PLUGIN TUTORIAL

Adding a Distance Tool

30 January 2008
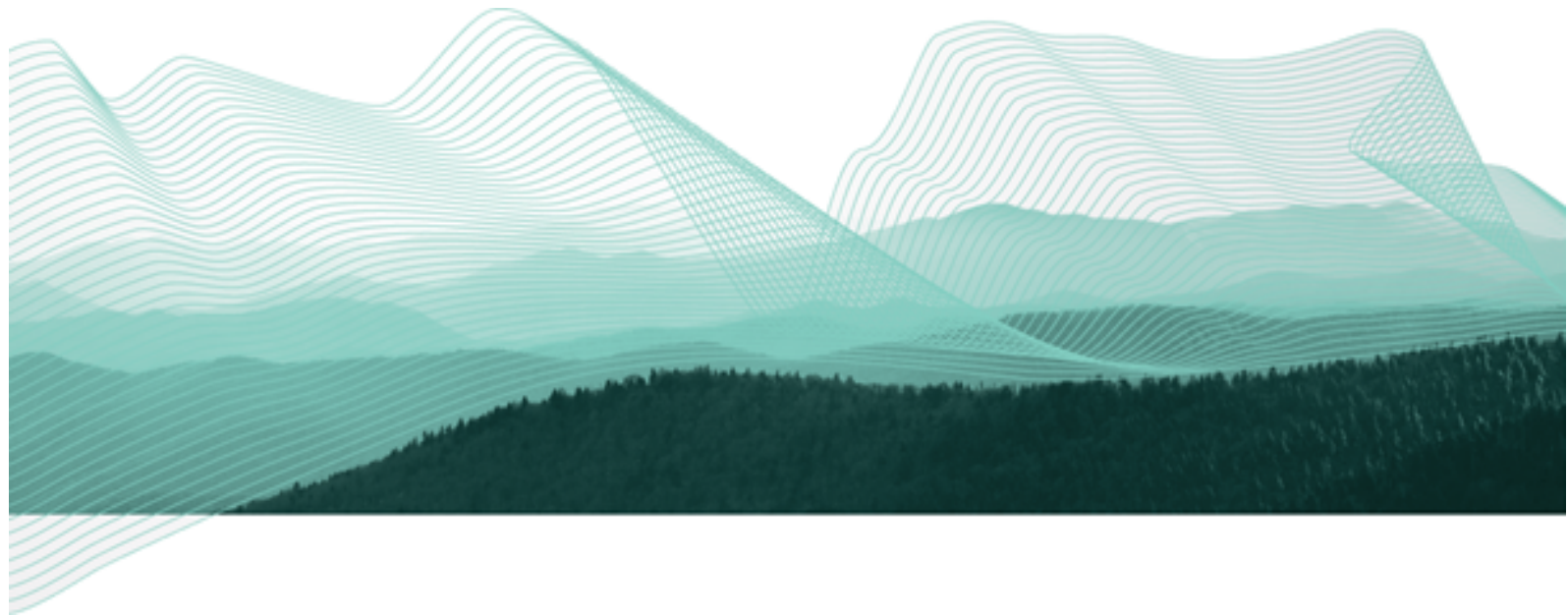
Refractions
RESEARCH
THE GEOSPATIAL EXPERTS

TABLE OF CONTENTS

1 GOALS

The uDig application is built on the "Eclipse Rich Client Platform" that offers an alternative to traditional application framework design. The RCP Platform is customized through **extensions** that contributed to **extensions points**. The good news is that everything is consistent, everything from adding a tool to creating an application is done in the same manner.

After completing this tutorial, you will have gained the skills to:

- Create a new Plugin
- Define a new Extension
- Implement a new Tool Extension
- Update the map from within a tool

We are going to create a tool that returns the distance between the point the mouse is clicked and the point the mouse is released.
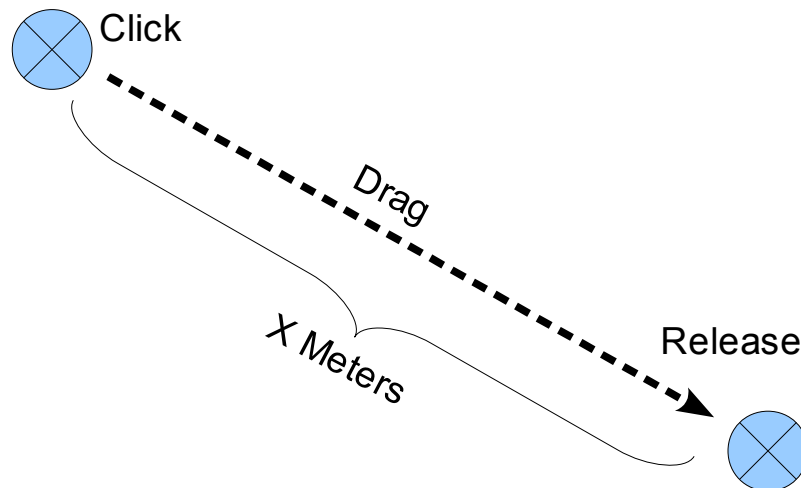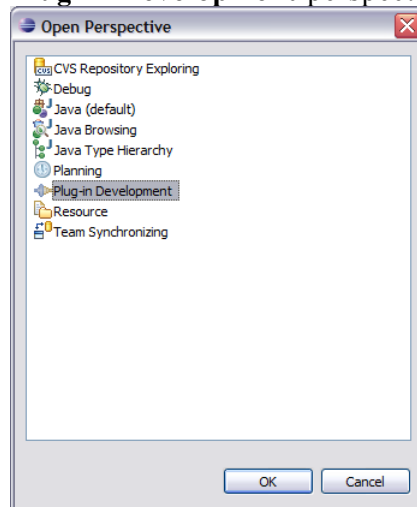
2 CREATE A NEW PLUGIN

In this section we are going to create a new plug-in. Eclipse plug-ins are managed bundles of code (often packaged up as a jar or folder).

