Geospatial for Java

FOSS4G 2009 Eclipse Quickstart

23 October 2009

Jody Garnett
Michael Bedward

page intentionally left blank

Table of Contents

1 Welcome Eclipse Developers

Welcome to Geospatial for Java -this workbook is aimed at Java developers who are new to geospatial and would like to get started.

We are going to start out carefully with the steps needed to set up your IDE and are pleased this year to cover both NetBeans and Eclipse. If you are comfortable with the build tool Maven, it is our preferred option for downloading and managing jars but we will also document how to set up things by hand.

Extra care has been taken to make this year's tutorial visually oriented right from the get go. While these examples will make use of Swing, please be assured that that this is only an aid in making the examples easy and fun to use.

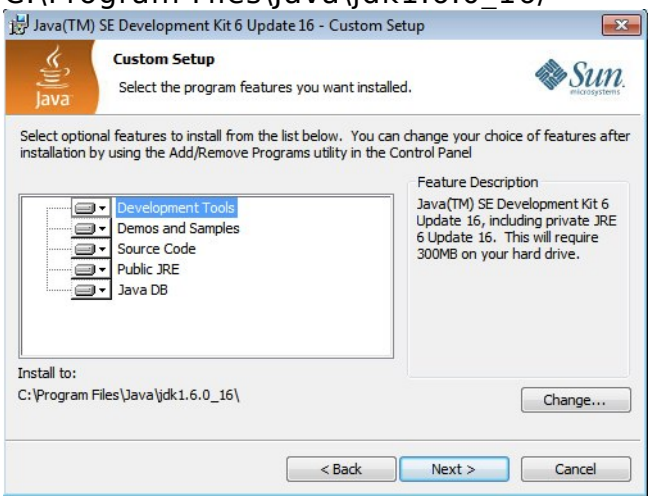
These sessions are applicable to both server side and client side development.

2 Java Install

We are going to be making use of Java – so if you don't have a Java Development Kit installed now is the time to do so. Even if you have Java installed already check out the optional Java Advanced Imaging and Java Image IO section.

1. Download the latest JDK from the the java.sun.com website:
   http://java.sun.com/javase/downloads/index.jsp
2. At the time of writing the latest JDK was:
   jdk-6u16-windows-i586.exe
3. Click through the installer you will need to set an acceptance a license agreement and so forth. By default this will install to:
   C:\Program Files\Java\jdk1.6.0_16/

If you are following this workbook in a lab setting you will find installer on the DVD.



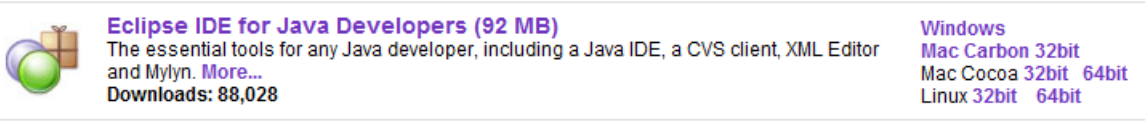
4. Optional – Java Advanced Imaging is used by GeoTools for raster support. If you install JAI 1.1.3 performance will be improved:
   https://jai.dev.java.net/binary-builds.html
   Both a JDK and JRE installer are available:
   jai-1_1_3-lib-windows-i586-jdk.exe
   jai-1_1_3-lib-windows-i586-jre.exe
5. Optional – ImageIO Is used to read and write raster files. GeoTools uses version 1_1 of the ImageIO library:
   https://jai-imageio.dev.java.net/binary-builds.html
   Both a JDK and JRE installer are available:
   jai_imageio-1_1-lib-windows-i586-jdk.exe
   jai_imageio-1_1-lib-windows-i586-jre.exe

3 Eclipse

Eclipse is a popular integrated development environment most often used for all kinds of Java development. For this tutorial we are doing straight up Java programming using the smallest download available - if you already have an Eclipse download please go ahead and use it and switch to the "Java Perspective".

Choose a local mirror – or give up and use Amazon Web Services if you are not sure.

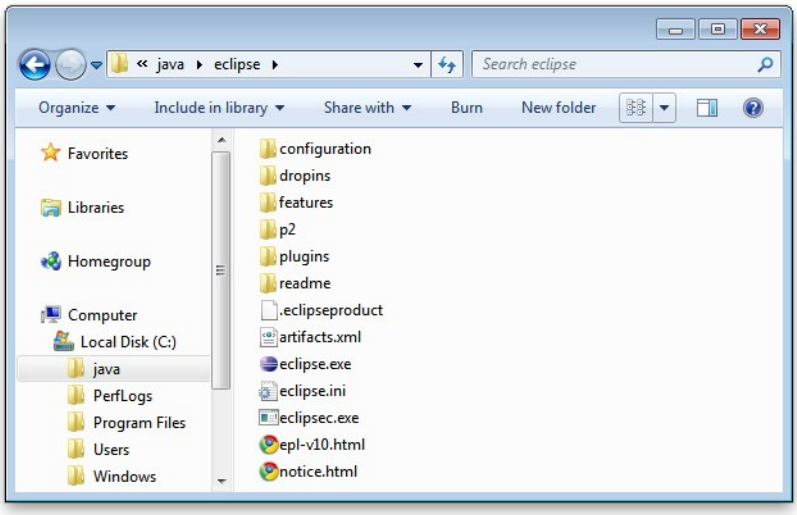
1. Visit the Eclipse download page (http://www.eclipse.org/downloads/) and download "Eclipse IDE for Java developers".



These instructions were written with the Eclipse Galileo 3.5.1 release.

2. Hopefully by now your eclipse download has finished and we can begin to installation.
3. Eclipse does not provide an installer; just a directory to unzip and run.
4. To start out with create the folder C:\java to keep all our java development in one spot.
5. Unzip the downloaded **eclipse-java-galileo-SR1-win32.zip** file to your C:\java directory – the folder C:\java\eclipse will be created.

