Using Maven within the Eclipse IDE - Tutorial

Lars Vogel, Simon Scholz, (c) 2016 vogella GmbHVersion 1.4,29.05.2016

Table of Contents

* [1. Using Maven with the Eclipse IDE](http://www.vogella.com/tutorials/EclipseMaven/article.html#m2e_overview)
* [2. Installation and configuration of Maven for Eclipse](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_eclipseinstallation)
  + [2.1. Install the Maven support for Eclipse (m2e)](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_eclipseinstallation_m2es)
  + [2.2. Download the Maven index](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_eclipseinstallation_index)
* [3. Exercise: Create a new Maven enabled project via Eclipse](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject)
  + [3.1. Target of this exercise](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject1)
  + [3.2. Create Maven project](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject2)
  + [3.3. Run the build](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject_runningthebuild)
  + [3.4. Adding dependencies to your project](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject_dependency)
  + [3.5. Use library](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject_usage)
* [4. Exercise: Add Maven support to a Java project in Eclipse](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipseproject)
  + [4.1. Target of this exercise](http://www.vogella.com/tutorials/EclipseMaven/article.html#target-of-this-exercise)
  + [4.2. Create Java project](http://www.vogella.com/tutorials/EclipseMaven/article.html#create-java-project)
  + [4.3. Convert to Maven project](http://www.vogella.com/tutorials/EclipseMaven/article.html#convert-to-maven-project)
  + [4.4. Execute the Maven build](http://www.vogella.com/tutorials/EclipseMaven/article.html#execute-the-maven-build)
* [5. Exercise: Create a Java web project in Eclipse using Maven](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_webeproject)
* [6. Target of this exercise](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_webeproject_target)
  + [6.1. Create Maven web project project](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_webeproject_project)
  + [6.2. Build your project](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_webeproject_projectbuild)
  + [6.3. Run on the server](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_webeproject_runserver)
* [7. References for Webdevelopment with Eclipse](http://www.vogella.com/tutorials/EclipseMaven/article.html#webserver_configuration)
* [8. Eclipse Maven (m2e) resources](http://www.vogella.com/tutorials/EclipseMaven/article.html#eclipse-maven-m2e-resources)

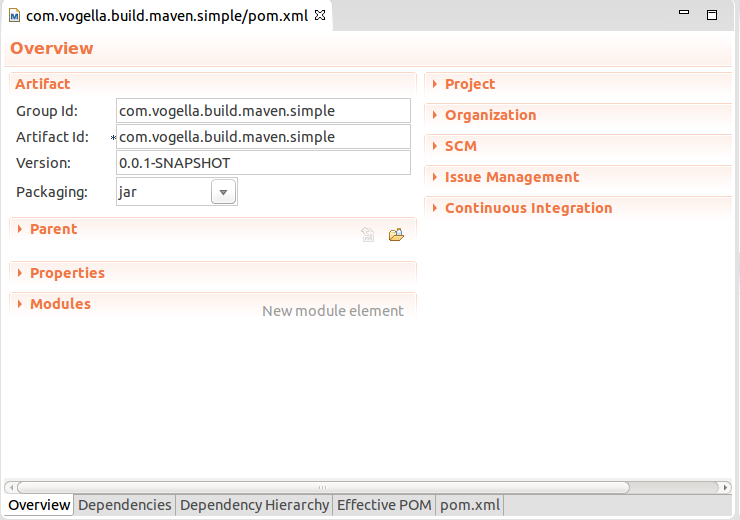
*This tutorial describes the usage of Maven within the Eclipse IDE for building Java applications.*

1. Using Maven with the Eclipse IDE

The Eclipse IDE provides excellent support for the Maven. This tooling is developed in the *M2Eclipse* project.

This tooling manages the project dependencies and updates the classpath of the project dependencies in the Eclipse IDE. It ensures that the Maven experience in Eclipse is as smooth as possible. The tooling also provides different kind of wizards import andto create new Maven based projects.

It also provides an editor for the *pom.xml* Maven configuration file via a structured interface. You can select the tab labeled *pom.xml* to edit the XML data directly.





2. Installation and configuration of Maven for Eclipse

2.1. Install the Maven support for Eclipse (m2e)

Most Eclipse downloads include the Maven tooling already. If it is missing in your installation, you can install it via the main update of your release via Help ▸ Install New Software. The following listing contains the update site for the Neon release and an update site maintained by the m2e project.