Our focus in this section is on creating a plug-in and getting the name and version information correct. We will also provide a list of uDig plug-ins we need in order to make a good tool.

An Eclipse class called **Platform** is responsible for loading up our plug-in when it is needed. The Platform class will use the information we provide to make sure all our requirements are met. Plug-ins are loaded up into separate class loaders; and Java class loader restrictions are in place so you really can only talk to plug-ins you depend on!

1. Open up Eclipse using the workspace configured for uDig SDK development.

2. Select **Windows > Open Perspective > Other** and choose the **Plug-in Development** perspective from the list.

The Plug-in Perspective will contain a few Views and Editors you may not be familiar with from day to day Java programming.

3. Choose **File > New > Project...** from the menu bar.

4. Select **Plug-in Project** and press **Next**.
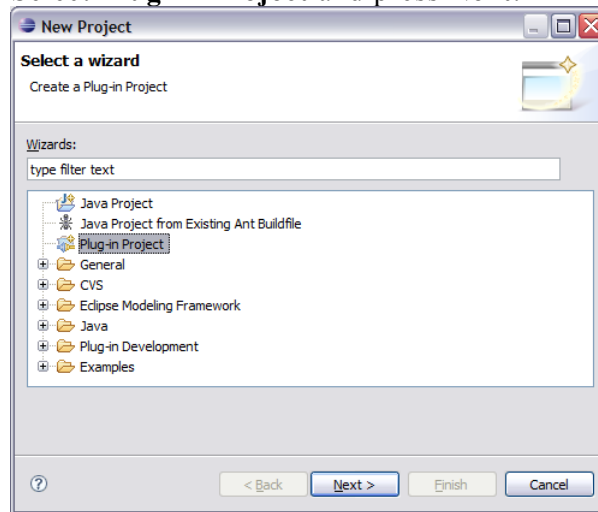
5. Create a name for the plug-in by entering
   Project Name: net.refractions.udig.tutorials.distancetool
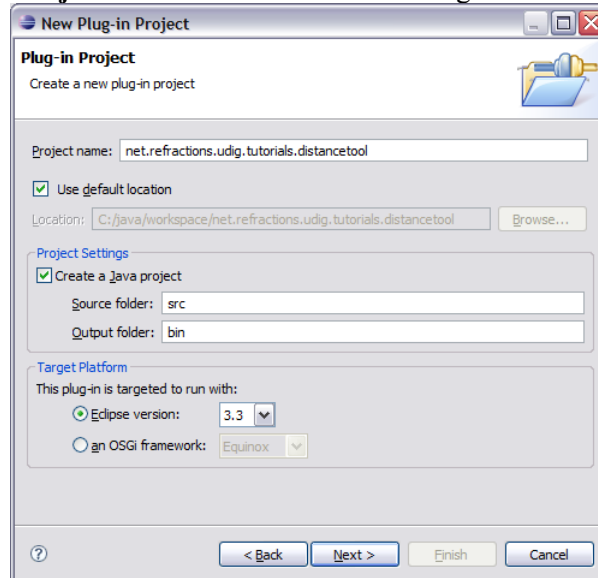
6. Press the **Next** Button

7. Accept the default values for Plug-in Content.

8. Press **Finish** to create your new project.

3 MANIFEST

Your new project is both: a plug-in project (with a MANIFEST.MF file for the Eclipse Platform class to read); and a Java project with .classpath file for the eclipse compiler to read.

Lets have a look at what information is MANIFEST.MF:

1. In the **Package Explorer** navigate to the plug-in created in the previous section. Find the file **META-INF/MANIFEST.MF** and double click to open the plug-in manifest editor.
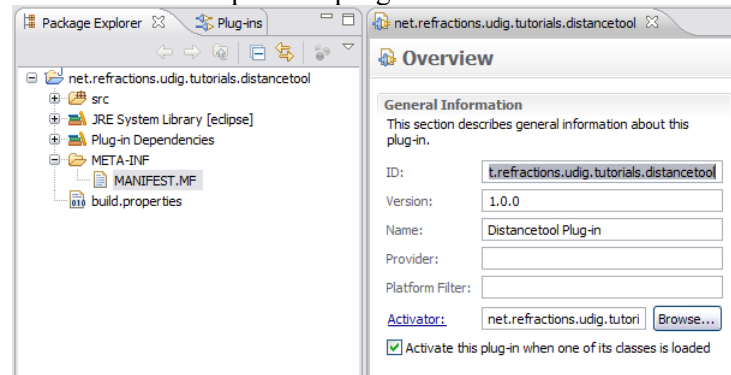
2. The overview tab shows much of the information you entered when you created the plug-in project.

