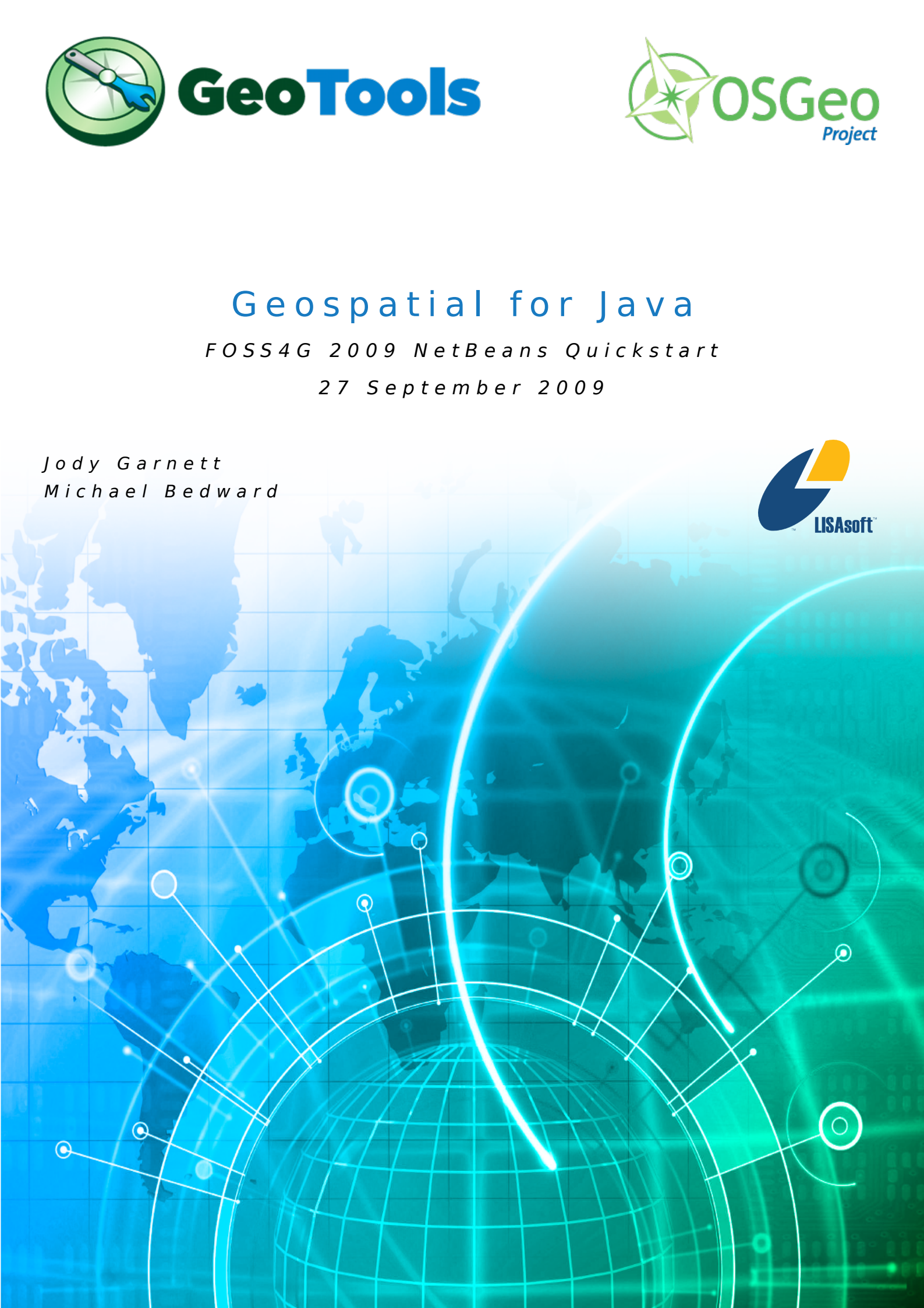


Geospatial for Java

FOSS4G 2009 NetBeans Quickstart

27 September 2009

Jody Garnett
Michael Bedward

page intentionally left blank

Table of Contents

1 Welcome NetBeans Developers..... 4

2 Java Install..... 5

3 NetBeans..... 6

3.1 Adding Dependencies using Maven..... 7

3.2 Adding Jars using Library Manager..... 11

4 Quickstart..... 15

5 Things to Try..... 17

1 Welcome NetBeans 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. The build tool Maven (http://maven.apache.org/) is our preferred option for downloading and managing jars for GeoTools projects because there tend to a large number of jars involved. If you are already familiar with Maven that is an advantage but if not, don't worry, we will be explaining things step by step and we will also document how to set up things by hand as an alternative to using Maven.