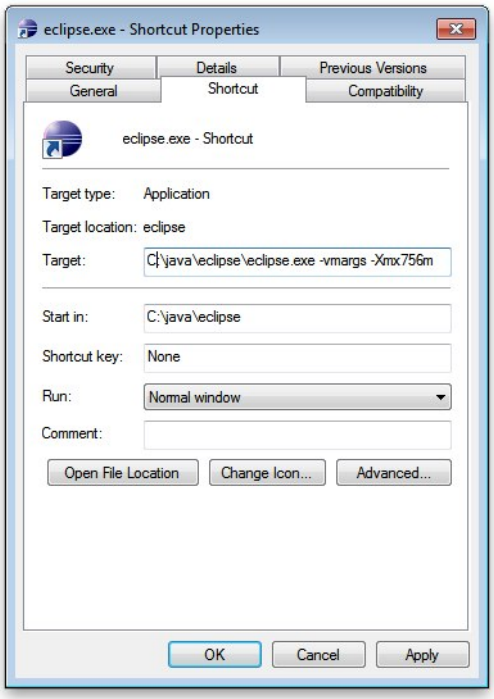
*If you need a good program to unzip archive files try: **www.7-zip.org***



6. Navigate to C:\java\eclipse and right-click on the eclipse.exe file and select Send To->Desktop (create shortcut).

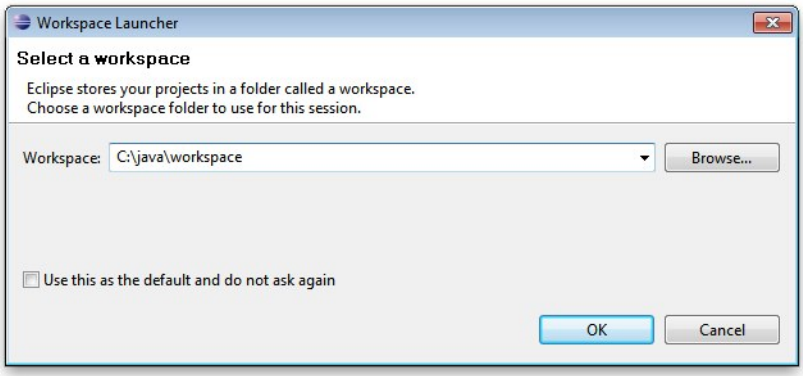
If you are on a linux or mac osx machine you may want to modify the "eclipse.ini" file to specify additional command line options.

7. Open up the desktop short cut properties and change the Target: C:\java\eclipse\eclipse.exe -vmargs -Xmx756m



If you have plenty of memory to burn on development you may wish to provide yourself some more memory.

8. Double click on your desktop short cut to start up eclipse.
9. When you start up eclipse for the first time it will prompt you for a workspace. To keep our java work in one spot you can type in: C:\java\workspace



10. On the Welcome view press Workbench along the right hand side.

11. We are now ready with an interesting choice...

There are two paths ahead – if you don't decide you will be eaten by a grue.

A "grue" is from the early text game Zork. Trust us, you don't want to be eaten by a grue.

- Maven Eclipse Plugin

  The Maven tool is intended to describe a project; rather then simply list the steps to build it. Part of that description is a list of the jars the project will use and a repository on the internet where the jars can be downloaded from.

  The maven command line tool download exactly what we need and set up our project for us.

- Adding Jars to your Project

  The GeoTools project provides a single download with all the needed jars. We can download this rather large file and unzip it into our project.

We recommend the use of maven; the single download is 40 megs in size and contains way more functionality then is needed.

3.1 Maven Eclipse Plugin

We are going to use maven as a command line tool (there is an IDE Plugin called M2Eclipse but I could not get it to work).

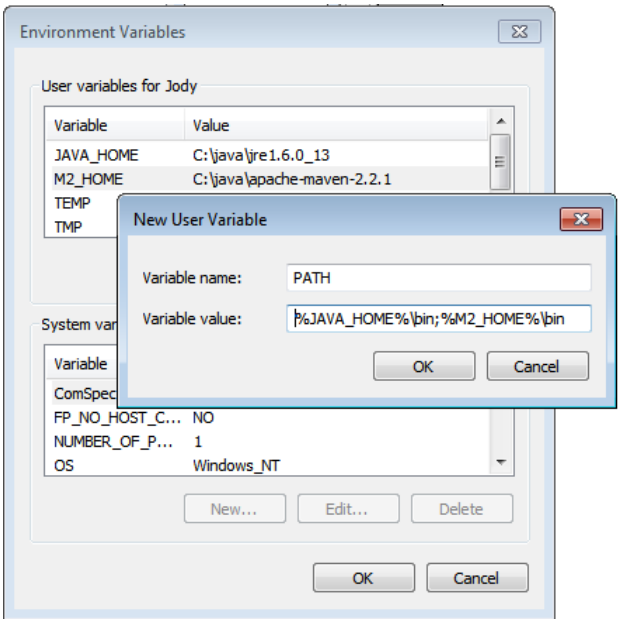
1. Download Maven from http://maven.apache.org/download.html

   The last version we tested with was: Maven 2.2.1

2. Unzip the file apache-maven-2.2.1-bin.zip to C:\java\apache-maven-2.2.1

3. You need to have a couple of environmental variables set for maven to work. Use Control Panel > System > Advanced > Environmental Variables to set the following:

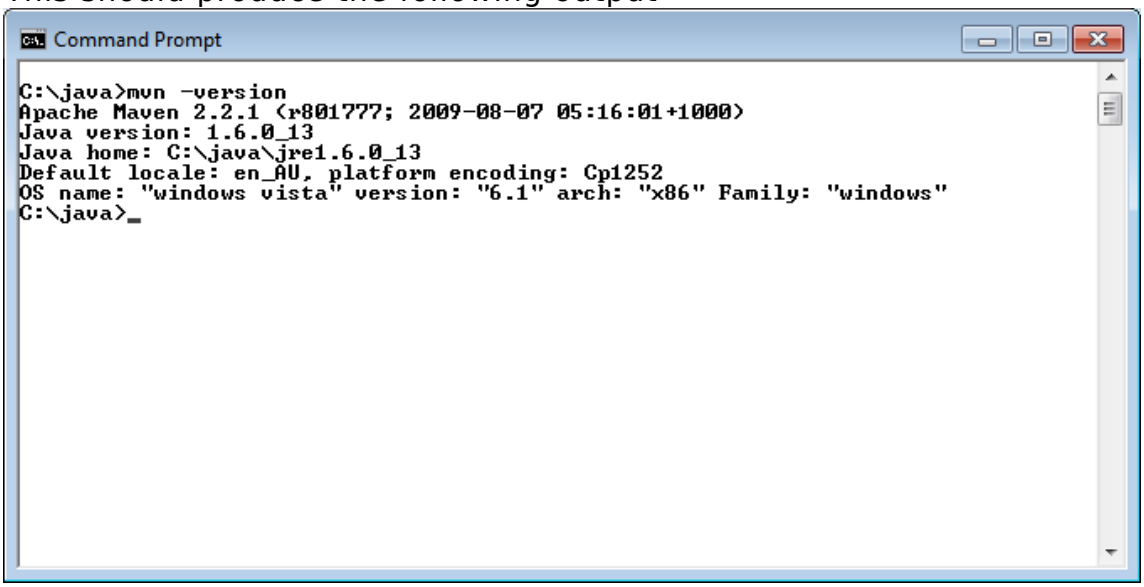
   JAVA_HOME=C:\Program Files\Java\jdk1.6.0_16/
   M2_HOME=C:\java\apache-maven-2.2.1
   PATH=%JAVA_HOME%\bin;%M2_HOME%\bin



4. Open up a commands prompt Accessories > Command Prompt
5. Type the following command to confirm you are set up correctly:

```
C:java> mvn -version
```

6. This should produce the following output



7. We can now create our project with:

```
C:>cd C:\java
C:java> mvn archetype:create -DgroupId=org.geotools.demo -DartifactId=example
```

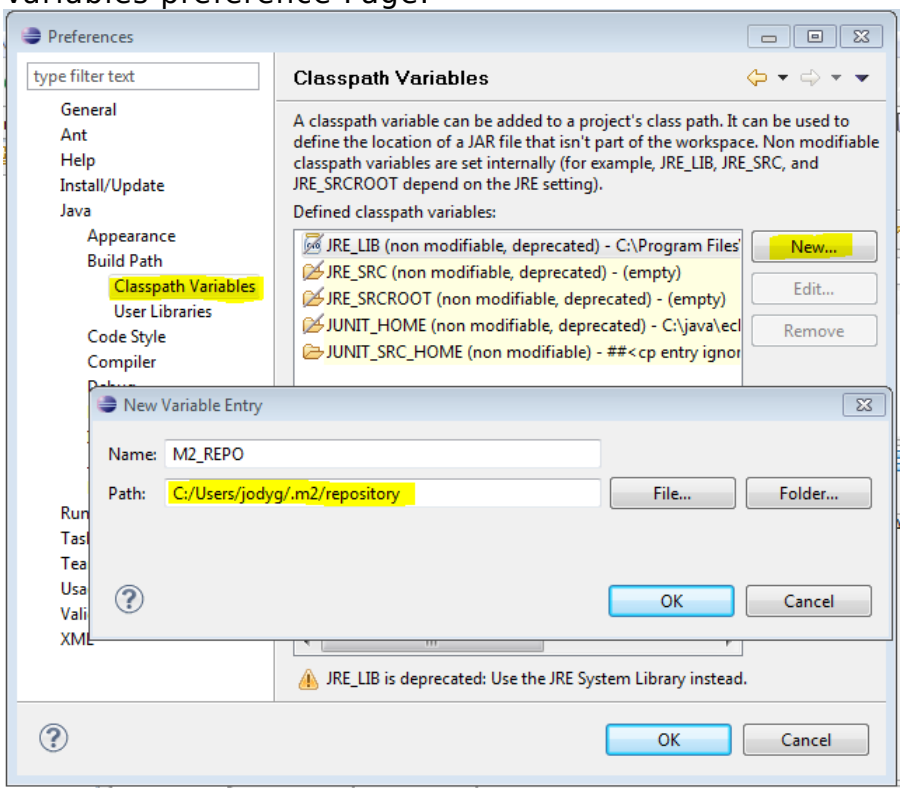
8. And ask for our project to be set up for eclipse:

```
C:java> cd example
C:java\example> mvn eclipse:eclipse
```

9. You can now give Eclipse the background information it needs to talk to your “maven repository” (maven downloaded something like 30 jars for you)

10. Return to Eclipse

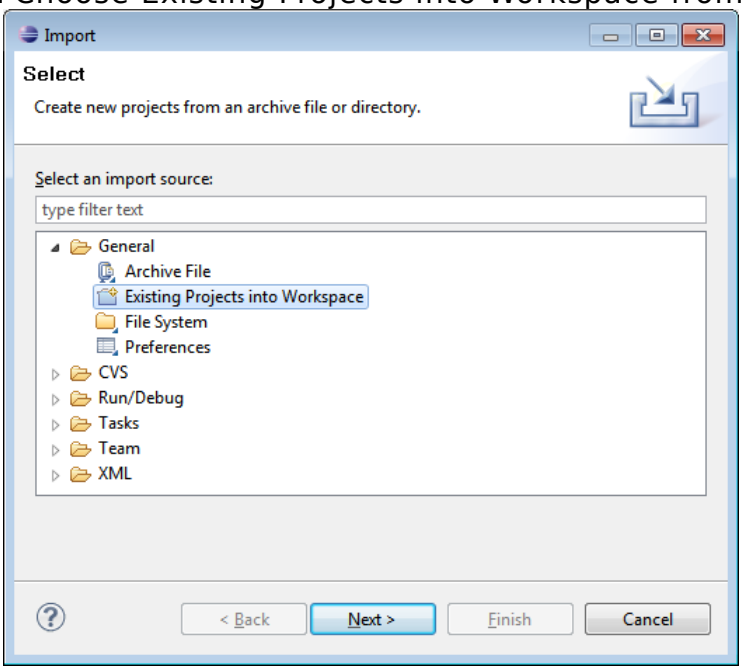
11. Use the Windows > Preferences menu to open the Preference Dialog. Using the tree on the left navigate to the Java > Build path > Classpath Variables preference Page.



