PLUGIN TUTORIAL

Adding a Distance Tool

30 January 2008





TABLE OF CONTENTS

1Goals	3
2Create a New Plugin	4
3Manifest	7
4Import Icons and Pointers	9
5Extensions	11
6Tool Definition	13
7Tool Implementation	15
8Testing the Plugin	19
9Question and Answer	20
10Things to Try	22

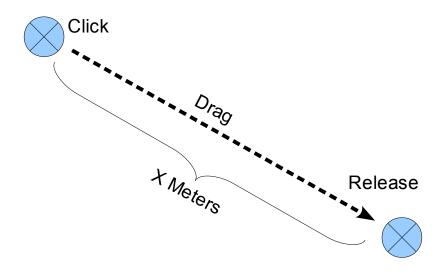
1 GOALS

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.



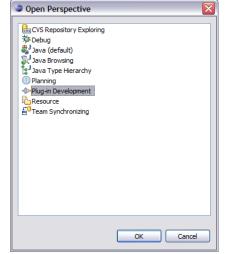
2 CREATE A NEW PLUGIN

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.
- Select Windows > Open Perspective > Other and choose the <u>Plug-in Development perspective from the list.</u>



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

The Plug-in Perspective will contain a few Views and Editors you may not be familiar with from day to day Java programming. 4. Select Plug-in Project and press Next.

New Project				_ 🗆 🔀
Select a wizard				
Create a Plug-in Project				
<u>W</u> izards:				
type filter text				
12 Java Project ※ Java Project from Existin ※ Plug-in Project ※ General ※ CVS ※ Eclipse Modeling Framew ※ Dug-in Development ※ Plug-in Development ※ Examples	-			
0	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

5. Create a name for the plug-in by entering **Project Name:** net.refractions.udig.tutorials.distancetool

🕏 New Plug-in Project 📃 🗆 🔀
Plug-in Project
Create a new plug-in project
Project name: net.refractions.udig.tutorials.distancetool
Use default location
Location: C:/java/workspace/net.refractions.udig.tutorials.distancetool Browse
Project Settings
Create a Java project
Source folder: src
Output folder: bin
Target Platform
This plug-in is targeted to run with:
Oan OSGi framework: Equinox 🗸
(?) < Back Next > Einish Cancel

6. Press the Next Button

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

😂 New Plug-in	Project	_ 🗆 🔀
Plug-in Content Enter the data req	t uired to generate the plug-in.	
Plug-in Properties		
Plug-in ID:	net.refractions.udig.tutorials.distancetool	
Plug-in Version:	1.0.0	
Plug-in Name:	Distancetool Plug-in	
Plug-in Provider:		
<u>C</u> lasspath:		
Ac <u>t</u> ivator: n	ctivator, a Java class that controls the plug-in's life cycle et.refractions.udig.tutorials.distancetool.Activator l make contributions to the UI	
Rich Client Applica Would you like to	ation create a rich client application?	⊙ No
0	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

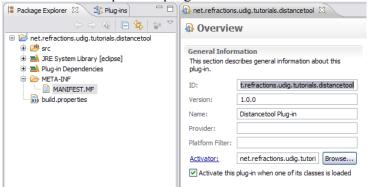
8. Press Finish to create your new project.

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)

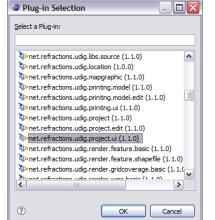
ent.refractions.udig.tutorials.distance	tool 🕅	
Required Plug-ins Specify the list of plug-ins required for t	$\downarrow^{a}_{\mathbb{Z}}$ he operation of this	Imported Packages Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.
plug-in.	Add Remove Up Down Properties	Add Remove Properties

4. Click on the Add button under Required Plug-ins.

You can use the field at

the top of the dialog to filter the list of avalable plug-ins.

Entery "*udig.ui" to quickly find the needed plug-in. 5. <u>Select the net.refractions.udig.project.ui</u> plug-in from the list.



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

Dependencies			(
equired Plug-ins ipecify the list of plug-ins required for the ilug-in.	$ \downarrow^a_{\mathbb{Z}} $	Imported Packages Specify packages on which this plug-in depends with explicitly identifying their originating plug-in.	out
org.eclipse.ui beorg.eclipse.core.runtime beorg.eclipse.core.runtime benet.refractions.udig.project.ui	Add Remove Up Down Properties	Add Rem Proper	ove

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

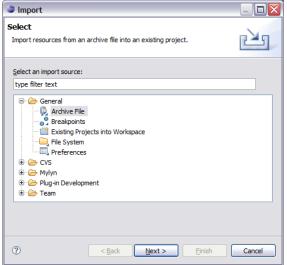
You can use normal image files to define tool bar icons, and ool pointers.

By convention icons are organized into the following directories:

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

Lets download the images used for this tutorial:

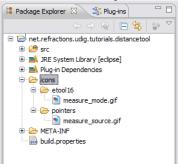
- 1. Download the following file: http://udig.refractions.net/tutorials/distance_icons.zip
- 2. Select File > Import to open up the Import wizard
- 3. Choose General > Archive File and press_Next



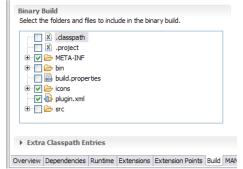
If you are using this work book in a lab setting you will find the file in the same folder as this pdf. Fill in the following details on the Archive File page: From archive file: distance_icons.zip Into folder: net.refractions.udig.tutorials.distancetool