Extra care has been taken to make this year's tutorial visual 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 so if your own work is based on another GUI framework the material that we cover here will still be relevant.

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

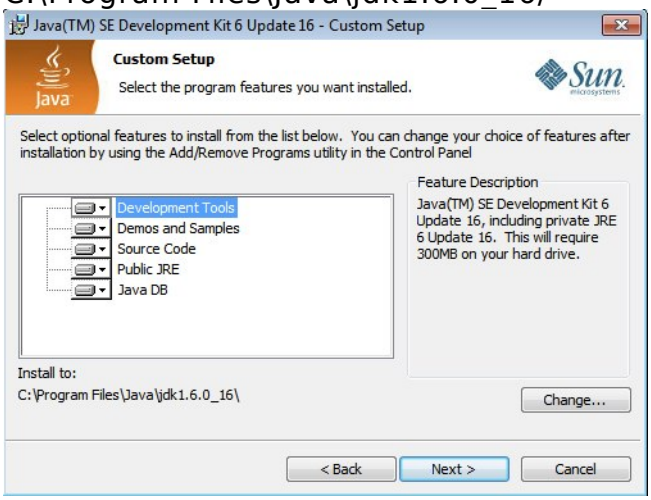
We would like thank members of the GeoTools users list for their feedback while were preparing the course material, with special thanks to Tom Williamson for reviewing early drafts.

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 – both of these libraries are used by GeoTools.

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 the installer on the DVD.

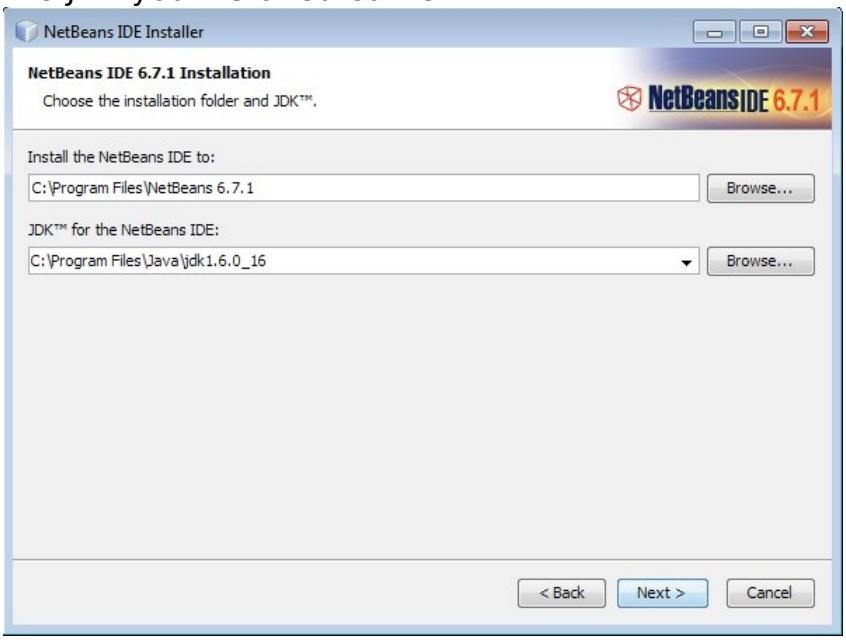


4. Optional – Java Advanced Imaging is used by GeoTools for raster support. If you install JAI 1.1.3 performance will be improved:
   https://jai.dev.java.net/binary-builds.html
   Both a JDK and JRE installer are available:
   jai-1_1_3-lib-windows-i586-jdk.exe
   jai-1_1_3-lib-windows-i586-jre.exe
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 NetBeans

The NetBeans IDE is a popular choice for Java development and features excellent Maven integration.

1. Download NetBeans (The Java SE download will be fine). http://www.netbeans.org/
2. At the time of writing nebeans-6.7.1-ml-javase-windows.exe was the latest installer.
3. Click through the steps of the installer. You will notice it will pick up on the JDK you installed earlier.



4. 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 one.

- Adding Dependencies using Maven

  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.

- Adding Jars using the NetBeans Library Manager

  NetBeans provides a "library manager" allowing you to gather up a group of jars by hand and set up your project to use them. In this case the GeoTools project provides a single download with all the needed jars.

Maven is recommended as it is built into Netbeans and allows fine grain control over what is downloaded (rather then wait for a 40 meg download).

3.1 Adding Dependencies using Maven

The GeoTools development community, for the most part, uses the build tool Maven to manage jars and their dependencies. Maven has been integrated into the latest releases of NetBeans.