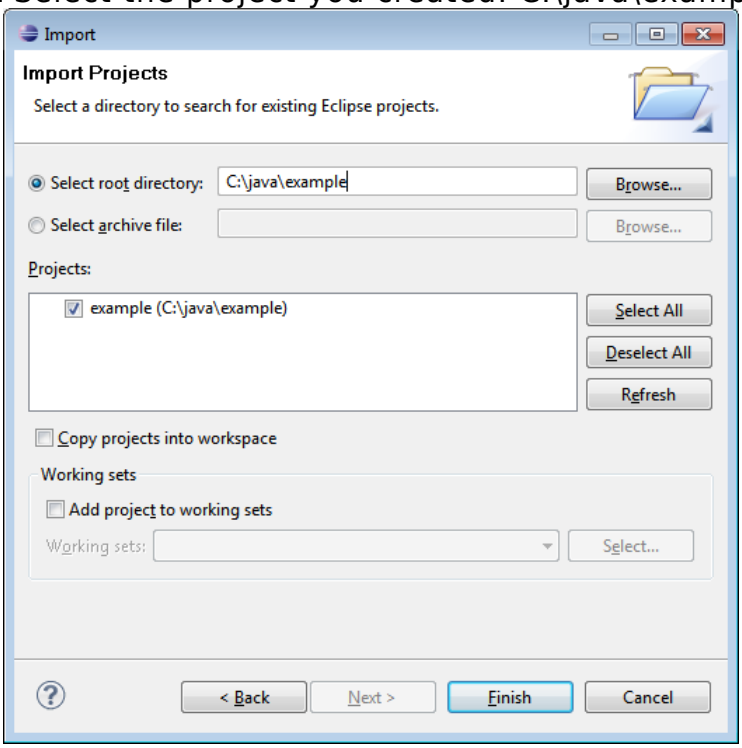
12. Add an M2_REPO classpath variable pointing to your “local repository”

- Windows XP: C:\Documents and Settings\Jody\.m2\repository
- Windows Vista: C:\Users\Jody\.m2\repository
- Linux and Mac: ~/.m2/repository

13. We can now import your new project into eclipse using File > Import
14. Choose Existing Projects into Workspace from the list, and press Next



15. Select the project you created: C:\java\example.



16. Press Finish to import your project

17. Navigate to the pom.xml file and double click to open it up.

The pom.xml file describes your project repositories to download jars from.

We are going to start by defining the version number of GeoTools we wish to use. This workbook was written for 2.6-RC you may wish to try a newer version – or make use of a nightly build by using 2.6-SNAPSHOT.

Please add in the sections marked in bold below. You may find cutting and pasting from the PDF easier then typing. Please be careful to cut and paste the sections in bold to the correct location.

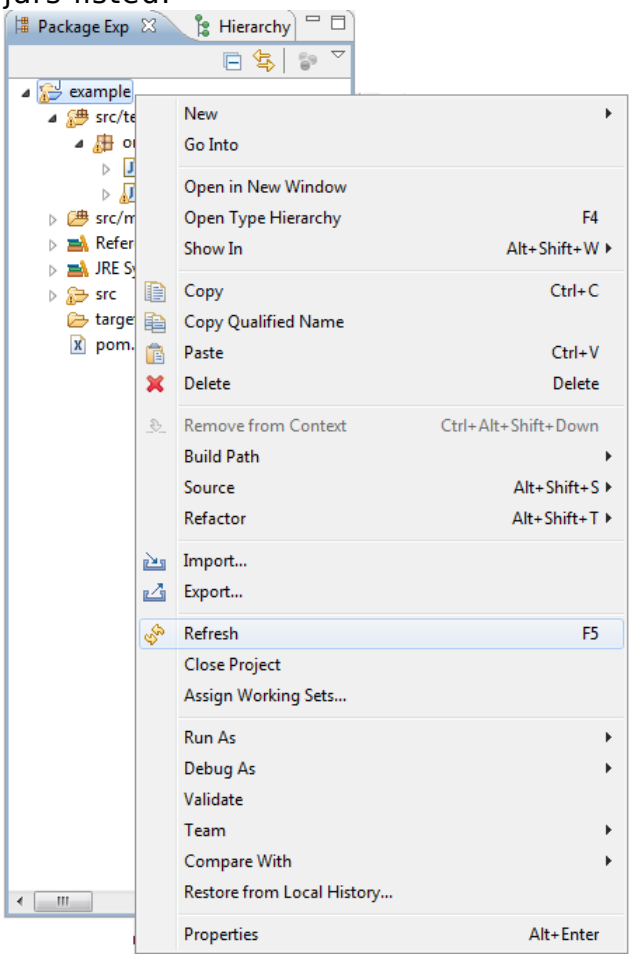
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <properties>
        <geotools.version>2.6.0</geotools.version>
    </properties>
    <groupId>org.geotools</groupId>
    <artifactId>example</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>example</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-shapefile</artifactId>
            <version>${geotools.version}</version>
        </dependency>
        <dependency>
            <groupId>org.geotools</groupId>
            <artifactId>gt-swing</artifactId>
            <version>${geotools.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>org.apache.xmlgraphics</groupId>
                    <artifactId>batik-transcoder</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
    <repositories>
        <repository>
            <id>maven2-repository.dev.java.net</id>
            <name>Java.net repository</name>
            <url>http://download.java.net/maven/2</url>
        </repository>
        <repository>
            <id>osgeo</id>
            <name>Open Source Geospatial Foundation Repository</name>
            <url>http://download.osgeo.org/webdav/geotools/</url>
        </repository>
        <repository>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
            <id>opengeo</id>
            <name>OpenGeo Maven Repository</name>
            <url>http://repo.opengeo.org</url>
        </repository>
    </repositories>
</project>
```

And easy way to pick up typing mistakes with tags is to Eclipse to format the xml file.

18. Return to the command line and maven to download the required jars and tell eclipse about it

