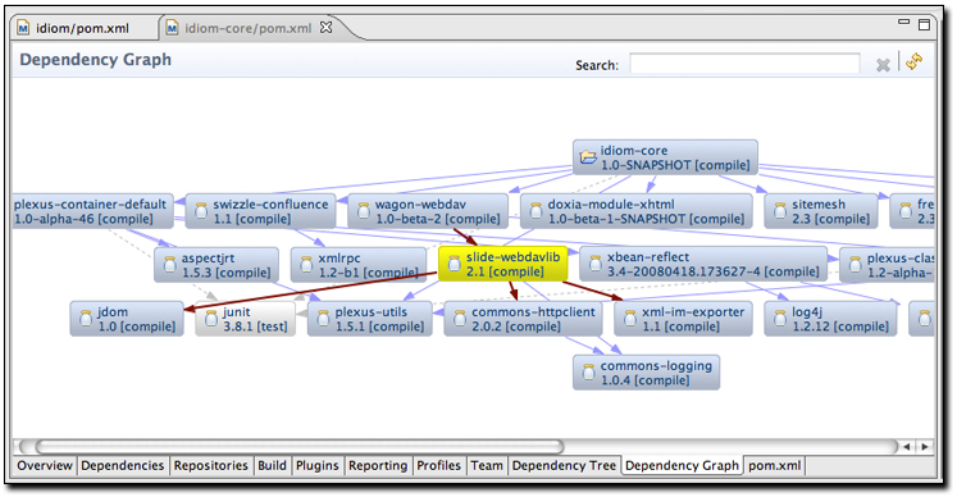

Figure 7.6. Viewing the Dependencies of a Project as a Graph

The graph presentation can be changed by right clicking in the editor. You can choose to show artifact ids, group ids, versions, scopes, or if you want to wrap node text or show icons. Figure 7.7, “Radial Layout of Dependency Graph” shows the same graph from Figure 7.6, “Viewing the Dependencies of a Project as a Graph” with a radial layout.

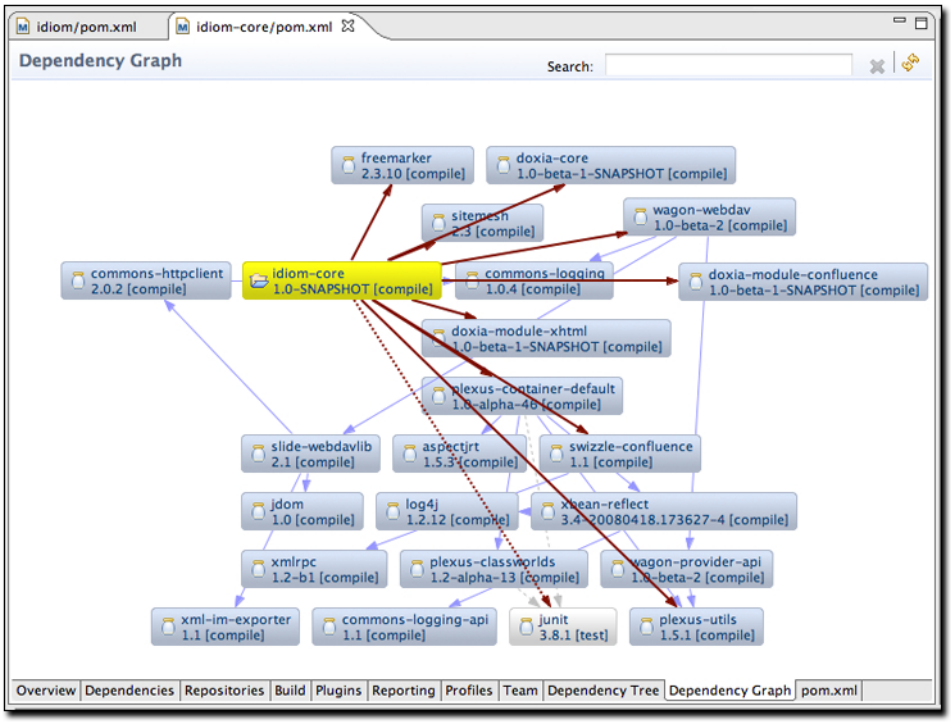

Figure 7.7. Radial Layout of Dependency Graph

Chapter 8. Using m2eclipse

8.1. Working with Maven Projects

The m2eclipse plugin also provides a set of features for working with Maven projects once they are inside of Eclipse. There are many features that ease the ability to use Maven in Eclipse so let's dive right into them. In the previous section, I materialized a Maven project and selected a subproject from the Apache Camel project named camel-core. We'll use that project to demonstrate these features.