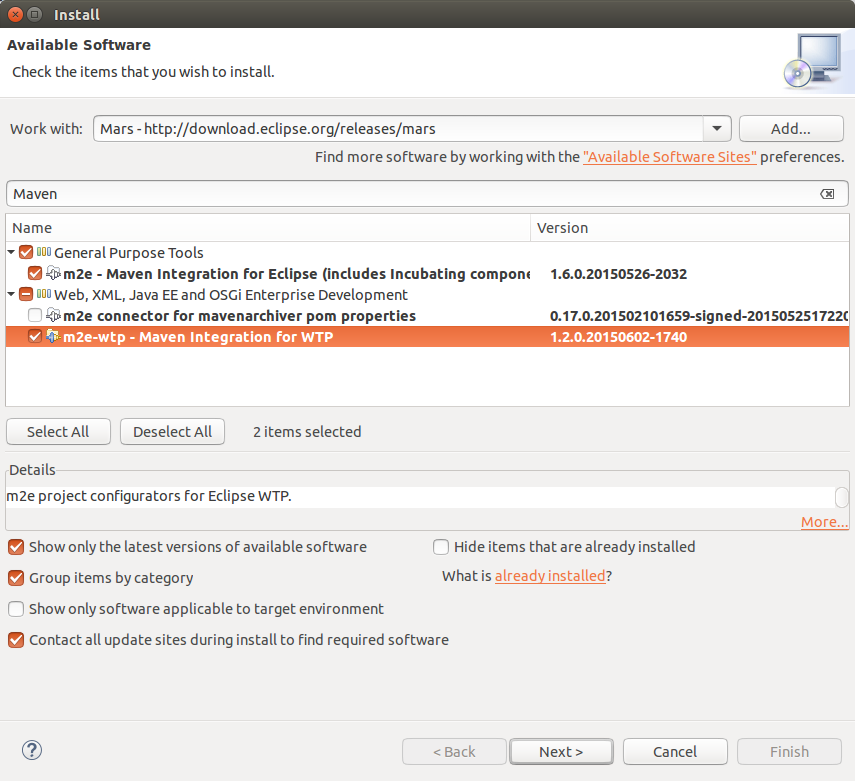
// Neon update site

http://download.eclipse.org/releases/neon

// Update site provided by m2e project

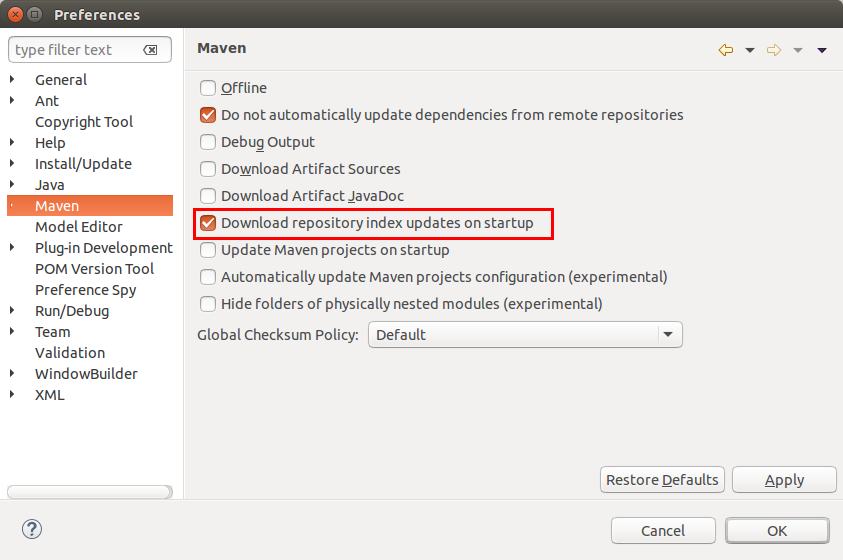
http://download.eclipse.org/technology/m2e/releases

For the usage of Maven for Java projects, you only need the m2e component. For Java web development you also want the m2e-wtp entry.



2.2. Download the Maven index

By default, the Maven tooling does not download the Maven index for the Eclipse IDE. Via the Maven index you can search for dependencies, select them and add them to your pom file. To download the index, select Windows ▸Preferences ▸ Maven and enable the *Download repository index updates on startup* option.