3. Switch to the **Dependencies** tab.
    (the tabs are located at the bottom of the editor area)
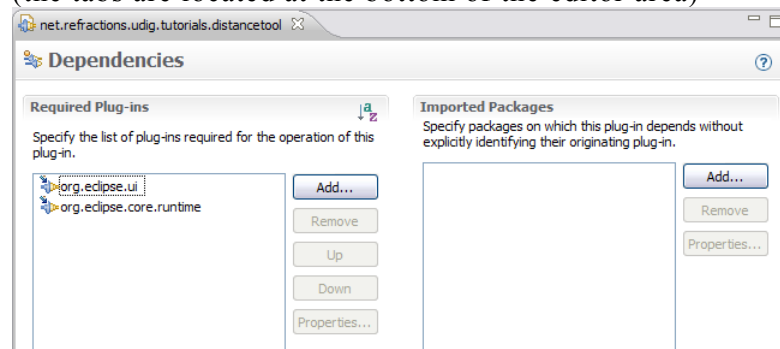
4. Click on the **Add** button under Required Plug-ins.

5. Select the **net.refractions.udig.project.ui** plug-in from the list.

Plug-in Selection

Select a Plug-in:

- net.refractions.udig.libs.source (1.1.0)
- net.refractions.udig.location (1.0.0)
- net.refractions.udig.mapgraphic (1.1.0)
- net.refractions.udig.printing.model (1.1.0)
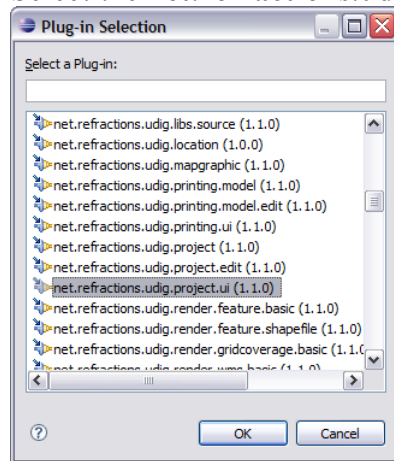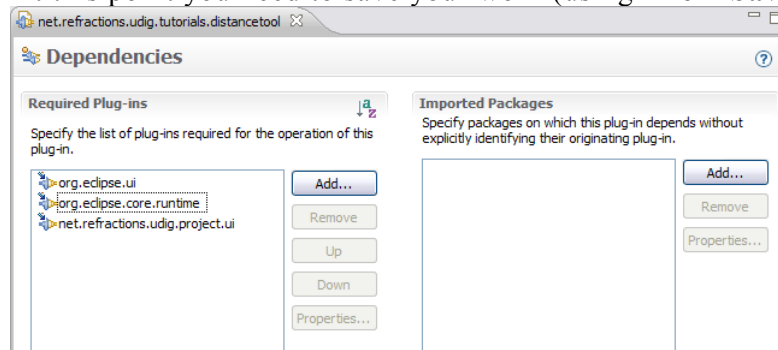- net.refractions.udig.printing.model.edit (1.1.0)
- net.refractions.udig.printing.ui (1.1.0)
- net.refractions.udig.project (1.1.0)
- net.refractions.udig.project.edit (1.1.0)
- net.refractions.udig.project.ui (1.1.0)
- net.refractions.udig.render.feature.basic (1.1.0)
- net.refractions.udig.render.feature.shapefile (1.1.0)
- net.refractions.udig.render.gridcoverage.basic (1.1.0)
- net.refractions.udig.render.wms.basic (1.1.0)

OK   Cancel

6. At this point you need to save your work (using **File > Save**).

net.refractions.udig.tutorials.distancetool

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

- org.eclipse.ui
- org.eclipse.core.runtime
- net.refractions.udig.project.ui

Add...
Remove
Up
Down
Properties...

Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

Add...
Remove
Properties...

7. Changes to the plug-in project (ie MANIFEST.MF file) are used to update the Java project (ie .classpath file). If you do not save your work you will not be able find the tool classes used later in the tutorial.

4 IMPORT ICONS AND POINTERS

You can use normal image files to define tool bar icons, and ool pointers.

By convention icons are organized into the following directories:

| icons/etool16 	Enabled icon used in application tool bar. (16x16 left and top clear)
icons/pointers 	Used to define a cursor (32x32)

Lets download the images used for this tutorial:

1. Download the following file:
   http://udig.refractions.net/tutorials/distance_icons.zip

2. Select **File > Import** to open up the Import wizard

3. Choose **General > Archive File** and press **Next**
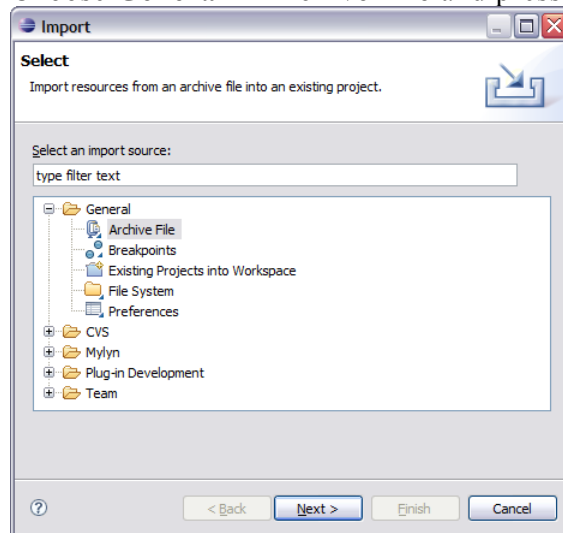
4. Fill in the following details on the Archive File page:
   From archive file: distance_icons.zip
   Into folder: net.refractions.udig.tutorials.distancetool
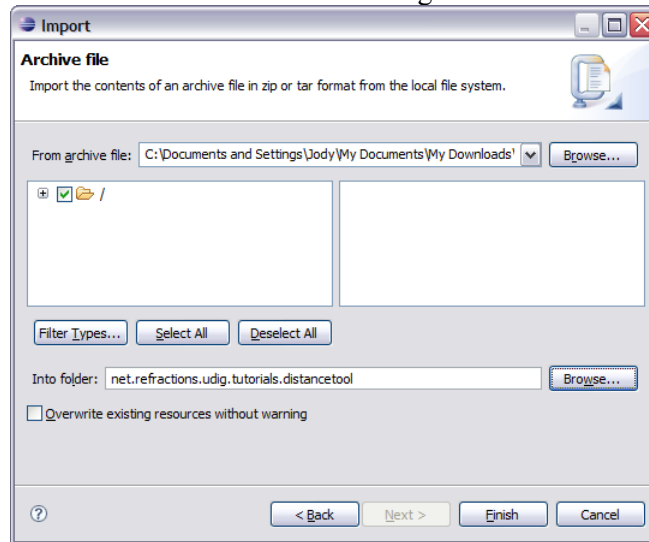
5. Press **Finish**

6. An icons directory will be created.
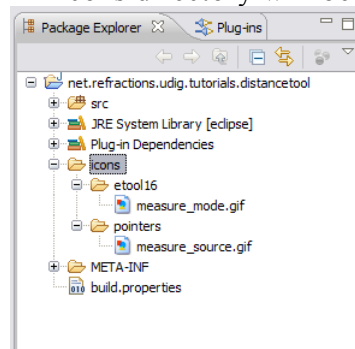
7. Change to the build tab of your MANIFEST.MF editor. Make sure your new **icons** folder is checked as part of the Binary Build.
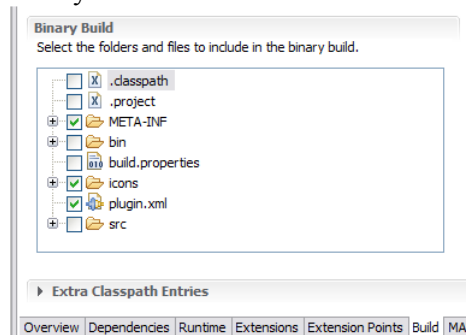
8. This step will ensure that the icons are included when the plug-in is bundled up into a jar.

5 EXTENSIONS

Programming with extensions is accomplished in two parts; filling in a bit of information (letting the Eclipse Platform class know what you are up to); and then implementing a Java class to do the work.

The Platform class acts as a mediator; hooking up the distance tool we define here to the uDig application that will display it on the tool bar and make use of the tool when the user asks.

This is becoming a popular approach; web applications made with the Spring framework often use a single xml file to wire everything together.

The information we provide is stored in a file called plugin.xml; the Platform class reads in all the plugin.xml files when the application is started it and wires up everything.

1. Return to your plug-in **MANIFEST.MF** editor, and switch the the **Extentions** tab.
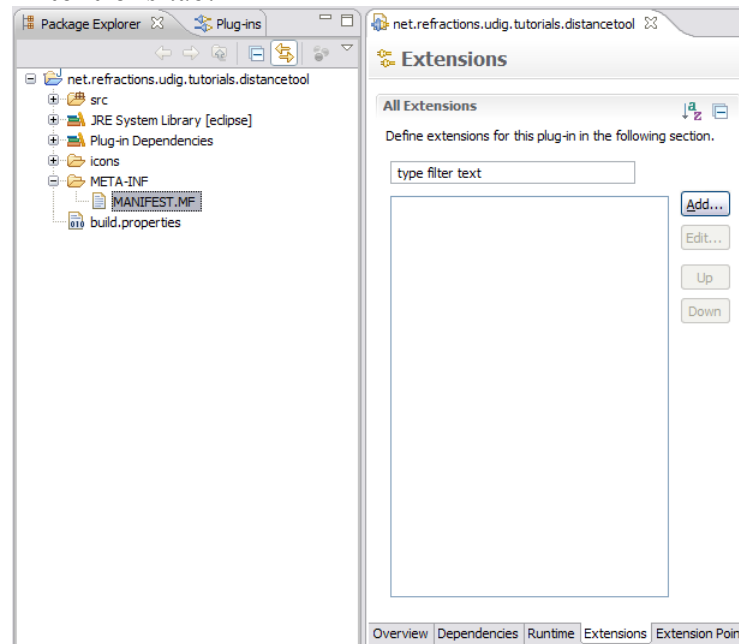
2. Press the **Add** button

3. Select **net.refractions.udig.project.ui.tool** from the list of extension points.
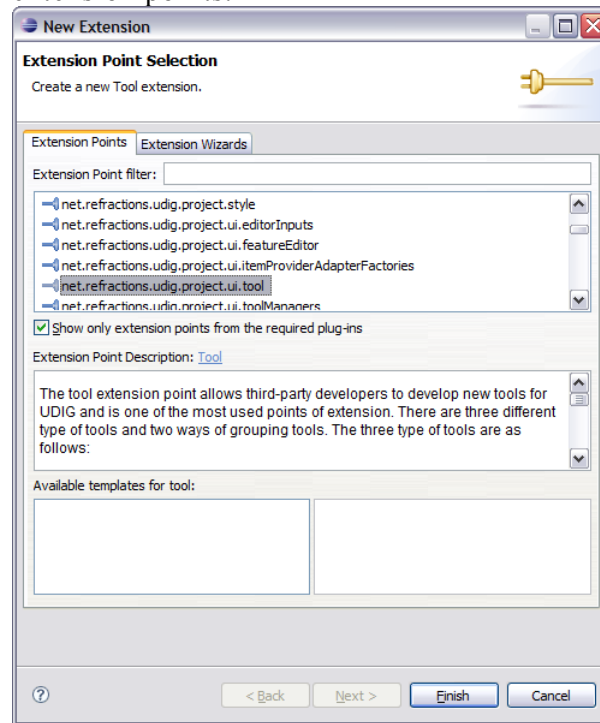
4. Click **Finish**