By right-clicking on the camel-core project, and selecting the Maven menu item, you can see the available Maven features. Figure 8.1, "Available Maven Features" shows a screenshot of this:

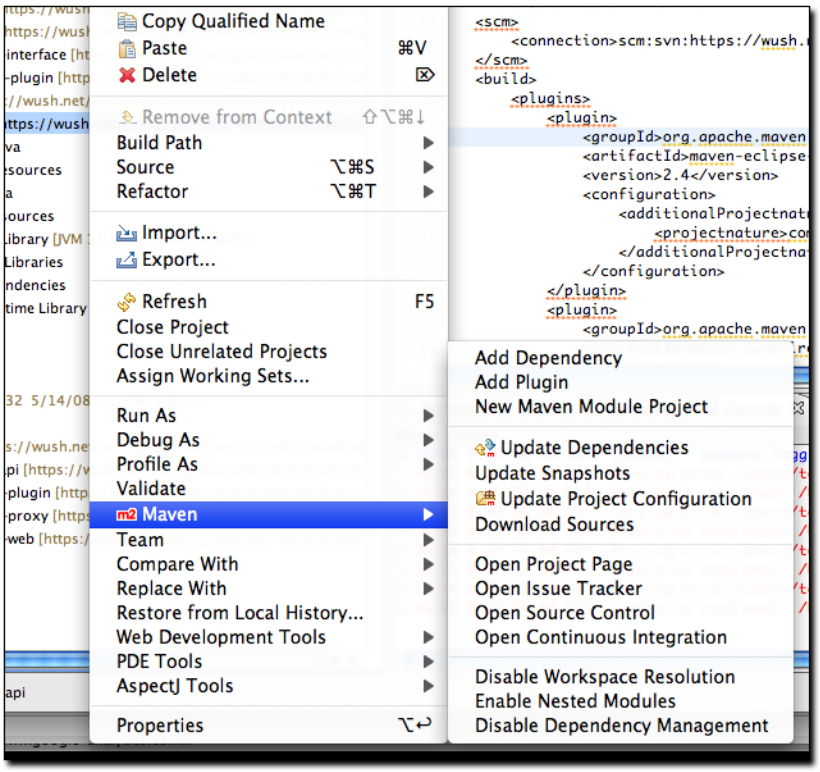

Figure 8.1. Available Maven Features

Notice in Figure 8.1, "Available Maven Features" the available Maven features for the camel-core project, including:

- Adding dependencies and plugins
- Updating dependencies, snapshots and source folders
- Creating a Maven module
- Downloading the source
- Opening Project URLs such as the Project Web Page, Issue Tracker, Source Control, and Continuous Integration tool.
- Enabling/Disabling workspace resolution, nested Maven modules and dependency management

These features are also big time savers so let's review them briefly.

8.1.1. Downloading Source

If the central Maven repository contains a source artifact for a particular project, you can download the source from the repository and expose it to the Eclipse environment. When you are trying to debug a complex issue in Eclipse, nothing can be easier than being able to right click on a third-party dependency and drill into the code in the Eclipse debugger. Select this option, and m2eclipse will attempt to download the source artifact from the Maven repository. If it is unable to retrieve this source artifact, you should ask the maintainers of the project in question to upload the appropriate Maven source bundle to the central Maven repository.

8.1.2. Opening Project Pages

A Maven POM contains some valuable URLs which a developer may need to consult. These are the project's web page, the URL for the source code repository, a URL for a continuous integration system like Hudson, and a URL for an issue tracker. If these URLs are present in a project's POM, m2eclipse will open these project pages in a browser.

8.1.3. Resolving Dependencies

You can configure a project to resolve dependencies from a workspace. This has the effect of altering the way that Maven locates dependency artifacts. If a project is configured to resolve dependencies from the workspace, these artifacts do not need to be present in your local repository. Assume that project-a and project-b are both in the same Eclipse workspace, and that project-a depends on project-b. If workspace resolution is disabled, the m2eclipse Maven build for `project-a` will only succeed if `project-b`'s artifact is present in the local repository. If workspace resolution is enabled, m2eclipse will resolve the dependency via the Eclipse workspace. In other words, when workspace resolution is enabled, project's don't have to be installed in the local repository to relate to one another.

You can also disable dependency management. This has the effect of telling m2eclipse to stop trying to manage your project's classpath, and it will remove the Maven Dependencies classpath container from your project. If you do this, you are essentially on your own when it comes to managing your project's classpath.

8.2. Using the Form-based POM Editor

The latest release of the m2eclipse plugin has a form-based POM editor which allows you to edit every part of a project's `pom.xml` with an easy-to-use GUI interface. To open the POM Editor, click on a project's `pom.xml` file. If you've customized the editors for a pom.xml file, and the POM Editor is not the default editor, you may need to right-click on the file and choose "Open With... / Maven POM Editor". The POM Editor will then display the Overview tab as shown in Figure 8.2, “Overview Tab of POM Editor for idiom-core”.

> Note
>
> The Form-based POM Editor is only available if you selected the POM Editor component when you installed the m2eclipse Eclipse plugin. For more information about installing the m2eclipse plugin, see Section 2.3.1, “Installing m2eclipse Core Components”.

One common complaint about Maven is that it forces a developer to confront large and often overwhelming XML documents in a highly complex multi-module project build. While the authors of this book believe this is a small price to pay for the flexibility of a tool like Maven, the graphical POM editor is a tool that makes it possible for people to use Maven without ever having to know about the XML structure behind a Maven POM.