After changing this setting, restart Eclipse. This triggers the download of the Maven index. You may want to remove this flag after restarting to avoid network traffic at every start of Eclipse.

|  |  |
| --- | --- |
|  | The m2e team works on a way to dynamically query for dependencies. Please register for the following bug to show that you are interested in this development: Provide an alternative Artifact search mechanism in Eclipse Maven - <https://bugs.eclipse.org/bugs/show_bug.cgi?id=478647>. |

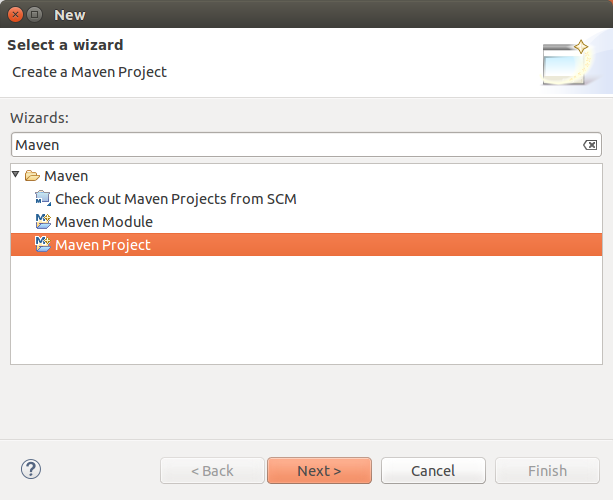
3. Exercise: Create a new Maven enabled project via Eclipse

3.1. Target of this exercise

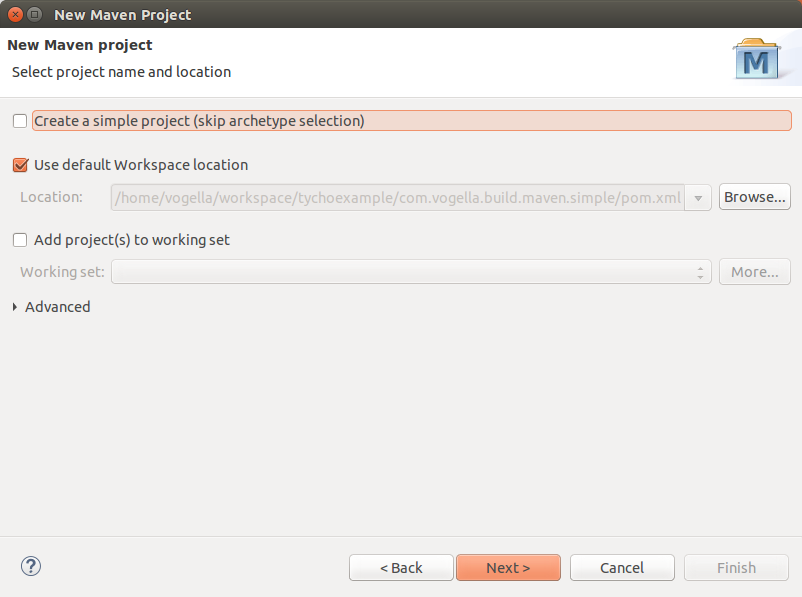
This exercise demonstrates the creation of a new Maven enabled project in Eclipse.

3.2. Create Maven project

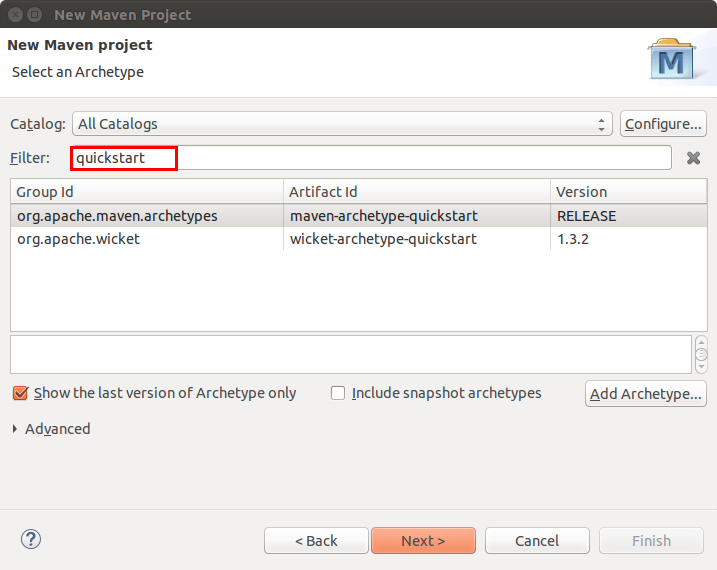
Create a new Maven project via File ▸ New ▸ Other…​ ▸ Maven ▸ Maven Project.