5. Enter the following Extention Details:
    ID: net.refractions.udig.tutorials.distancetool
    Name: Distance Tool Example

The id and name provided here is used by the Platform class to log errors associated with your tool.
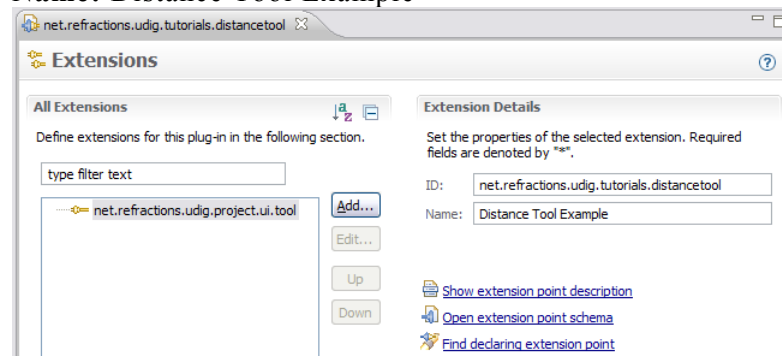
6. Proceed to the next section and we will define our distance tool.

6 TOOL DEFINITION

We can now use the extensions tab to define our distance tool. A single plug-in may define multiple tools, and indeed provide contributions to several extension points.

To create a new tool:

1. Right click on newly **net.refractions.udig.project.ui.tool**, and select **New > modalTool**
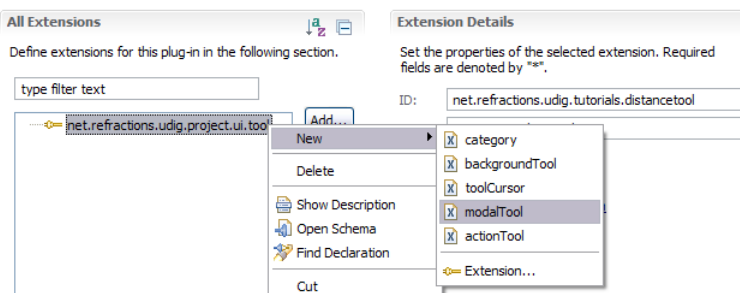
2. Fill in the following value for **id**: net.refractions.udig.tutorial.distancetool.

3. Enter a tool tip message into **tooltip**: Measure the surface distance between two points

4. For **icon**, click on the browse button and select: icons/etool16/measure_mode.gif
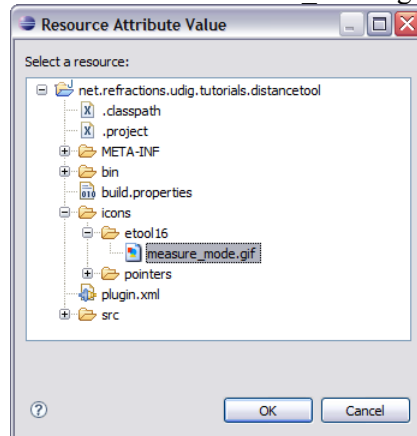
5. Fill in the following for **name**: Distance

6. For **onToolbar**: select true from the list

7. For the tool **categoryId**: net.refractions.udig.tool.category.info

8. Save your work.

All Extensions	
Define extensions for this plug-in in the following section.	

type filter text

☐ ◇ net.refractions.udig.project.ui.tool
   └ |↔| Distance

Add...
Edit...
Up
Down

Extension Element Details

Set the properties of "modalTool". Required fields are denoted by "*".

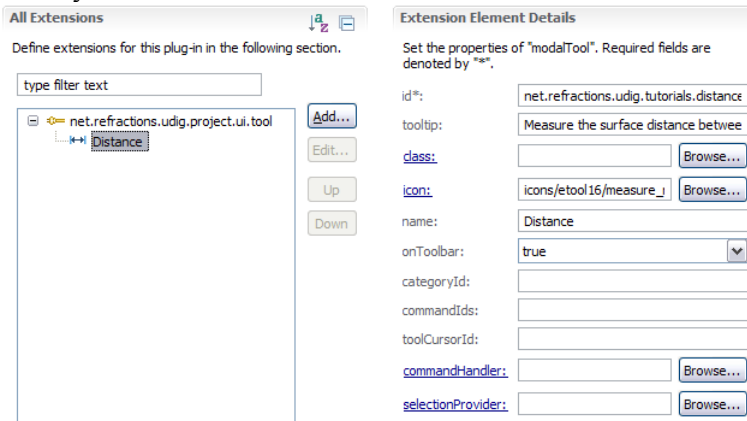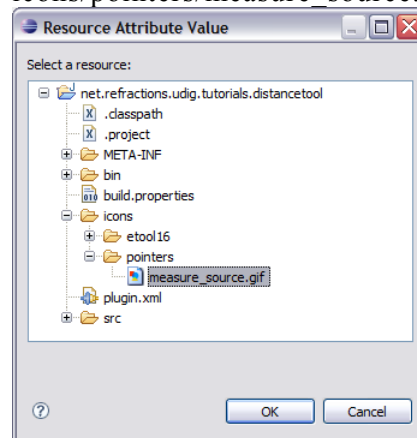|	
id*:	net.refractions.udig.tutorials.distance
tooltip:	Measure the surface distance betwee
class:	Browse...
icon:	icons/etool16/measure_ Browse...
name:	Distance
onToolbar:	true
categoryId:	
commandIds:	
toolCursorId:	
commandHandler:	Browse...
selectionProvider:	Browse...