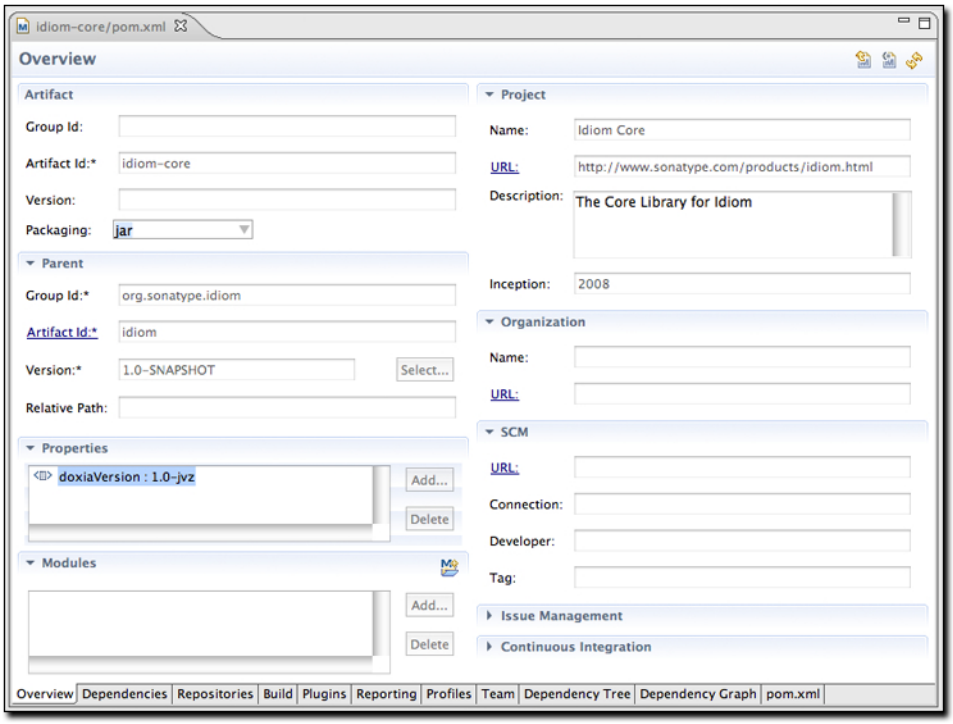

Figure 8.2. Overview Tab of POM Editor for idiom-core

The project shown in Figure 8.2, “Overview Tab of POM Editor for idiom-core” is a project with an `artifactId` of of `idiom-core`. You'll notice that most of the fields in this `idiom-core` project are blank. There is no `groupId` or `version` and there is no SCM information supplied in the POM editor. This is due to the fact that `idiom-core` inherits most of this information from a parent project named `idiom`. If we open the `pom.xml` for the parent project in the POM Editor we would see the Overview tab shown in Figure 8.3, “Overview Tab of POM Editor for idiom Parent Project”.

That “open folder” icon on the various list entries throughout the POM editor indicate that the corresponding entry is present in the Eclipse workspace and “jar” icon indicates artifacts which are referenced from the Maven repository. You can double-click on those entries in order to open its POM in the POM editor. This works for modules, dependencies, plugins and other elements that have corresponding Maven artifacts. Underlined labels in several POM editor sections represent hyperlinks which can be used to open the POM editor for corresponding Maven artifact.

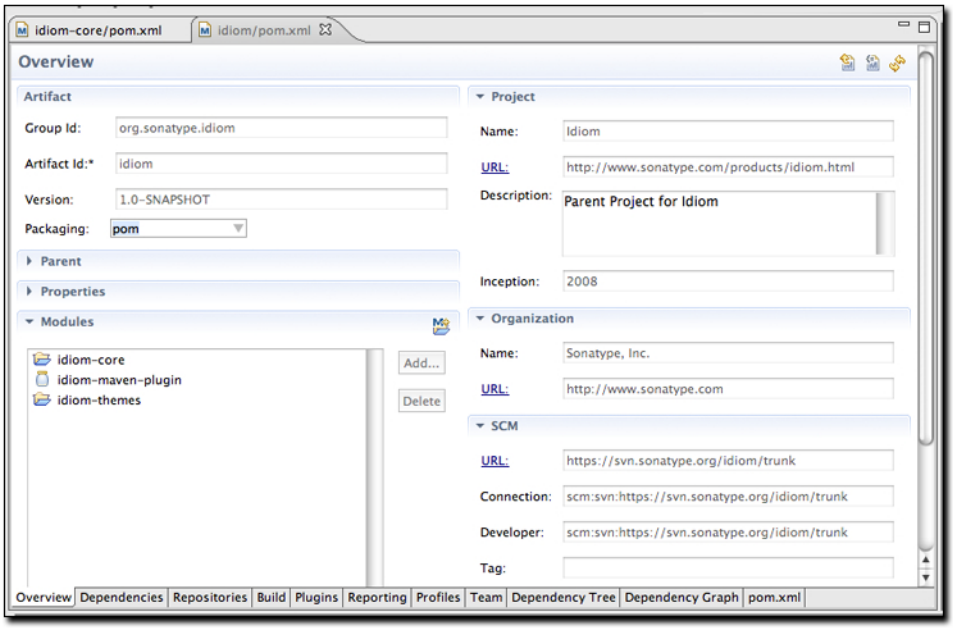

Figure 8.3. Overview Tab of POM Editor for idiom Parent Project

In this parent POM, we see that the `groupId` and `version` are defined and that the parent POM supplies much of the information which was missing in the `idiom-core` project. The POM editor is going to show you the contents of the POM that you are editing, and it will not show you any of the inherited values. If you wanted to look at the `idiom-core` project's effective POM in the POM editor, you can use "Show Effective POM" action from the tool-bar in the upper right-hand corner of the POM editor, which shows a left bracket and an equals sign on a page with a blue M. It will load the effective POM for `idiom-code` in the POM Editor as shown in Figure 8.4, "Effective POM for idiom-core".