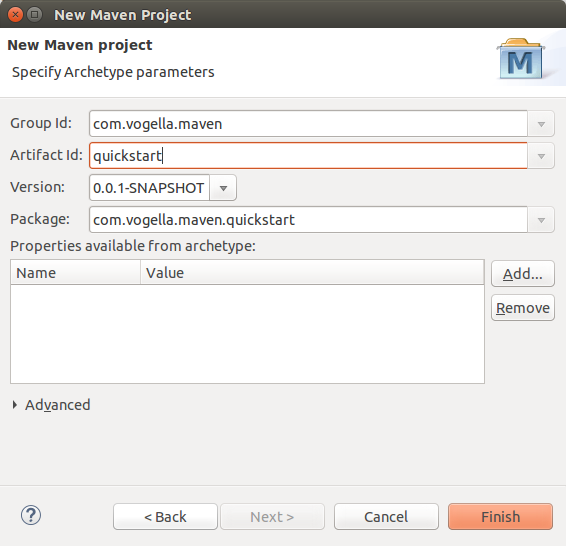
On the first wizard page you can select if you want to create a simple project. In this case you skip the archetype selection. In this exercise we want to use an archetype as template for our project creation.



Press next, filter for the "quickstart" archetype and select the maven-archetype-quickstart entry. This is the classical Maven example archetype for project creation.

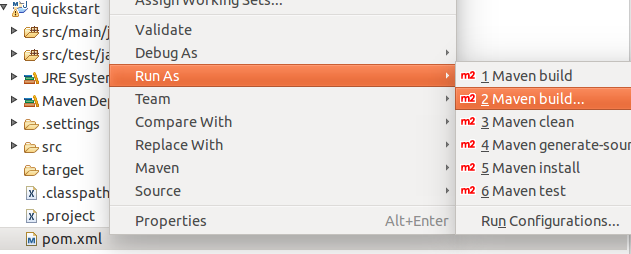


On the last tab enter the GAV of your project similar to the following screenshot.

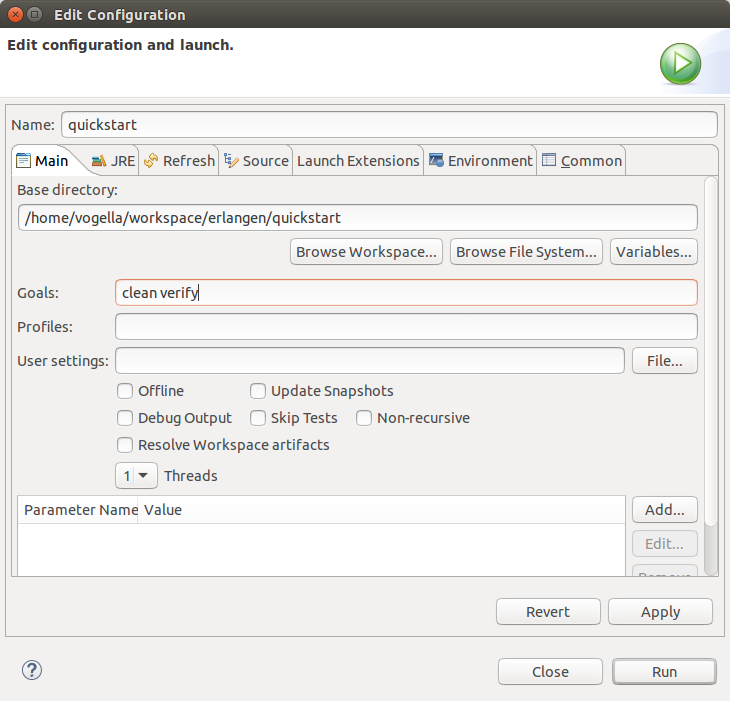


3.3. Run the build

Validate that the generate setup works correctly by running the build. For this right-click the *pom.xml* file and select Run As ▸ Maven build.