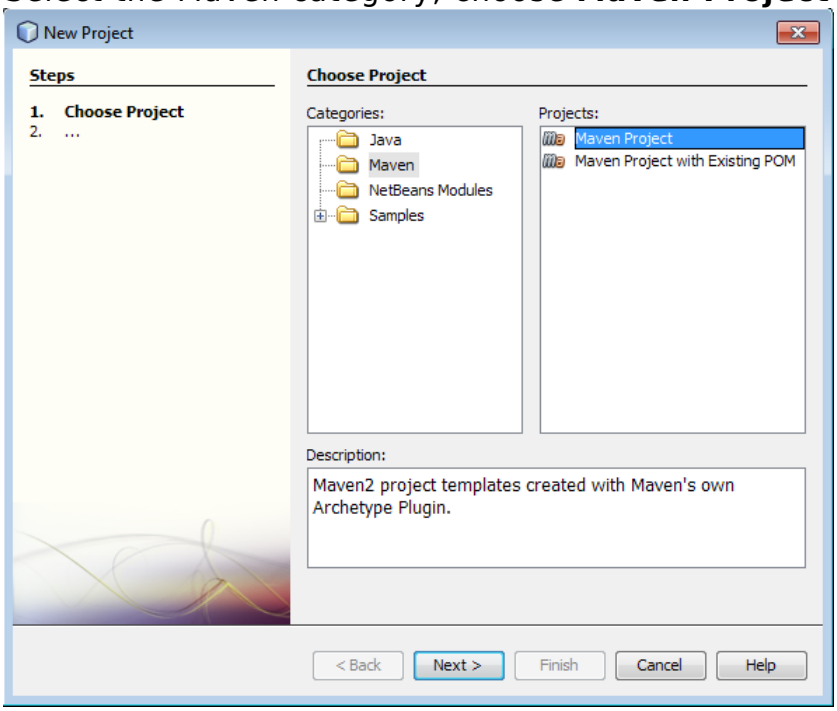
The advantages of using Maven are:

- You only download as much of GeoTools as your application requires
- Jars are downloaded to a single location in your home directory (in a hidden folder called .m2/repository)
- Source code and javadocs are automatically downloaded and hooked up

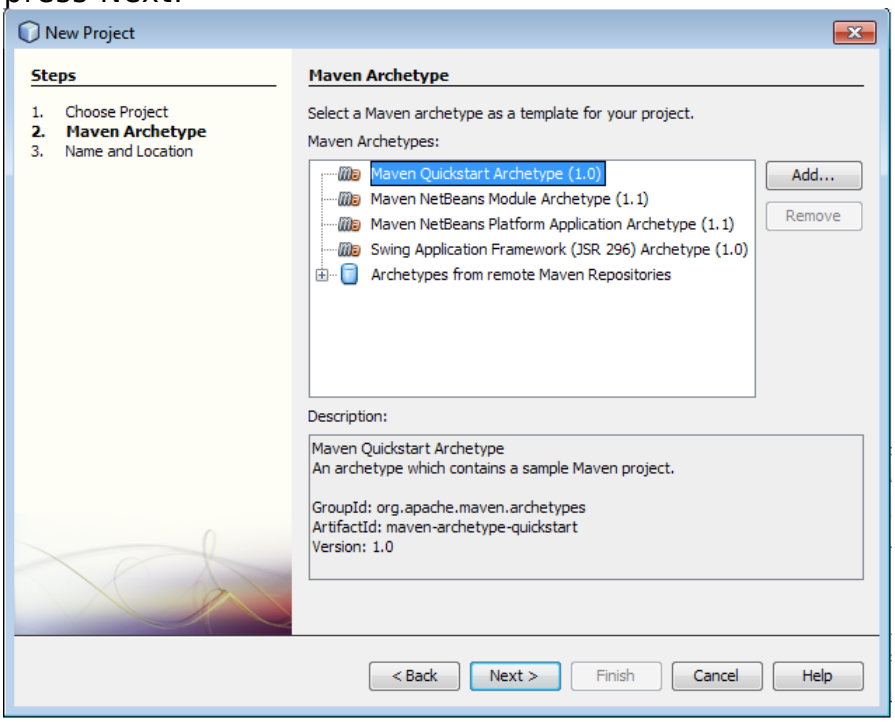
In a lab setting the instructor will provide you with a local repository to place in your home directory. This step will allow you to avoid the long download.

Let's get started:

1. Start with File > New Project to open the New Project wizard
2. Select the Maven category; choose **Maven Project** and press **Next**.