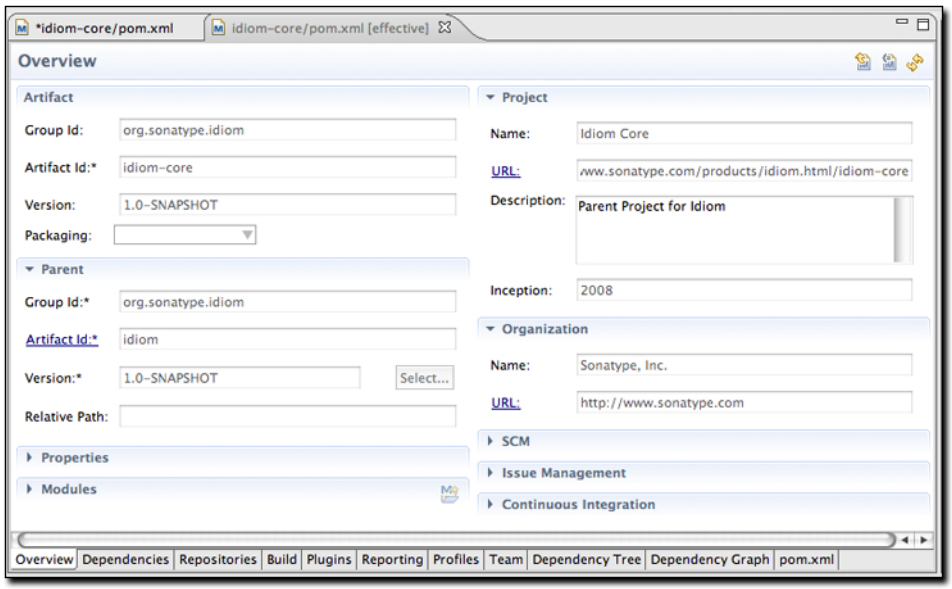

Figure 8.4. Effective POM for idiom-core

This effective view of the POM merges the `idiom-core` POM with the ancestor POMs (the parent, the grandparent, etc.), similarly to "mvn help:effective-pom" command and displays the POM editor with the effective values. Because the POM editor is display a composite view of many different merged POMs, this effective POM Editor is read-only, and you will not be able to update any of the fields in this effective POM view.

If you were looking at the POM editor for the idiom-core project as shown in Figure 8.2, “Overview Tab of POM Editor for idiom-core”, you can also navigate to the parent POM using, ”Open Parent POM” action from the POM editor tool-bar in the upper right-hand of the POM editor.

The POM editor shows a number of showing various information from the POM. The final tab exposes the `pom.xml` as an XML document. There is a dependencies tab shown in Figure 8.5, “Dependencies Tab of the POM Editor” which exposes an easy-to-use interface for adding and editing dependencies to your project, as well as editing the `dependencyManagement` section of the POM. This dependency management screen is also integrated with the artifact searching facilities in the m2eclipse plugin. You can use actions from the editor sections, as well as Ctrl-Space typing assistance for the fields in “Dependency Details” section.

If you need to know more about one of the artifacts, you can use “Open Web Page” action from “Dependency Details” section tool-bar to check the project web page.