9. We are going to add a child element that specifies the cursor.

10. Right click on **Distance** and select **New > cursor**.

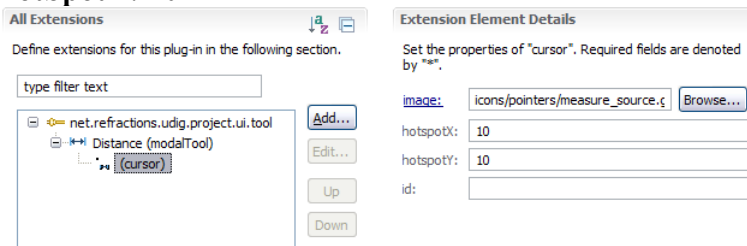11. For the **image**: use the browse button to select:
icons/pointers/measure_source.gif

Resource Attribute Value

Select a resource:

☐ 📂 net.refractions.udig.tutorials.distancetool
   ├ ☒ .classpath
   ├ ☒ .project
   ├ ⊞ 📂 META-INF
   ├ ⊞ 📂 bin
   ├ 📄 build.properties
   ├ ☐ 📂 icons
   │ ├ ⊞ 📂 etool16
   │ └ ☐ 📂 pointers
   │   └ 📄 measure_source.gif
   ├ 📄 plugin.xml
   └ ⊞ 📂 src

⊘   OK   Cancel

12. Fill in the location of the hot spot where the user clicks:
hotSpotX: 10
hotSpotY: 10

All Extensions	
Define extensions for this plug-in in the following section.	

type filter text

☐ ◇ net.refractions.udig.project.ui.tool
   └ |↔| Distance (modalTool)
      └ (cursor)

Add...
Edit...
Up
Down

Extension Element Details

Set the properties of "cursor". Required fields are denoted by "*".

|	
image:	icons/pointers/measure_source.g Browse...
hotspotX:	10
hotspotY:	10
id:	

13. Save your work before continuing to the next section.

7 TOOL IMPLEMENTATION

Now that all the description is out of the way we can implement the DistanceTool class.

1. Return to the **Distance (modalTool)** element
    (it is child of net.refractions.udig.project.ui.tool)

2. Enter in the following for **class**:
    net.refractions.udig.tutorial.distancetool.DistanceTool
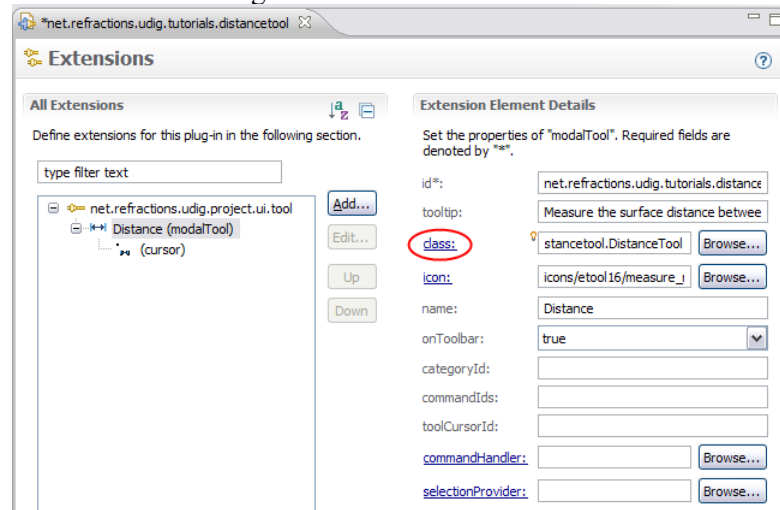
3. Click the **class** link circled above.

4. A **New Java Class** wizard is opened, the needed details should have be already filled in for you.
   Constructors from superclass: uncheck
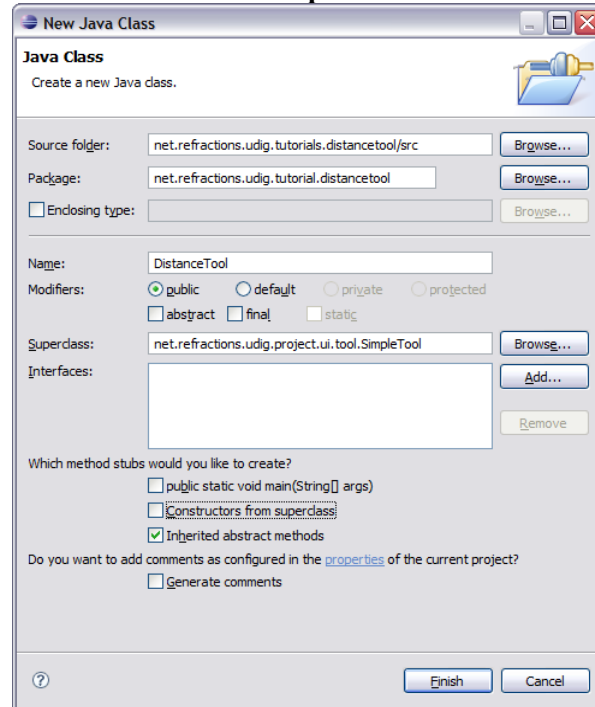
5. Press **Finish**