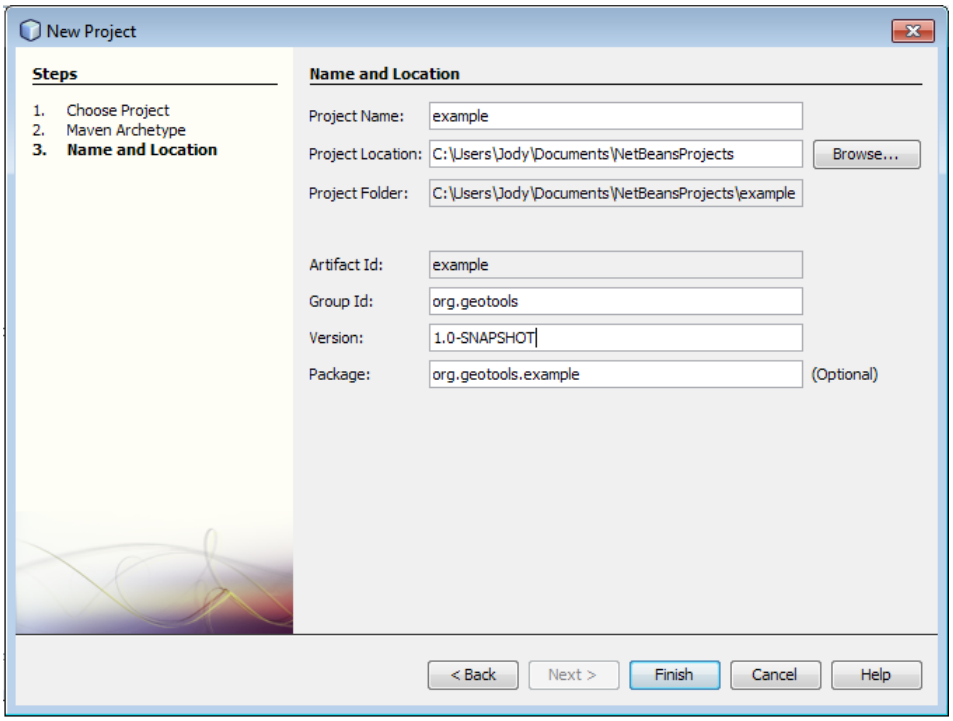


3. On the Maven Archetype page select “Maven Quickstart Archetype” and press Next.



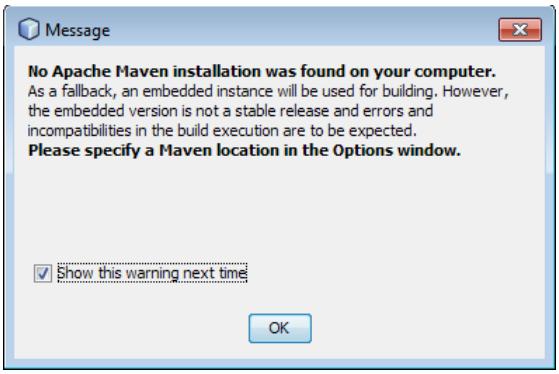
4. We can now fill in the blanks

   Project name: example
   GroupId: org.geotools



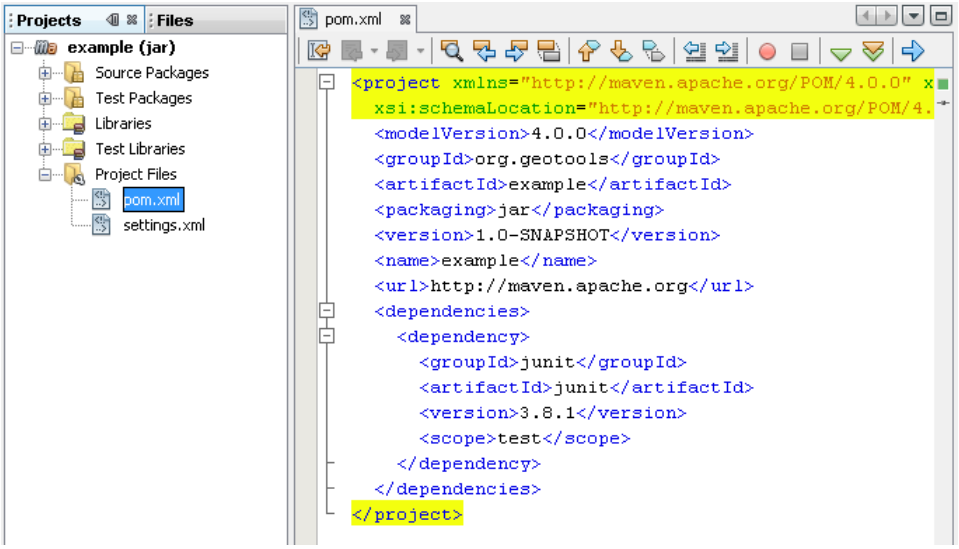
5. Click on the **Finish** button and the new project will be created.