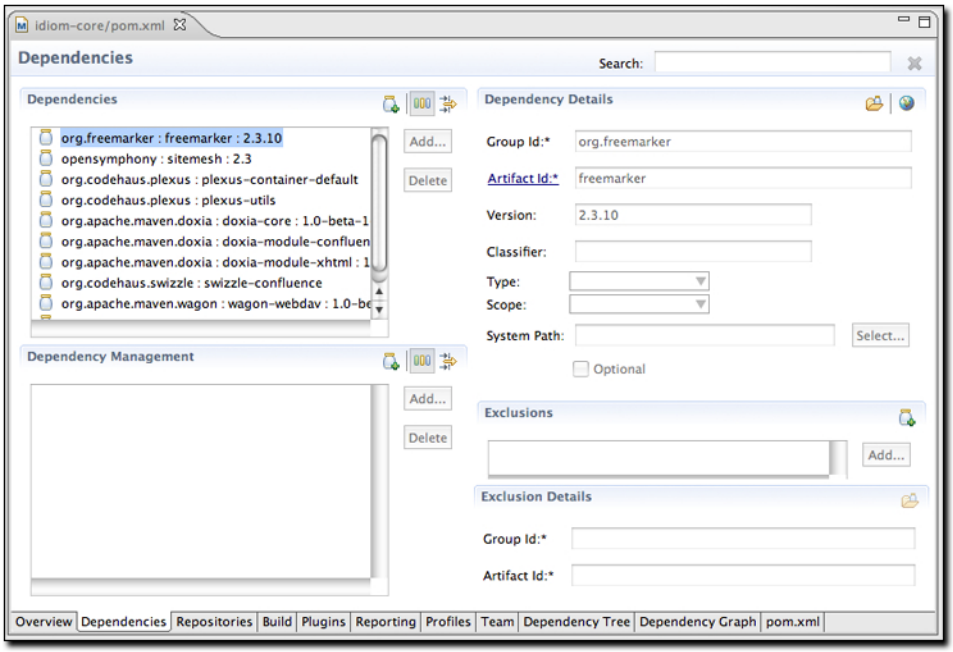

Figure 8.5. Dependencies Tab of the POM Editor

The build tab shown in Figure 8.6, “Build Tab of the POM Editor” provides access to the contents of the build element. From this tab you can customize source directories, add extensions, change the default goal name, and add resources directories.

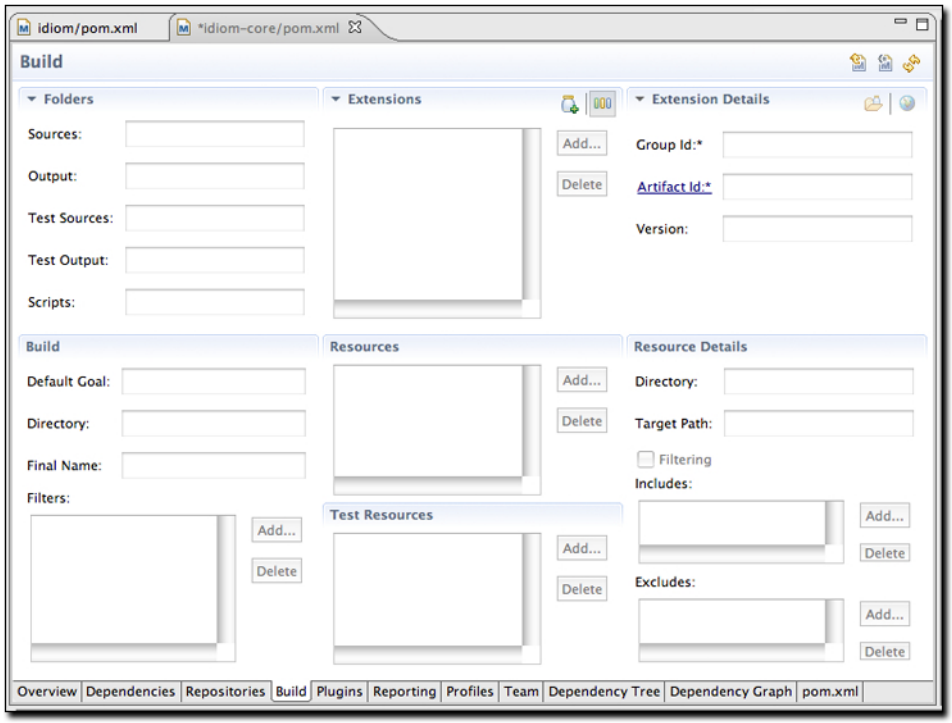

Figure 8.6. Build Tab of the POM Editor

We only showed a small subset of the POM editor. If you are interested in seeing the rest of the tabs, please download and install the m2eclipse plugin.

8.3. Summary

m2eclipse is more than just a simple plugin which adds Maven support to Eclipse, it is a comprehensive integration that will make everything from creating new projects to locating third-party dependencies orders of magnitude easier. m2eclipse is the first step toward an IDE that is aware of the rich semantic treasure that is the central Maven repository. As more people come to use m2eclipse, more projects are going to be releasing Maven Archetypes, and more projects are going to see value in publishing source artifacts to the Maven repository. If you've tried to use Eclipse and Maven together without a tool that can comprehend the hierarchical project relationships that are central to any multi-module Maven project, you will know that the ability to work with nested projects is essential to smooth integration between the Eclipse IDE and Maven.

Appendix A. Creative Commons License

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States license. For more information about this license, see http://creativecommons.org/licenses/by-nc-nd/3.0/us/. You are free to share, copy, distribute, display, and perform the work under the following conditions:

- You must attribute the work to Sonatype, Inc. with a link to http://www.sonatype.com.
- You may not use this work for commercial purposes.
- You may not alter, transform, or build upon this work.