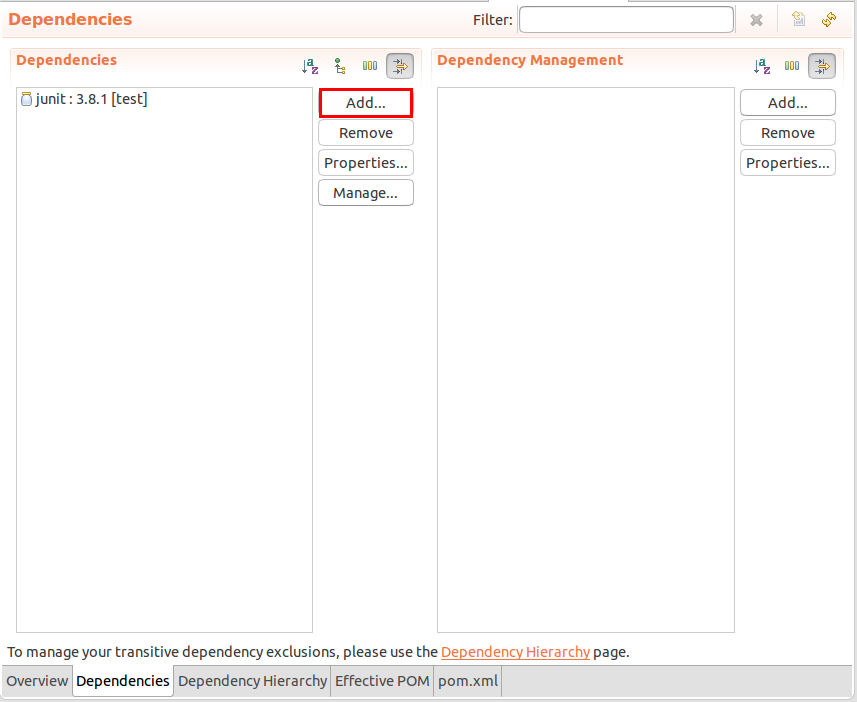
This opens a dialog which allows to define the parameters for the start. Enter clean verify in the *Goals:* field and press the *Run* button.



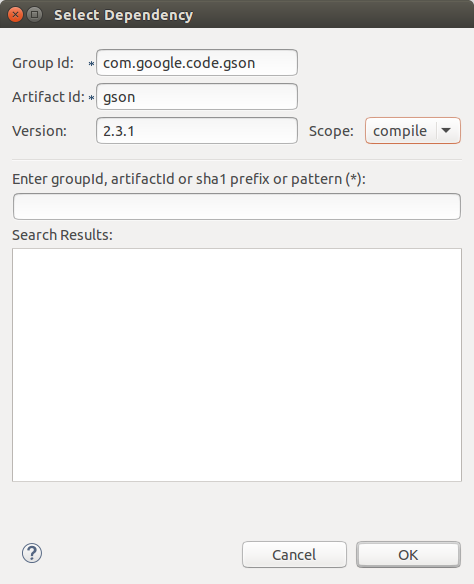
3.4. Adding dependencies to your project

The Eclipse Maven tooling makes adding dependencies to the classpath of your project simple. In can directly add it to your pom file, or use the *Dependencies* tab of the pom editor.

Switch to the *Dependencies* tab and press the *Add* button.



In this example we add Gson as dependency. For this we use the GAV which we found via the [http://search.maven.org](http://search.maven.org/)website. If the Maven index was downloaded (See [Download the Maven index](http://www.vogella.com/tutorials/EclipseMaven/article.html#maven_eclipseinstallation_index) you can also search directly this dependency via the dialog.



3.5. Use library

Change or create the App.java class in your src/main/java folder. This classes uses Gson. As Maven added it to your classpath, it should compile and you should be able to start the class via Eclipse.

**package** com.vogella.maven.lars;

**import** com.google.gson.Gson;

**public** **class** **App**

{

**public** **static** **void** main( String**[]** args )

{

Gson gson = **new** Gson();

System.out.println(gson.toJson("Hello World!") );

}

}

4. Exercise: Add Maven support to a Java project in Eclipse

4.1. Target of this exercise

This exercise demonstrates how to convert a Java project to a Maven project.

4.2. Create Java project

Create a new Java project called *com.vogella.build.maven.simple* in Eclipse.

Add one class called Main. This class should have a main method, which write "Hello Maven!" to the command line.