```
C:\java\example> mvn eclipse:eclipse
```

19. Return to eclipse and select the project folder. Use F5 to refresh the project – if you open up referenced libraries you will see the required jars listed.



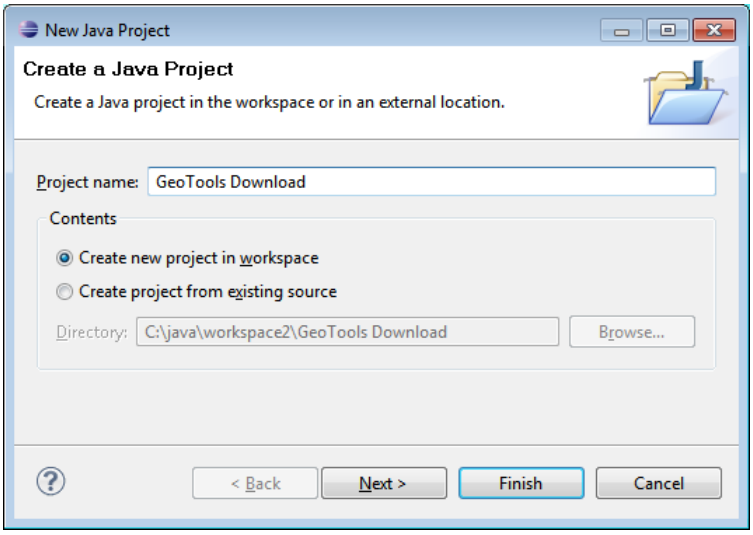
20. You can now skip to 4 Quickstart to try this stuff out.

3.2 Adding Jars to your Project

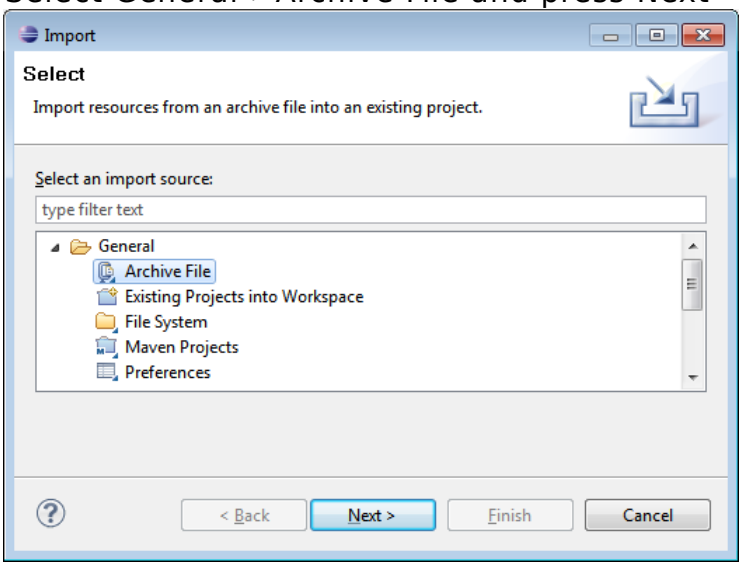
We can also download the GeoTools project bundle from source forge and set up our project to use them. Please follow these steps carefully as not all the GeoTools jars can be used at the same time.

1. Download the GeoTools binrary release from http://sourceforge.net/projects/geotools/files
2. We are now going to make a project for the required jars. By placing the jars into their own project is is easier to upgrade GeoTools.

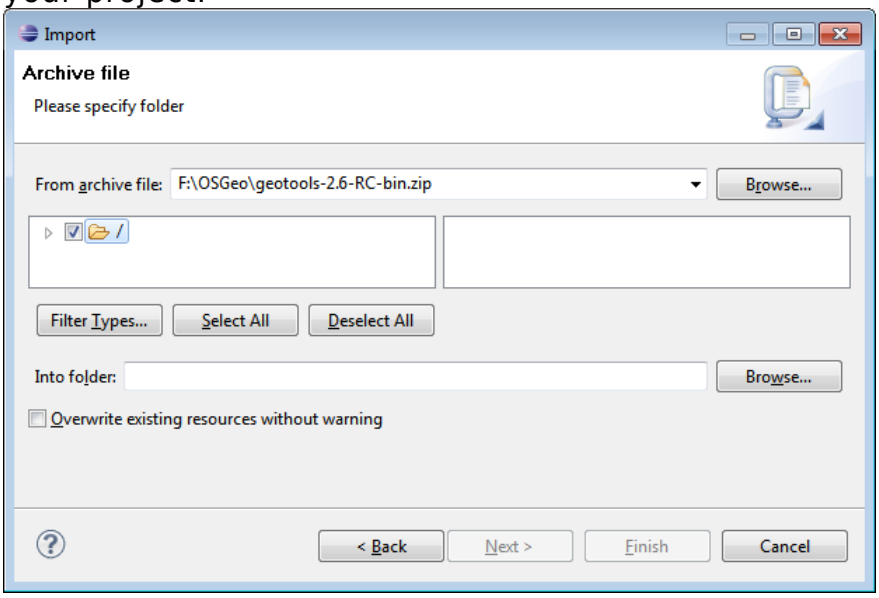
   Select File > New > Java Project to open the New Java Project wizard
3. Type in “GeoTools Download” as the name of the project and press Finish.