If you redistribute this work on a web page, you must include the following link with the URL in the about attribute listed on a single line (remove the backslashes and join all URL parameters):

```
<div xmlns:cc="http://creativecommons.org/ns#"
     about="http://creativecommons.org/license/results-one?q_1=2&q_1=1\
            &field_commercial=n&field_derivatives=n&field_jurisdiction=us\
            &field_format=StillImage&field_worktitle=Repository%3A+\Management\
            &field_attribute_to_name=Sonatype%2C+Inc.\
            &field_attribute_to_url=http%3A%2F%2Fwww.sonatype.com\
            &field_sourceurl=http%3A%2F%2Fwww.sonatype.com%2Fbook\
            &lang=en_US&language=en_US&n_questions=3">
   <a rel="cc:attributionURL" property="cc:attributionName"
      href="http://www.sonatype.com">Sonatype, Inc.</a> /
   <a rel="license"
        href="http://creativecommons.org/licenses/by-nc-nd/3.0/us/">
     CC BY-NC-ND 3.0</a>
</div>
```

When downloaded or distributed in a jurisdiction other than the United States of America, this work shall be covered by the appropriate ported version of Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 license for the specific jurisdiction. If the Creative Commons Attribution-Noncommercial-No Derivative Works version 3.0 license is not available for a specific jurisdiction, this work shall be covered under the Creative Commons Attribution-Noncommercial-No Derivate Works version 2.5 license for the jurisdiction in which the work was downloaded or distributed. A comprehensive list of jurisdictions for which a Creative Commons license is available can be found on the Creative Commons International web site at http://creativecommons.org/international.

If no ported version of the Creative Commons license exists for a particular jurisdiction, this work shall be covered by the generic, unported Creative Commons Attribution-Noncommercial-No Derivative Works version 3.0 license available from http://creativecommons.org/licenses/by-nc-nd/3.0/.

A.1. Creative Commons BY-NC-ND 3.0 US License

Creative Commons Attribution-NonCommercial-NoDerivs 3.0 United States[1]

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/us/legalcode

1. Definitions

    a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with one or more other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

    b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

    c. "Licensor" means the individual, individuals, entity or entities that offers the Work under the terms of this License.

    d. "Original Author" means the individual, individuals, entity or entities who created the Work.

    e. "Work" means the copyrightable work of authorship offered under the terms of this License.

    f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

    a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works; and,

    b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.

    The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(d) and 4(e).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

    a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of a recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. When You distribute, publicly display, publicly perform, or publicly digitally perform the Work, You may not impose any technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by Section 4(c), as requested.

    b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted

works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

c. If You distribute, publicly display, publicly perform, or publicly digitally perform the Work (as defined in Section 1 above) or Collective Works (as defined in Section 1 above), You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear, if a credit for all contributing authors of the Collective Work appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this clause for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect whether individually or, in the event that Licensor is a member of a performance rights society (e.g. ASCAP, BMI, SESAC), via that society, royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation.

e. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

   UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND ONLY TO THE EXTENT OF ANY RIGHTS HELD IN THE LICENSED WORK BY THE LICENSOR. THE LICENSOR MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MARKETABILITY, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

    a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works (as defined in Section 1 above) from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

    b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

    a. Each time You distribute or publicly digitally perform the Work (as defined in Section 1 above) or a Collective Work (as defined in Section 1 above), the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

    b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

    c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

    d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.

Creative Commons may be contacted at http://creativecommons.org/.

Appendix B. Contributing to the m2eclipse Book

This appendix covers the basics of contributing to the book you are currently reading. This book is an open source project, you can participate in the writing effort if you have an idea for documentation, or you can use the source code of this book as an example of how to build a book with Maven. Sonatype's books are different: they are open writing efforts and we see documentation contributions as having equal value to code contributions. If you are interested in our technology, we'd welcome your contribution.

B.1. Contributor License Agreement (CLA)

In order to contrbute to the m2eclipse book, you will first need to fill out a contributor license agreement. This is a legal agreement between you and Sonatype which ensures that your contributions are not covered by any other legal requirements. Sonatype requires contributors to sign this agreement for all major contributions which are larger than a single section. If your contribution consists of finding and fixing simple typos or suggesting minor changes to the wording or sequence of a particular section, you can contribute these changes via the Sonatype JIRA instance. If you contribution involves direct contribution of a number of sections or chapters you will first need to sign our Contributor License Agreement (CLA).

To download the CLA from the following URL: http://www.sonatype.org/SonatypeCLA.pdf

Once you have completed and signed this document, you can fax it to: (650) 472-9197

B.2. Contributors, Authors, and Editors

As with any open source effort, the contributors to the m2eclipse book are grouped into a simple hierarchy. Sonatype's writing efforts are loosely structured, but we have found it necessary to define some formal categories for contributors.

Reviewers
: Many individuals have read the book and taken the time to report typos and bugs. Reviewers are always credited in the Foreword of the book and they make an important contribution to the quality of the book.

Contributors
: Contributors are individuals who have contributed one or more sections to the book. Many contributors make a one time contribution to a particular section or collection of sections. Contributors are always credited in the Foreword of the book, and if a contributor sustains a constant level of contribution which adds up to the equivalent of an entire chapter, a contributors name will be added to the list of contributing authors.

Authors
: Authors have made a significant contribution to the book equal to the equivalent of one or more chapters. A long-time contributor can also be transitioned to the status of Author at the discretion of an Editor. Authors are often given editorial control over specific chapters or sections of a book, working with Contributors to review, accept, and refine contributions to defined sections of the book.

Editors
: An Author can also be an Editor. Each book has at least one editor (and ideally no more than two Editors at any time). On Sonatype Open book projects, Editors are the arbiters of content, they review content submissions and make final decisions about content direction.