**package** com.vogella.build.maven.simple;

**public** **class** **Main** {

**public** **static** **void** main(String**[]** args) {

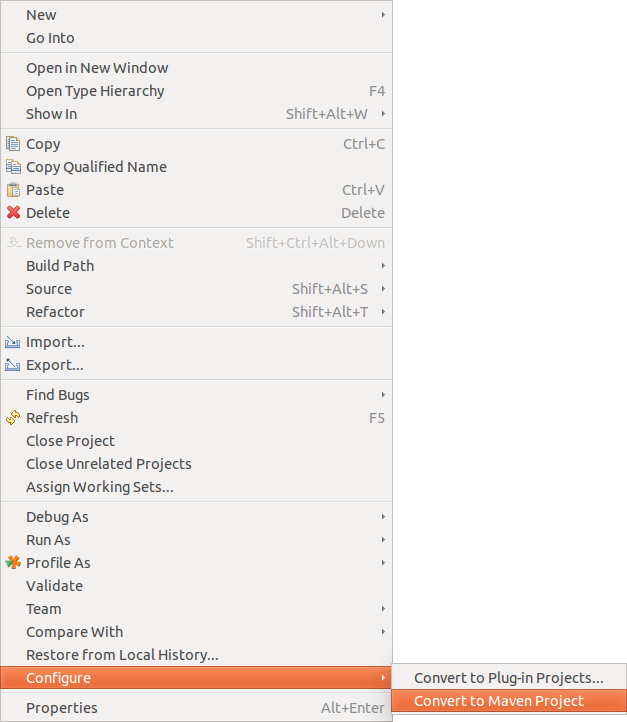
System.out.println("Hello Maven!");

}

}

4.3. Convert to Maven project

Select your project, right-click on it and select Configure ▸ Convert to Maven project…​.



This creates a *pom.xml* file similar to the following.

<project

xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.vogella.build.maven.simple</groupId>

<artifactId>com.vogella.build.maven.simple</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

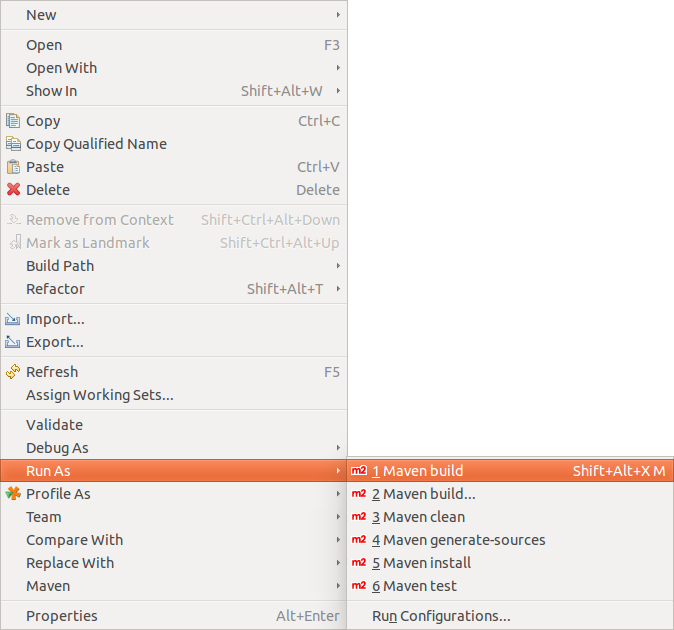
</plugins>

</build>

</project>

4.4. Execute the Maven build

Right-click the *pom.xml* file and select Run As ▸ Maven build.



Enter "clean install" as Goal.

|  |  |
| --- | --- |
|  | You have to enter the goals manually. The *Select…​* button does not work, the dialog it displays is always empty. |

Press the **Finish** button. This starts the build, which you can follow in the *Console* view.

Once the build finishes, press F5 on the project to refresh it. You see a *target* folder, which contains the build artifacts, e.g., a JAR file.