4. Choose File > Import to open the Import Wizard.
5. Select General >Archive File and press Next



6. Navigate to the geotools-bin.zip download and import the contents into your project.



7. GeoTools includes a copy of the “EPSG” database; but also allows you to hook up your own copy of the EPSG database as an option..

   However only one copy can be used at a time so we will need to remove the following jars from the Library Manager:

   gt-epsg-h2
   gt-epsg-oracle
   gt-epsg-postgresql
   gt-epsg-wkt

8. GeoTools allows you to work with many different databases; however to make them work you will need to download jdbc drivers from the manufacturer.

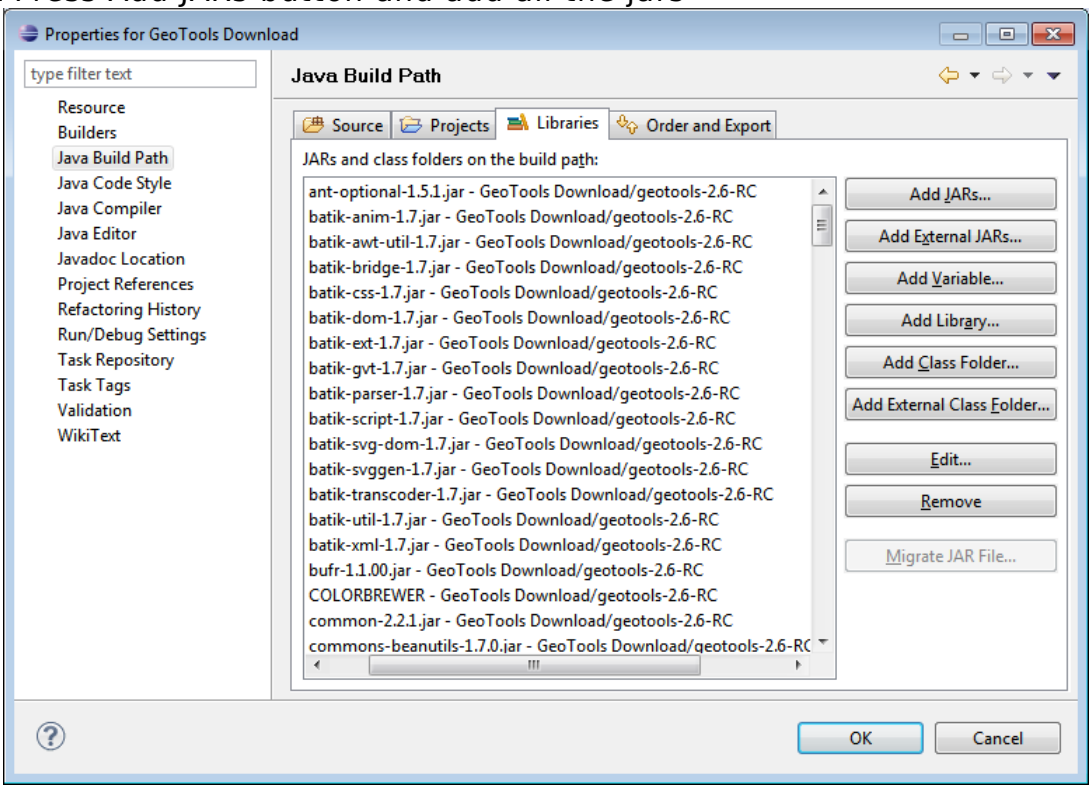
   For now remove the follow plugins from your Library Manager definition:

   gt-arcsde
   gt-arcsde-common
   gt-db2
   gt-jdbc-db2
   gt-oracle-spatial
   gt-jdbc-oracle

9. Next we update our java build path to include the remaining jars. Choose Project > Properties from the menu bar