For each of these levels, the adjective "Active" can be used if a contributor, author, or editor has been active during the previous 12 months. If you have any questions about contributor status, send any inquiries to book@sonatype.com[1].

[1] mailto:book@sonatype.com

B.3. Tools Used to Build and Write this Book

The following tools are used to write this book. If you are interested in contributing to this book, you will need to download the following software:

Apache Maven
: Of course we use Maven to build the book. We're always updating our book builds to use the latest production-ready version of Maven, so feel free to download and install the latest available version.

  To download Maven, go to http://maven.apache.org and click on Downloads.

XMLmind XML Editor
: Whie there are many Docbook-capable XML Editors on the market, XMLmind is one of the more capable, "WYSIWYG" Docbook editors on the market. This means that the experience is more like a modern word-processor and less like an "XML Editor". That being said, XMLmind can take some getting used to. For a good introduction to some of the more "ergonomic" approaches to using this tool, see James Elliott's "Getting Productive with XMLmind"[2] article on the O'Reilly site.

  If you are new to Docbook, you will need to invest a day or two experimenting with key shortcuts. Don't approach this tool as just another "word processor" as you are guaranteed come away from the experience frustrated and disappointed. The process of writing a Word document is straightforward. The process of writing a technical book with all of the associated inline markup and semantic "stuff" that goes into making this a value reference is an order of magnitude more involved than what you might be used to. Just give yourself time to learn the tool. In six months, you won't be able to write without it.

  To Download XMLmind XML Editor, go to http://www.xmlmind.com/xmleditor/.

Git
: Sonatype stores the source code for all books in Github so you will need to download Git. In addition to downloading Git, if you need read/write access to the repository you will also need to sign up for an account on GitHub (http://www.github.com).

  Download the latest version of Git from http://git-scm.com/

This next set of tools are optional, and are only required if you are involved in generating diagrams or formatting the final PDF for the pre-print production process. In short, there is only one or two people who will need to have access to the following set of tools, and, in a normal publishing house, all of these functions would likely be performed by a separate "Production" team.

Omigraffle
: Many of the diagrams in this book have been generated using a OSX-specific tool named Omnigraffle.

  If you are interested in helping us create diagrams don't feel compelled to purchase a copy of Omnigraffle. Send us a rough outline of your diagram, and Sonatype will gladly transform your idea into a diagram if your contributions are accepted into the book.

Adobe Photoshop
: All of the screenshots are generated using simple screen capture tools. The resulting raw images (PNGs) are then processed using a set of very simple Photoshop macros. These macros add a border to each screenshot and apply a standard drop shadow. Once the drop shadow has been applied, these macros then save a 72 dpi PNG image for the HTML version of the book in addition to a 150 dpi PDF image for the printed version of the book,

  While Adobe Photoshop is a capable (and somewhat formidable) graphics manipulation tool, Sonatype is exploring alternatives to using this commercial utility in the content generation process. Alternatives currently being investigated are open source packages such as GIMP or systems which can rely on ImageMagick for scripted conversion of raw screenshots to multiple web and print image formats.

[2] http://www.xml.com/pub/a/2007/06/20/getting-productive-with-xmlmind.html

> **Note**
>
> If you are interested in contributing, but you do not want to bother with the process of formatting images for both web and print, Sonatype welcomes contributions which include raw screencaptures. We can take care of the formatting.

Adobe Illustrator
: The book cover, the promotional material insert, and the print binder for Lulu are all generated with Adobe Illustrator. Adobe Illustrator can open and edit PDFs natively, and can be used to generate the static PDFs which are concatenated together to produce the final book output.

B.4. How to Build the Book

You can probably guess what tool is used to build this book - Apache Maven. To build this book, follow these simple steps:

1. Clone the book's Git repository. To clone this book's repository execute the following command at a command-line:

   ```
   $ git clone git@github.com:sonatype/m2eclipse-book.git
   ...a bunch of Git output...
   ```

   Running this command will create a subdirectory named m2eclipse-book which is a copy of this book's source.

2. Assuming you've already installed Apache Maven 3, change directories to the `m2eclipse-book/` directory and run: **`mvn clean install`**.

   ```
   $ cd m2eclipse-book
   $ mvn clean install
   ...5-10 minutes of Maven build output...
   ```

   > **Warning**
   >
   > This project makes use of both the `repositories` and `pluginRepositories` element in the top-level POM. If you have configured your Maven Settings to point to Nexus, you may need to add the repositories listed in this POM to your Nexus group.