5. Exercise: Create a Java web project in Eclipse using Maven

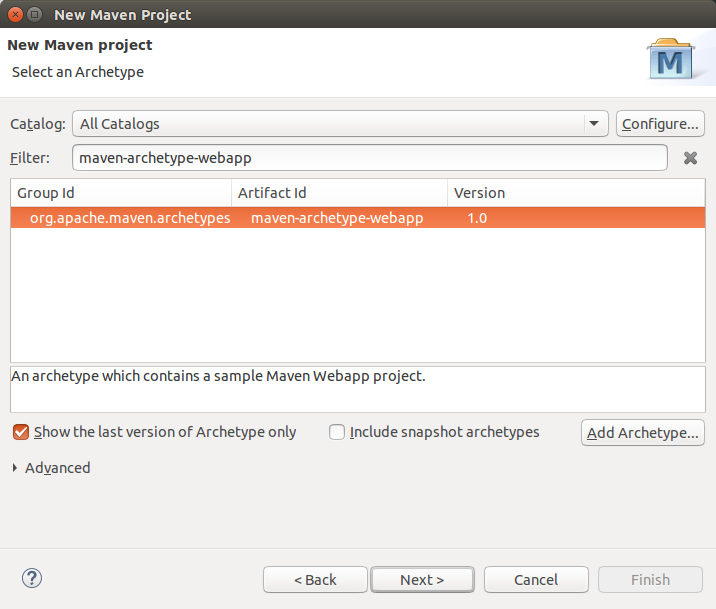
6. Target of this exercise

This exercise demonstrates how to create a web application in Eclipse which uses Maven. It assumes that you have already configured Eclipse for the creation of web applications.

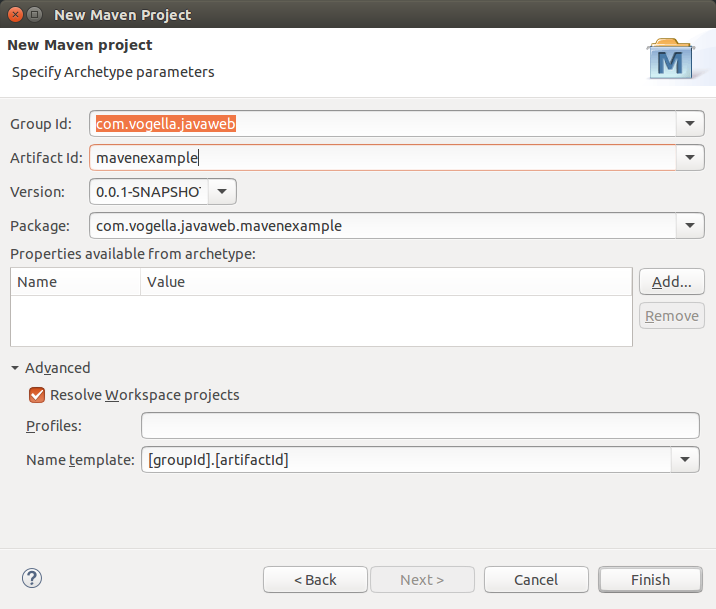
6.1. Create Maven web project project

Create a new Maven project called *com.vogella.javaweb.maven.first* via the File ▸ New ▸ Other ▸ Maven ▸ Maven Projectentry.

On the archetype selection, select the *maven-archetype-webapp* entry and click the *Next* button.



Enter the group, artifact and version of your new Maven component.



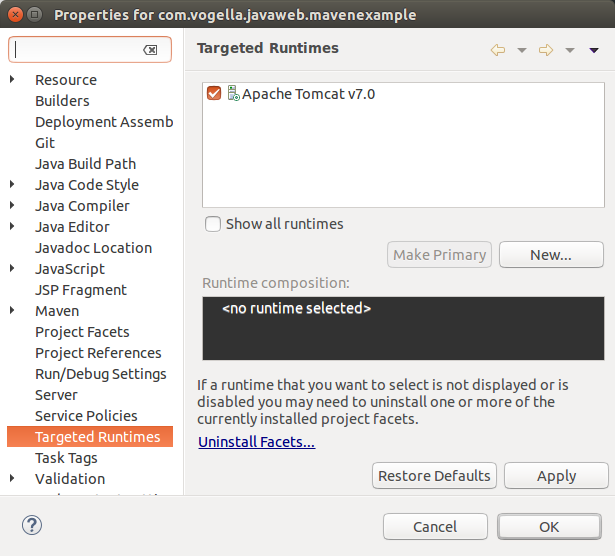
|  |  |
| --- | --- |
|  | You may see the error: *The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Path*. To fix this, right click on your project and select *Properties*. On the *Targeted Runtimes* select your web server entry, e.g., Tomcat. |