6. If this is your first time using Maven with NetBeans it will want to confirm that it is okay to use the copy of Maven included with NetBeans (it is also possible to use an external Maven executable from within Netbeans which is convenient if, for instance, you want to work with the same version of Maven within the IDE and from the command line).



7. The next step is for us to make it a GeoTools project by adding information to Maven's project description file ("project object model" in Maven-speak) - pom.xml

   In the Projects panel open up the Project Files folder and double click on pom.xml to open it.



Cutting and pasting from the PDF may be easier then typing.

8. We are going to start by defining the version number of GeoTools we wish to use.

```
<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>
```

   If you make any mistakes when editing the xml file you'll see that your project will be renamed "**<Badley formed Maven project>**" in the Projects window. You can choose "Format" as a quick way to check if the tags line up. Or just hit undo and try again.

9. Next we add two GeoTools modules to the **dependencies** section: gt-shapefile and gt-swing for our project.

```
<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>
```

10. You can now right click on **Libraries** in the Projects window, then **Download missing Dependencies** from the pop-up menu. When downloading it will check the repositories we have listed above.

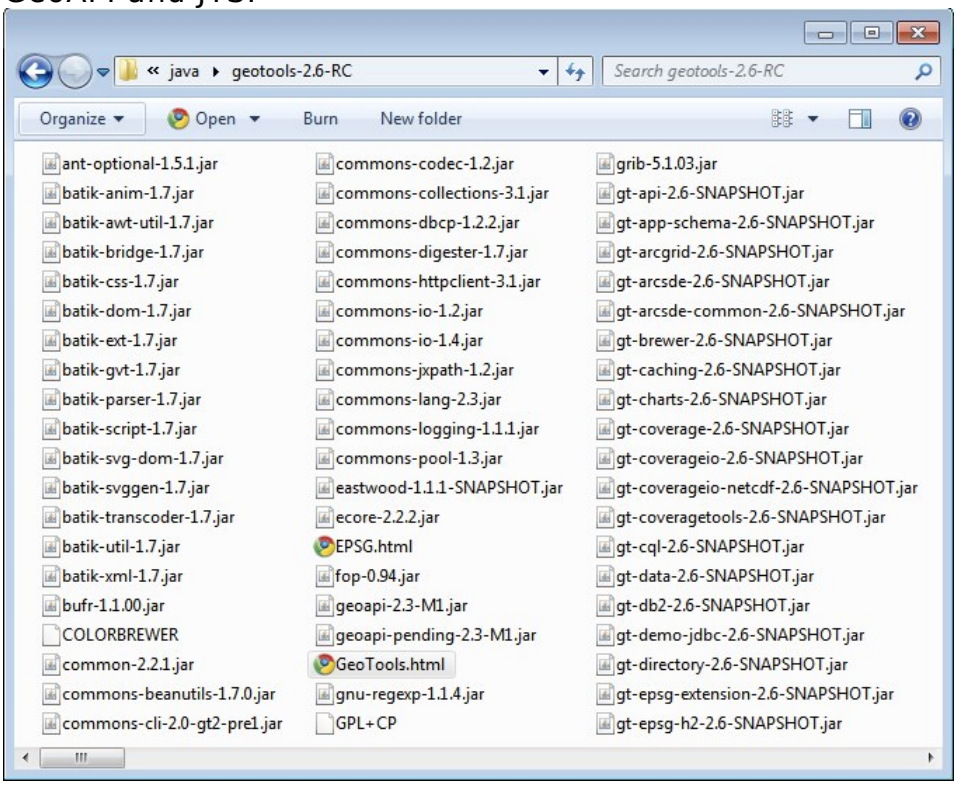
11. We will continue to add dependencies on different parts of the GeoTools library as we work through these exercises; this fine grain control and the ability to download exactly what is needed is one of the advantages of using Maven.

Congratulations you are now ready for the Quickstart!

3.2 Adding Jars using Library Manager

The alternative to using Maven to download and manage jars for you is to manually install them. To start with we will obtain GeoTools from the website:

1. Download the GeoTools binary release from http://sourceforge.net/projects/geotools/files
2. Extract the geotools-2.6.0-bin.zip file to C:\java\geotools-2.6.0 folder.
3. If you open up the folder and have a look you will see GeoTools and all of the other jars that it uses including those from other libraries such as GeoAPI and JTS.