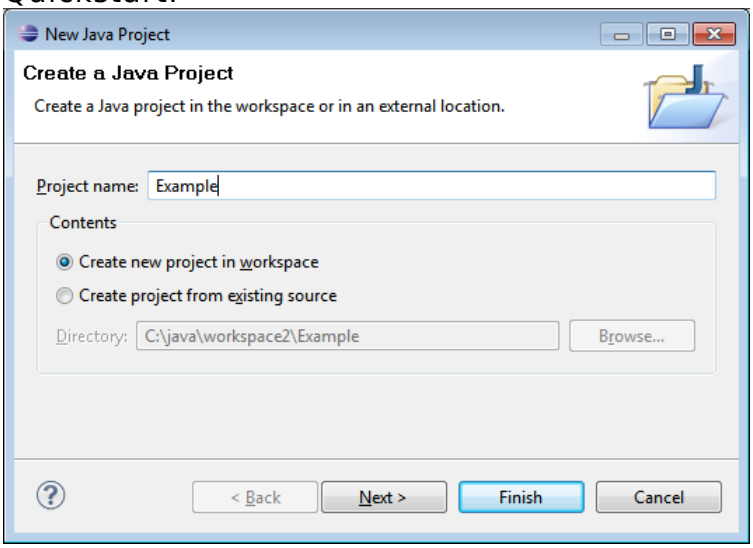
10. Select Java Build Path property page; and switch to the library tab.

11. Press Add JARs button and add all the jars



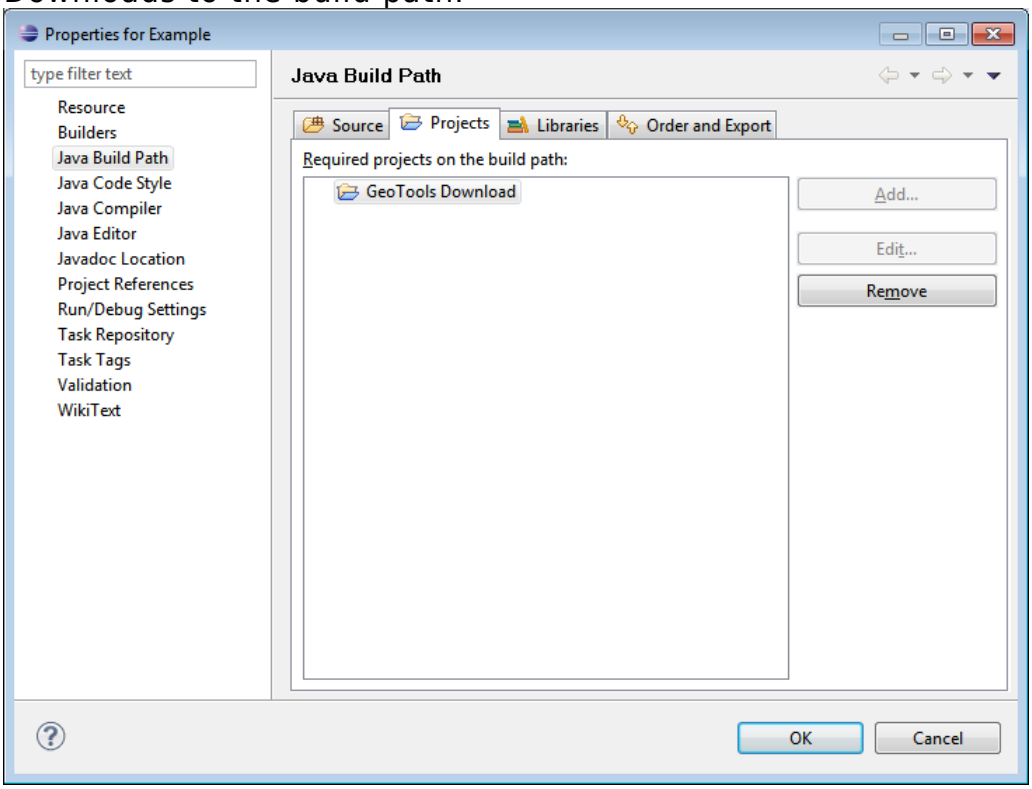
12. Switch to the Order and Export tab and press Select All

13. We can now create a new Example project to get going on our Quickstart.



14. Use Project > Properties on your new Example project to open up the Java Build Path page.

15. Switch to the Projects tab and use the Add.. button to add GeoTools Downloads to the build path.

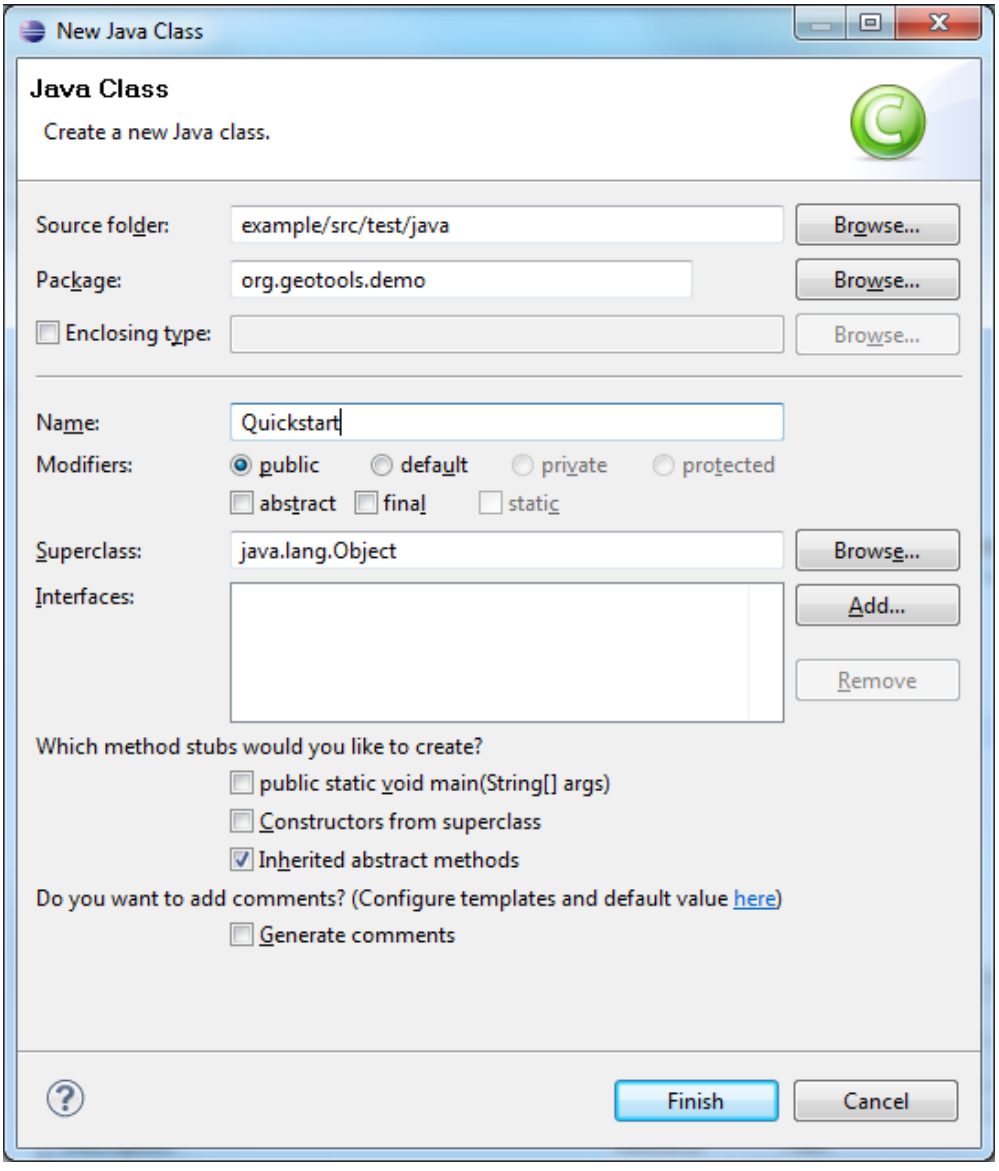


16. Our example project can now use all the GeoTools jars.
17. Please proceed to the Quickstart.

4 Quickstart

Now that your environment is setup we can put together a simple Quickstart. This example will display a shapefile on screen.

1. Create the org.geotools.demo.Quickstart class using your IDE.



2. Fill in the following code