6. The following file will be created for you.

```
package net.refractions.udig.tutorial.distancetool;

import net.refractions.udig.project.ui.tool.SimpleTool;

public class DistanceTool extends SimpleTool {

}
```

7. Add the following field, to jot down where the user clicked.

```
import com.vividsolutions.jts.geom.Coordinate;
public class DistanceTool extends SimpleTool {
       /** records where in the world the user clicked */
       Coordinate start;

}
```

8. Right click on editor and select **Source > Override/Implement Methods**

9. Expand SimpleTool node and check the following:

- Check onMousePressed(MapMouseEvent)

- Check onMouseReleased(MapMouseEvent)
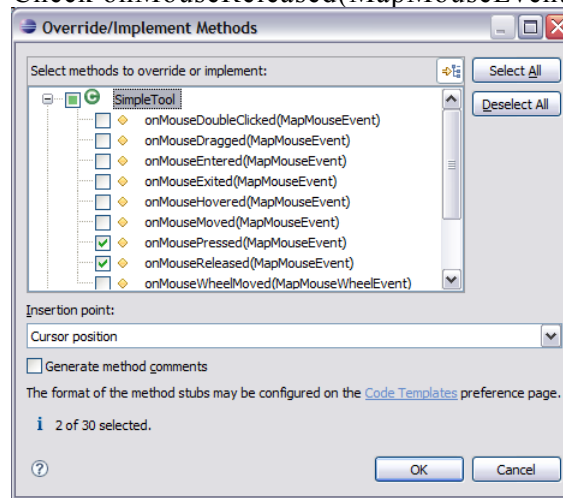
10. Click on the **OK** button to create these methods.

11. Implement the onMousePressed(MapMouseEvent) method

```
@Override
protected void onMousePressed(MapMouseEvent e) {
        start=getContext().pixelToWorld(e.x, e.y);
}
```

12. Implement the onMouseReleased(MapMouseEvent) method.
    We are using the utility **JTS** class to calculate the distance
    between two coordiantes.

```
@Override
protected void onMouseReleased(MapMouseEvent e) {
        Coordinate end=getContext().pixelToWorld(e.x, e.y);
        try {
                double distance=JTS.orthodromicDistance(
                start, end,
                getContext().getCRS() );
                displayOnStatusBar(distance);
        } catch (Exception e1) {
         displayError();
        }
}
```

13.

14. Implement the displayOnStatusBar(double) method.
    We need to

```
private void displayOnStatusBar(double distance) {
      final IStatusLineManager statusBar =
       getContext().getActionBars().getStatusLineManager();

    if( statusBar==null ){
            return; // shouldn't happen if the tool is being used.
    }
    int totalmeters=(int)distance;
      final int km=totalmeters/1000;
      final int meters=totalmeters-(km*1000);
      float cm = (float) (distance-totalmeters)*10000;
      cm = Math.round(cm);
      final float finalcm=cm/100;

      getContext().updateUI(new Runnable(){
            public void run() {
                  statusBar.setMessage(
                   "Distance =  "+km+","+meters+"m "+finalcm+"cm"
                );
            }
      });
}
```

15. Implement the displayError () method

```
private void displayError() {
      final IStatusLineManager statusBar =
            getContext().getActionBars().getStatusLineManager ();

      if( statusBar==null )
            return; // shouldn't happen if the tool is being used.

      getContext().updateUI(new Runnable() {
            public void run() {
                  statusBar.setErrorMessage("Unable to calculate
the distance");
            }
      });
  }
```

16. The file will not compile as we have a few imports to sort out.

17. Press **Ctrl-Shift-0** to import any needed classes. You will have
   one conflict to sort out, please choose:
    org.geotools.geometry.jts.JTS

18. Save you file, this should refresh the project and clean up any
    remaining error message.

8 TESTING THE PLUGIN

We can now run uDig and try out your new plug-in:

1. Select Project > Run from the menu bar and choose the configuration you set-up in the previous tutorial

2. After the application has started up we can put together a map to try out the distance tool.

3. Select **File > New Map**

4. Change to the **Web** view and choose **WMS: jpl nasa** on the page

5. Select the **Blue Marble Next Generation** layer and press **Finish**.

6. Select the Distance Tool from the toolbar and drag from one point to another on the Map.
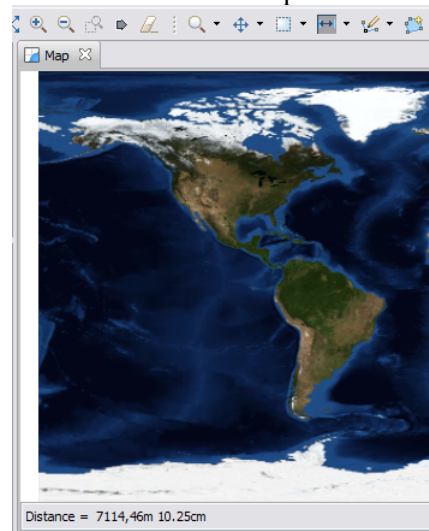
7. The distance will be displayed in the Map status bar.