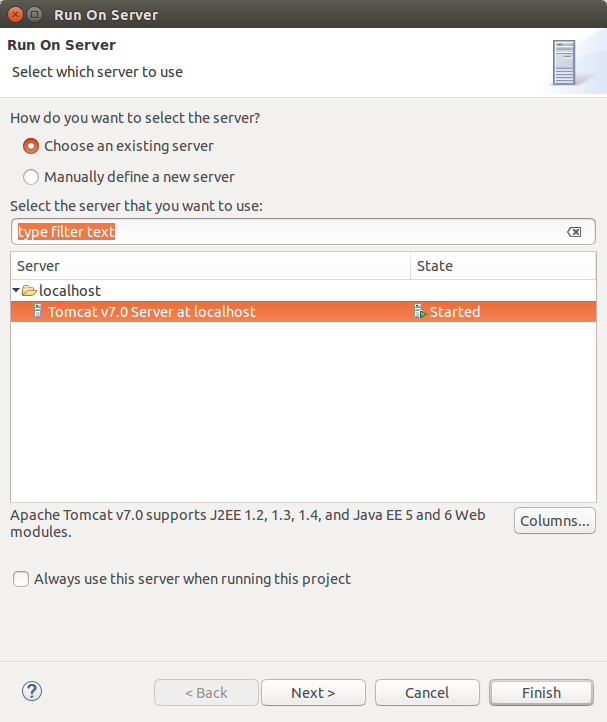
6.2. Build your project

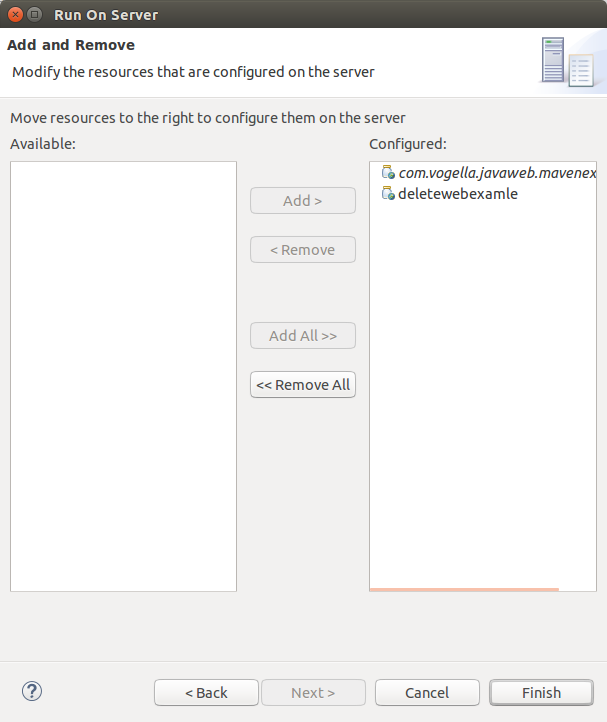
Similar to [Run the build](http://www.vogella.com/tutorials/EclipseMaven/article.html#example_eclipsemavenproject_runningthebuild) run your mvn clean verify build command from Eclipse. Validate that there are no issues with the build.

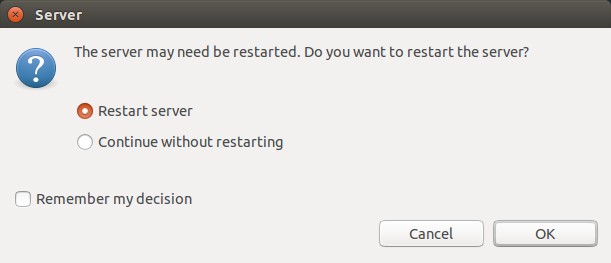
6.3. Run on the server

Right-click your project and select the Run As ▸ Run on Server menu entry.

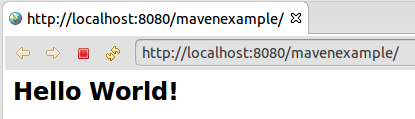








If you open a browser you should be able to access your webapplication.



7. References for Webdevelopment with Eclipse

To use Maven in Eclipse for Java web development, you should also install an configure the Eclipse web development tools (WTP). See Eclipse Web Development Tools (<http://www.vogella.com/tutorials/EclipseWTP/article.html#installation>) for its installation and usage.

8. Eclipse Maven (m2e) resources

<http://books.sonatype.com/m2eclipse-book/reference/> - m2e book

<http://takari.io/2015/03/19/core-extensions.html> - New Maven 3.3.1 Features: Core Extensions