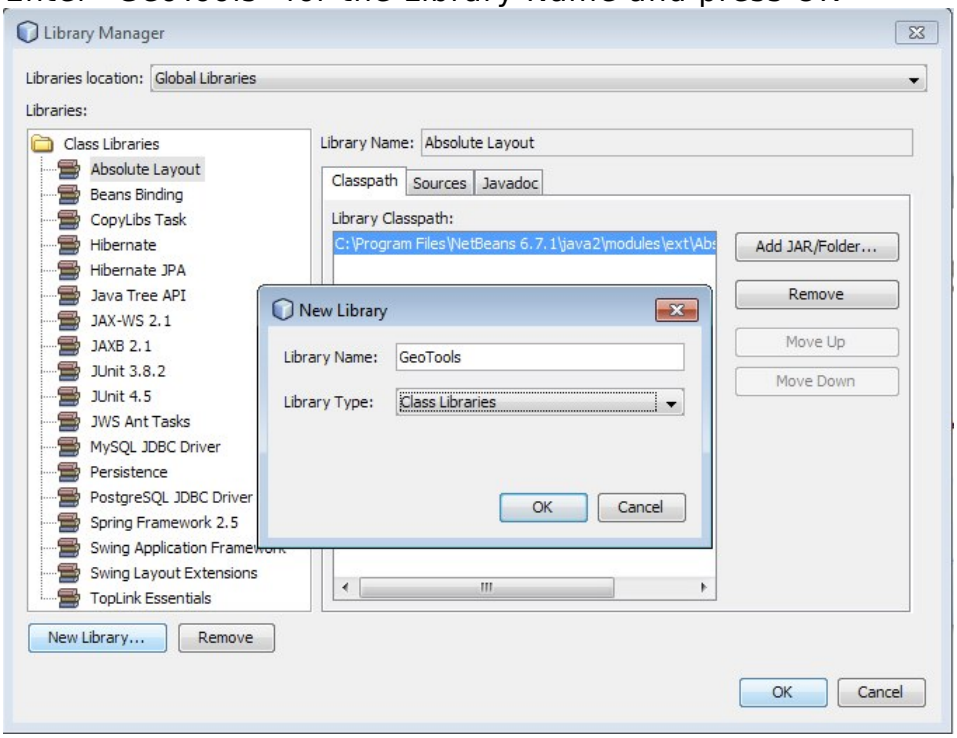


4. We can now set up GeoTools as a library in NetBeans:

   From the menu bar choose **Tools > Libraries** to open the Library Manager

5. From the Library Manager press the **New Library** button.

6. Enter “GeoTools” for the Library Name and press OK



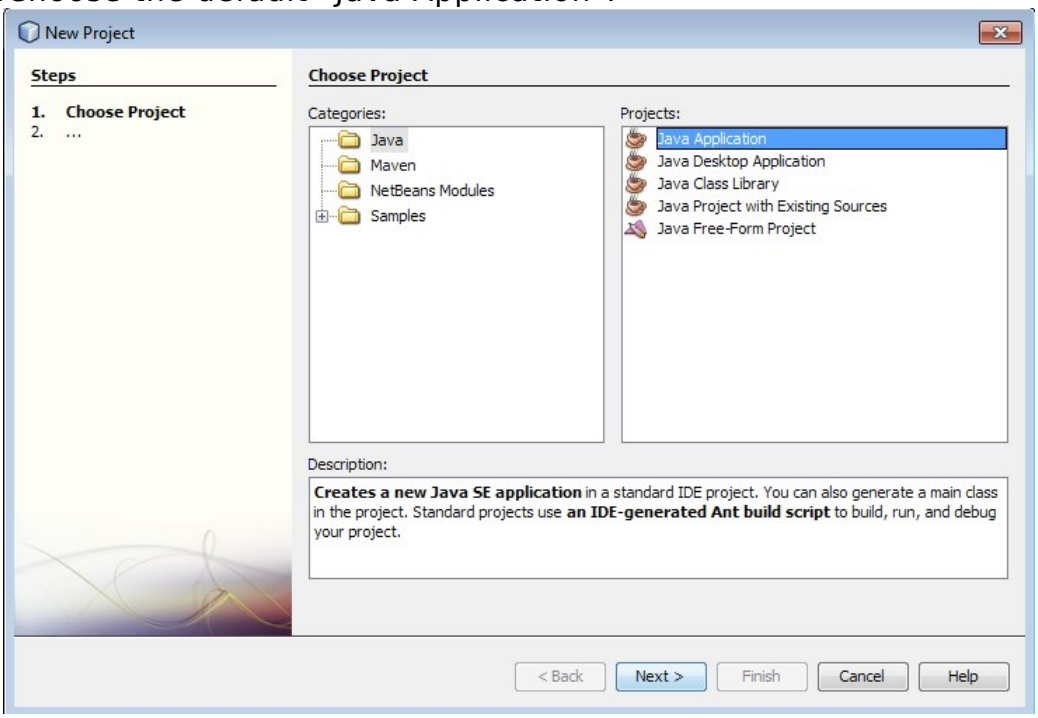
7. You can now press the Add JAR/Folder button and add in all the jars from C:\java\GeoTools-2.6.0