Once the Maven build has completed, you can then access the following build artifacts:

- The m2eclipse Book PDF can be found in `m2ebook-pdf/target`
- The m2eclipse Book ePub can be found in `m2ebook-epub/target`
- The m2eclipse Book HTML can be found in `m2ebook-html/target`

B.5. Book Project Layout

The m2eclipse book is a multi-module project which contains the following projects:

m2ebook-content
: This project contains all of the book's source code and figures in `src/main/resources`. If you need to update the book's DocBook XML, you will find all of the Docbook XML in `src/main/resources`, and if you are manipulating figures, you will find these images under the `src/main/resources/figs` directory. The build output for this project is a JAR which contains the DocBook XML source and the figures.

m2ebook-html
: This project contains the HTML-specific XSL stylesheets and project configuration to apply the DocBook XSL stylesheets to the DocBook source content from nxbook-content. This project also contains some HTML-specific admonition graphics. The customization stylesheet and graphics are found in `src/main/resources`.

m2ebook-pdf
: This project contains the PDF-specific XSL stylesheets, any pre-print assets such as covers, and any additional PDFs which are combined to produce the final, electronic output. All images, PDFs, and stylesheets can be found in `src/main/resources`.

m2ebook-epub
: This project contains the ePub-specific XSL stylesheets. The ePub specific stylesheets can be found in `src/main/resources`.

m2ebook-site
: This project combines all of the book projects into a single, deployable site and is only meant to be published by internal Sonatype resources. When the books are published, the build uses the Site plugin to perform a deployment using the Maven SSH Wagon.

B.6. Subscribing to the Book Developers List

Sonatype maintains a Book Developers mailing list as a single mailing list for contributors, authors, and editors working on any of our Sonatype Open Books. This is a high volume list which contains both discussion and automated emails from GitHub and our Sonatype Matrix continuous integration server.

To subscribe to this mailing list, send and email to: book-dev-subscribe@sonatype.org[3]

[3] mailto:book-dev-subscribe@sonatype.org

Appendix C. Book Revision History

The following sections list changes made to the book in reverse chronological order starting with 1.3.

C.1. Changes in Edition 1.2

The following changes were made in 1.2:

- Added a new appendix: Appendix B, Contributing to the m2eclipse Book. This chapter provides some of the basic information would be required by someone looking to participate in the book project including information about the tools used to write the book, how to clone the Git repository, and how to execute the Maven build for this book. (MEBOOK-78 and MEBOOK-79)
- The book now contains instructions for people who want to subscribe to the book announcement mailing list. (MEBOOK-77)
- Updated the Book's copyright to 2011. (MEBOOK-74 and MEBOOK-82)
- Update version of m2eclipse to 0.12.0. (MEBOOK-84)

C.2. Changes in Edition 1.1

The following changes were introduced in Edition 1.1 in July, 2010:

- Added Section 2.2, “Installing m2eclipse in Eclipse 3.6 (Helios) with the Eclipse Marketplace”

C.3. Changes in Edition 0.8

The following changes were introduced in Edition 0.8 on February 15, 2010:

- Updated book to cover m2eclipse 0.10.0. (MEBOOK-49[1])
- Updated Section 2.3.3.1, “Installing Subclipse” (MEBOOK-52[2])
- Updated Section 2.3.3.2, “Installing Mylyn” (MEBOOK-53[3])
- Updated Section 2.3.3.3, “Installing the Web Tools Platform (WTP)” (MEBOOK-54[4])
- Added Section 2.1, “Installing the Eclipse IDE” (MEBOOK-57[5])
- Update Chapter 2, Installing m2eclipse to provide instructions for an Eclipse 3.5 plugin installation. The initial versions of this book discuss Eclipse 3.2 which had a different approach to installing Eclipse plugins. (MEBOOK-56[6])
- Removed reference to AJDT support. (MEBOOK-50[7])
- Chapter 2, Installing m2eclipse now discusses the Core and Extras update sites. (MEBOOK-55[8])

C.4. Changes in Edition 0.7

The following changes were introduced in Edition 0.7 on November 16, 2009:

- Created a new chapter Chapter 3, Creating and Importing Projects. (MEBOOK-40[9])

C.5. Changes in Edition 0.6

The following changes were introduced in Edition 0.6 on September 15, 2009:

- Modified project to generate a PDF for pre-print:
  - Embedded the fonts in the generated PDF. (MEBOOK-17[10])
  - Resized book to Royal Quarto (7.444" x 9.681"). (MEBOOK-16[11])
  - Added a Title Page to the PDF. (MEBOOK-21[12])
  - Added a blank page to the end of the PDF. (MEBOOK-20[13])
  - Automated the production of print and web figures. (MEBOOK-23[14])
  - Standardized on 0.5" Margins. (MEBOOK-22[15])
  - Added the appropriate roles to all imageobjects. (MEBOOK-18[16])
  - Assigned the book an ISBN. (MEBOOK-32[17])
  - Added the full text of the Creative Commons license to the front matter. (MEBOOK-33[18])
- Fixed a typo: redundant mention of Subversion in Section 3.1.1, “Checking Out a Maven Project from SCM”. (MEBOOK-12[19])
- Fixed a typo: capitalized "eclipse" in Section 8.1.3, “Resolving Dependencies”. (MEBOOK-9[20])
- Added clarification about the availability of the Form-based POM Editor to Section 8.2, “Using the Form-based POM Editor”. (MEBOOK-8[21])
- Added new section about the Maven Repository view: ???. (MEBOOK-19[22])
- Address feedback from Proofread of Edition 0.5 PDF:
  - Fixed a typo: removed duplicate "about" from page 29. (MEBOOK-28[23])
  - Removed extraneous space from page 28: (MEBOOK-29[24])
  - Fixed a spelling error on page 51. (MEBOOK-25[25])

Index