9 QUESTION AND ANSWER

Here are some common questions:

Q: Unable to calculate the distance!

A: The projection of your data must be defined, you may see this if you are working with a shapefile that does not have a ".prj" file defined.
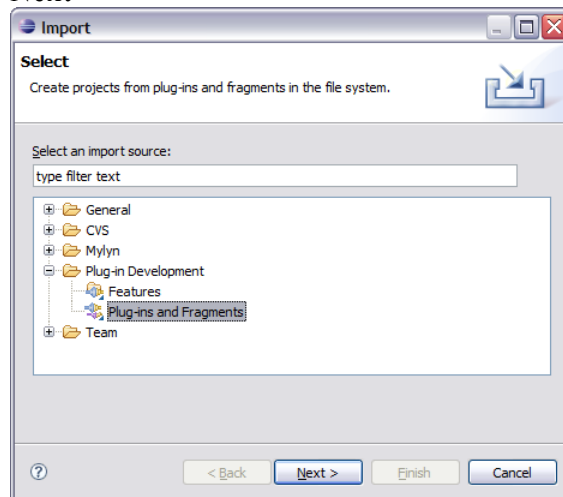
Q: Connection error has occurred

A: Sounds like the data you were looking for is unavailable, try a different WMS.
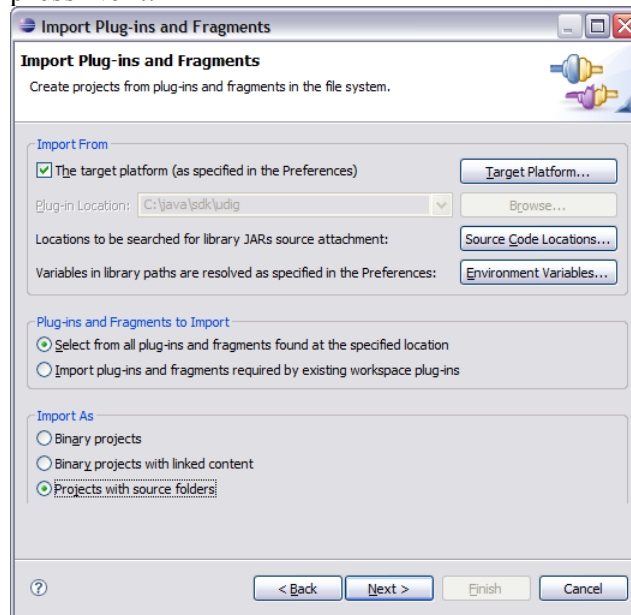
Q: How can I check out the source code
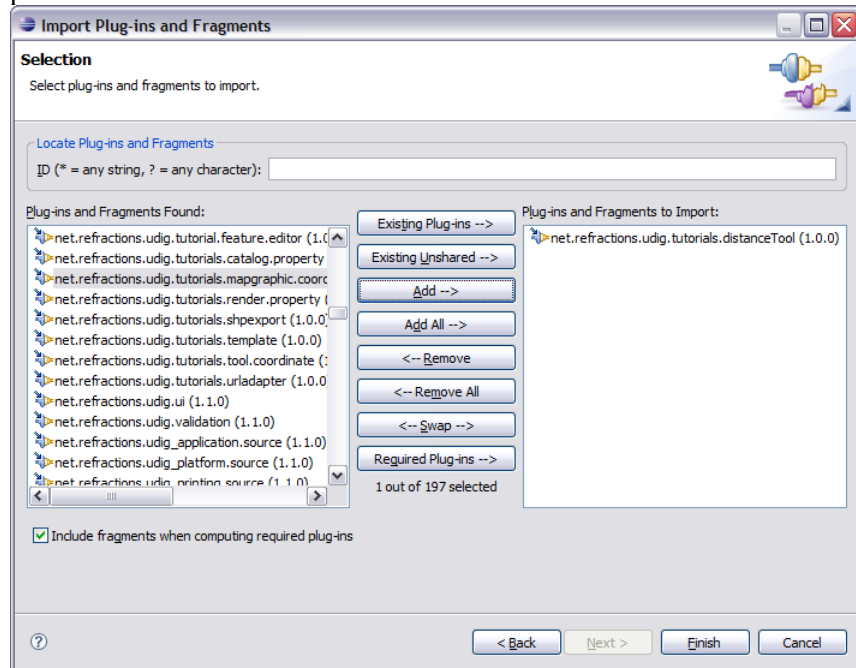
A: It is included in the SDK:

1. File > Import

2. Select Plug-in Development > Plug-ins and Fragments and press Next

3. Change the Import As setting to: Projects with source folders and press Next.

4. Find the net.refractions.udig.tutorials.distanceTool in the list and press the Add button.

5. Press **Finish** and the project will be copied into your workspace.

10 THINGS TO TRY

Here are some additional challenges for you to try:

- Tools are organized into "Categories" each with their own keyboard short-cut, the Distance Tool is in the category "Information".

- You can select the "Information" category by pressing "i" while working with the Map, you can cycle between the different information tools by pressing "i" additional times.

- Plug-ins have a life cycle (controlled by the Platform class); you can add a few System.out statements to your Activator class and watch when the distance tool is started. Remember to add the -consoleLog command line option if you expect to see that System.out.