```
package org.geotools.demo;

import java.io.File;

import org.geotools.data.FeatureSource;
import org.geotools.data.FileDataStore;
import org.geotools.data.FileDataStoreFinder;
import org.geotools.map.DefaultMapContext;
import org.geotools.map.MapContext;
import org.geotools.swing.JMapFrame;
import org.geotools.swing.data.JFileDataStoreChooser;

/**
 * GeoTools Quickstart demo application. Prompts the user for a shapefile
 * and displays its contents on the screen in a map frame
 */
public class Quickstart {

    /**
     * GeoTools Quickstart demo application. Prompts the user for a shapefile
     * and displays its contents on the screen in a map frame
     */
    public static void main(String[] args) throws Exception {
        // display a data store file chooser dialog for shapefiles
        File file = JFileDataStoreChooser.showOpenFile("shp", null);
        if (file == null) {
            return;
        }

        FileDataStore store = FileDataStoreFinder.getDataStore(file);
        FeatureSource featureSource = store.getFeatureSource();

        // Create a map context and add our shapefile to it
        MapContext map = new DefaultMapContext();
        map.addLayer(featureSource, null);

        // Now display the map
        JMapFrame.showMap(map);
    }
}
```

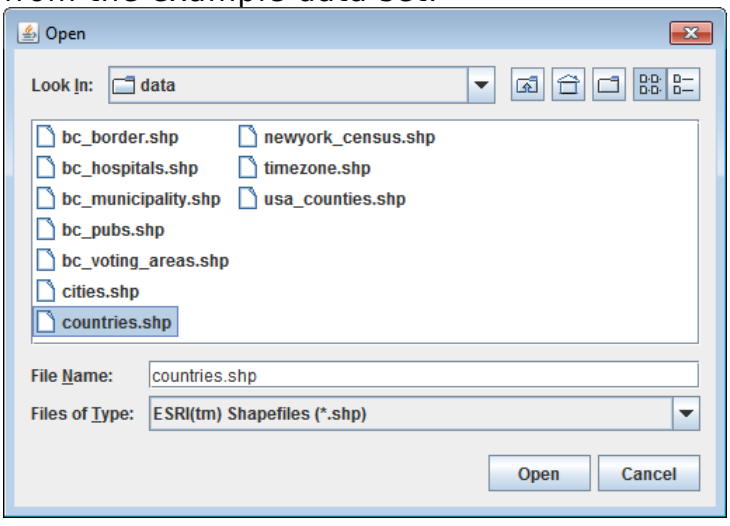
3. We need to download some sample data to work with. We are going to use some sample data provided with the uDig project (which is written with GeoTools).

   http://udig.refractions.net/docs/data-v1_2.zip

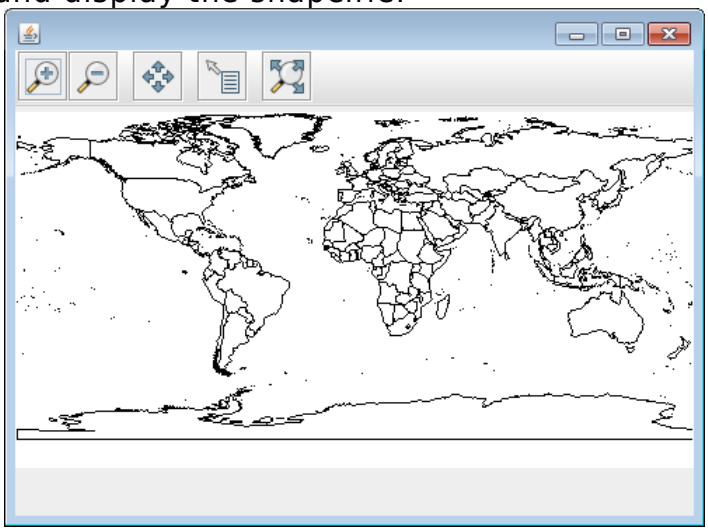
   Please unzip this data directory to a location you can find easily like your desktop.

If you need a good program to unzip archive files try: www.7-zip.org

4. Run the application to open up a file chooser. Please choose a shapefile from the example data set.



5. The application will connect to your shapefile, produce a map context and display the shapefile.



6. A couple of things to note about the code example:

- The shapefile is not loaded into memory – instead it is read from disk each and every time it is needed
- This approach allows you to work with data sets larger then available memory

5 Things to Try

Here are some additional challenges for you to try:

- Try out the different sample data sets
- You can zoom in, zoom out and show the full extents
- Use the select tool to examine individual countries in the sample countries.shp file
- Download the largest shapefile you can find and see how quickly it can be rendered. You should find that the very first time it will take a while as a spatial index is generated. After that performance should be very good when zoomed in.
- Try and sort out what all the different "side car" files are – and what they are for. The sample data set includes "shp", "dbf" and "shx". How many other side car files are there?
- The use of FileDataStoreFinder allows us to work easily with files. The other way to do things is with a map of connection parameters. This techniques gives us a little more control over how we work with a shapefile and also allows us to connect to databases and web feature servers.

```
        File file = JFileDataStoreChooser.showOpenFile("shp", null);

        Map<String,Object> params = new HashMap<String,Object>();
        params.put( ShapefileDataStoreFactory.URLP.key, file.toURI().toURL() );
        params.put( ShapefileDataStoreFactory.CREATE_SPATIAL_INDEX.key, false );
        params.put( ShapefileDataStoreFactory.MEMORY_MAPPED.key, false );
        params.put( ShapefileDataStoreFactory.DBFCHARSET.key, "ISO-8859-1" );

        DataStore store = DataStoreFinder.getDataStore( params );
        FeatureSource featureSource = store.getFeatureSource( store.getTypeNames()[0] );
```

- GeoTools is an active open source project – you can quickly use maven to try out the latest nightly build by changing your pom.xml file to use a "SNAPSHOT" release.

  At the time of writing 2.6-SNAPSHOT under active development.

```
    <properties>
        <geotools.version>2.6-SNAPSHOT</geotools.version>
    </properties>
```

- If you duck out to the command line you can ask maven to show you a list of all the required jars as a dependency tree.

```
C:\java\example> mvn dependency:tree
```

For me this produced the following tree with 2.6-SNAPSHOT

```
org.geotools:example:jar:1.0-SNAPSHOT
+- junit:junit:jar:3.8.1:test
+- org.geotools:gt-shapefile:jar:2.6-SNAPSHOT:compile
|  +- org.geotools:gt-main:jar:2.6-SNAPSHOT:compile
|  |  +- org.geotools:gt-api:jar:2.6-SNAPSHOT:compile
|  |  +- com.vividsolutions:jts:jar:1.10:compile
|  |  \- commons-beanutils:commons-beanutils:jar:1.7.0:compile
|  |     \- commons-logging:commons-logging:jar:1.0.3:compile
|  +- org.geotools:gt-referencing:jar:2.6-SNAPSHOT:compile
|  |  +- java3d:vecmath:jar:1.3.2:compile
|  |  +- commons-pool:commons-pool:jar:1.3:compile
|  |  \- org.geotools:gt-metadata:jar:2.6-SNAPSHOT:compile
|  |     +- org.opengis:geoapi:jar:2.3-M1:compile
|  |     +- org.opengis:geoapi-pending:jar:2.3-M1:compile
|  |     \- net.java.dev.jsr-275:jsr-275:jar:1.0-beta-2:compile
|  \- jdom:jdom:jar:1.0:compile
\- org.geotools:gt-swing:jar:2.6-SNAPSHOT:compile
   +- org.geotools:gt-render:jar:2.6-SNAPSHOT:compile
   |  +- org.geotools:gt-coverage:jar:2.6-SNAPSHOT:compile
   |  \- org.geotools:gt-cql:jar:2.6-SNAPSHOT:compile
   \- com.miglayout:miglayout:jar:swing:3.7:compile
```