The EPSG databaes is distributed as an Access database and has been converted into the pure java database HSQL for our use.

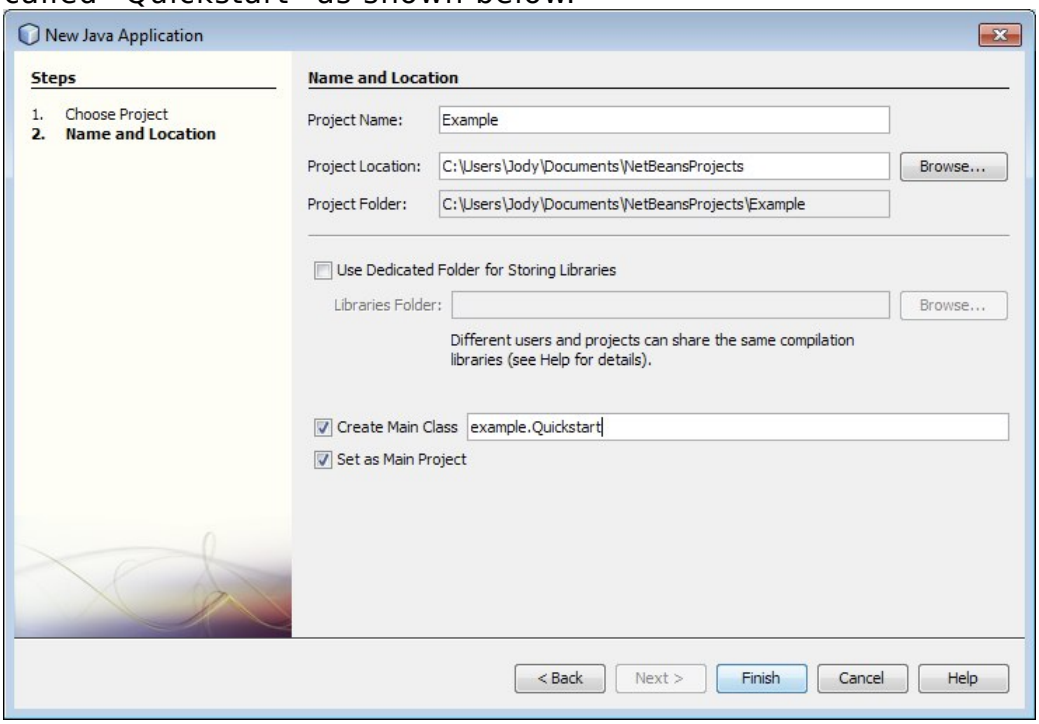
8. GeoTools includes a copy of the “EPSG” map projections database; but also allows you to hook up your own copy of the EPSG database as an option. However, only one copy can be used at a time so we will need to remove the following jars from the Library Manager:
   gt-epsg-h2
   gt-epsg-oracle
   gt-epsg-postgresql
   gt-epsg-wkt-2.6
9. 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 following plugins from the Library Manager:
   gt-arcsde
   gt-arcsde-common
   gt-db2
   gt-jdbc-db2
   gt-oracle-spatial
   gt-jdbc-oracle
10. We are now ready to proceed with creating an example project. Select **File > New Project**

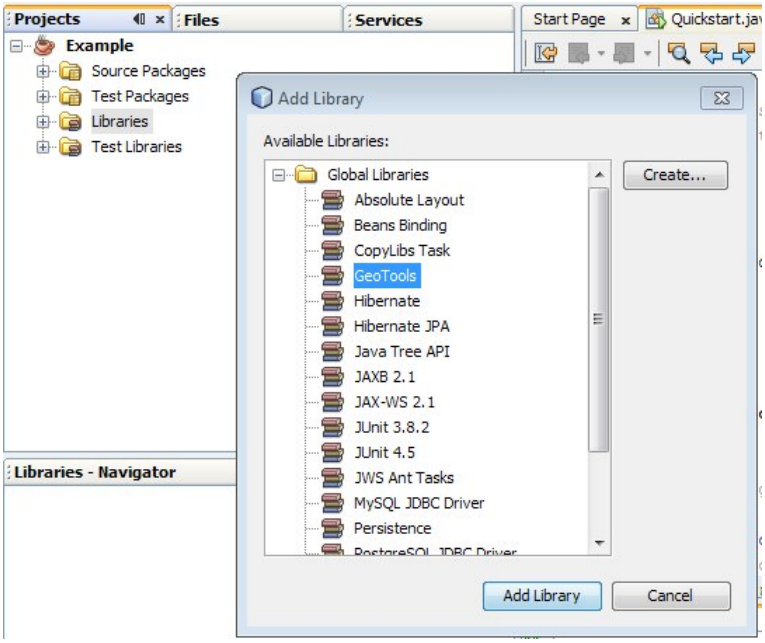
11. Choose the default “Java Application”.



12. Fill in “Example” as the project name; and our initial Main class will be called “Quickstart” as shown below.



13. Open up Example in the Projects window, right click on **Libraries** and select **Add Libraries.** Choose GeoTools from the Add Library dialog.



14. Congratulations ! You can now proceed to the Quickstart in the next section.

4 Quickstart

Now that your environment is set up 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

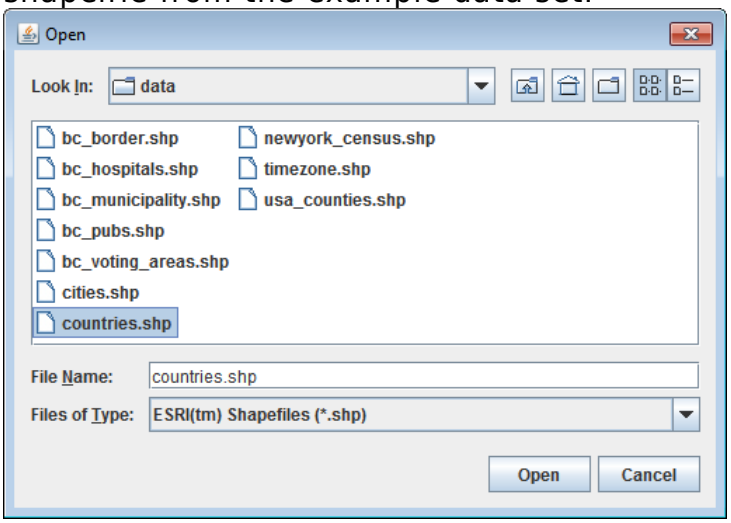
```
package org.geotools.demo;

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. Now build the application and check that all is well in the Output window (in Netbeans you use the same commands to build both a Maven project or a project with jars installed with the Library Manager).
4. 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).

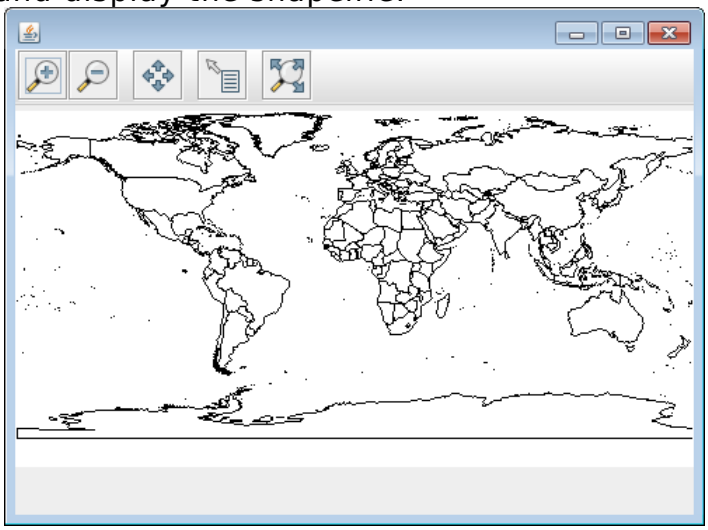
   http://udig.refractions.net/docs/data-v1_2.zip
5. Please unzip this data directory to a location you can find easily like your desktop.

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

6. Run the application. A file chooser will be displayed. Please choose a shapefile from the example data set.



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



8. Things to note about this 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.
   - We are using a very basic display style here that just shows feature outlines. In the examples that follow we will see how to specify more sophisticated styles.

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 a very 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, the active development version is 2.6-SNAPSHOT.

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

- NetBeans has an interesting feature to show how the dependency system works - Right click on Libraries and choose **Show Dependency Graph**.