⇒ Import	_ 🗆 🔀
Archive file Import the contents of an archive file in zip or tar format from the local file system.	<u> </u>
From archive file: C:\Documents and Settings\Jody\Wy Documents\Wy Downloads1	Browse
Into folder: net.refractions.udig.tutorials.distancetool	Browse
Qverwrite existing resources without warning	
⑦ < <u>Back</u> <u>N</u> ext> <u>Finish</u>	Cancel

- 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

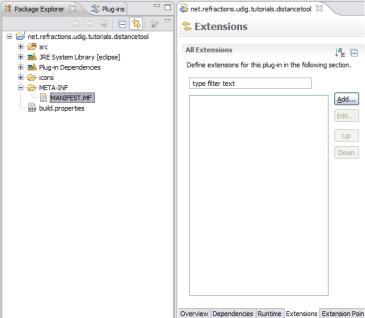
Programming with extensions is accomplished in two parts; filling in a bit of information (letting the Eclipse Platform class know what you are up to); and then implementing a Java class to do the work.

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.

New Extension
Extension Point Selection
Create a new Tool extension.
Extension Points Extension Wizards
Extension Point filter:
net.refractions.udig.project.style
- net.refractions.udig.project.ui.editorInputs
- net.refractions.udig.project.ui.featureEditor
net.refractions.udig.project.ui.itemProviderAdapterFactories inet.refractions.udig.project.ui.tool
Show only extension points from the required plug-ins
Extension Point Description: Tool
The tool extension point allows third-party developers to develop new tools for UDIG and is one of the most used points of extension. There are three different type of tools and two ways of grouping tools. The three type of tools are as follows:
Available templates for tool:
Image: Second

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

All Extensions	la, F	Extension Details	
Define extensions for this plug-in in the following		Set the properties of the selected extension. Required fields are denoted by "*".	
type filter text <pre>net.refractions.udig.project.ui.tool</pre>	Add	ID: net.refractions.udig.tutorials.distancetool Name: Distance Tool Example	
	Up	⇔ Show extension point description ♠ Open extension point schema	

provided here is used by the Platform class to log errors associated with your tool.

The id and name

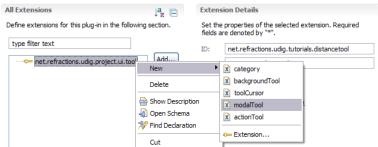
6. Proceed to the next section and we will define our distance tool.

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:

 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

🛢 Resource Attribute Value 📃 🗆 🔀
Select a resource:
🖃 🔛 net.refractions.udig.tutorials.distancetool
.classpath
x .project
🕀 🗁 META-INF
🗈 🗁 bin
build.properties
e 🔁 icons
etool16
measure_mode.gif
pointers
🙀 plugin.xml
🗄 🗁 src
OK Cancel

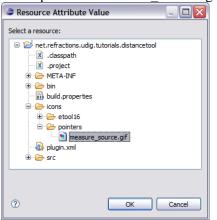
- 5. Fill in the following for name: Distance
- 6. For **onToolbar**: select true from the list

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

8. Save your work.

All Extensions	↓ ^a z ⊟	Extension Eleme	nt Details	
Define extensions for this plug-in in the following section.		Set the properties of "modalTool". Required fields are denoted by "*".		
type filter text		id*:	net.refractions.udig.tutorials.distance	
•• ••	<u>A</u> dd	tooltip:	Measure the surface distance betwee	
	Edit	dass:	Browse	
	Up	icon:	icons/etool16/measure_I Browse	
	Down	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



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

All Extensions	sions 🗸 📮		Extension Element Details		
Define extensions for this plug-in in the following section.		Set the properties of "cursor". Required fields are denoted by " $*$ ".			
type filter text		image:	icons/pointers/measure_source.c		
the interfactions.udig.project.ui.tool definition Distance (modalTool) finition (cursor)	<u>A</u> dd Edit	hotspotX:	10		
		hotspotY:	10		
	Up	id:			
	Down				

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

*net.refractions.udig.tutorials.distancetool 🛛			-	
Extensions			(
All Extensions	↓ ^a z 🖻	Extension Element Details		
Define extensions for this plug-in in the following section.		Set the properties of "modalTool". Required fields are denoted by "*".		
•- net.refractions.udig.project.ui.tool in-++ Distance (modalTool)	<u>A</u> dd	id*: tooltip:	Measure the surface distance between stance of the surface distance between stance tool.DistanceTool Browse	
⊷ ••• (cursor)	Up	icon:	icons/etool16/measure_I Browse	
	Down	name: onToolbar:	bistance true	
		categoryId: commandIds:		
		toolCursorId:	Browse	
		selectionProvider:	Browse	

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 New Java Class Java Class Create a new Java dass. Source folder: net.refractions.udig.tutorials.distancetool/src Browse. Package: net.refractions.udig.tutorial.distancetool Browse. Enclosing type: Brows Name: DistanceTool ⊙ public O default Modifiers: abstract final static net.refractions.udig.project.ui.tool.SimpleTool Superclass: Browse... Interfaces: <u>A</u>dd... Remove Which method stubs would you like to create? public static void main(String] args) Constructors from superclass ✓ Inherited abstract methods Do you want to add comments as configured in the properties of the current project? Generate comments ? Einish Cancel

- 5. Press Finish
- 6. The following file will be created for you.

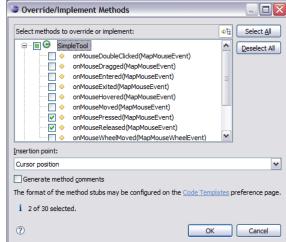
package net.refractions.udig.tutorial.distancetool; import net.refractions.udig.project.ui.tool.SimpleTool; public class DistanceTool extends SimpleTool { }

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.

- 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.

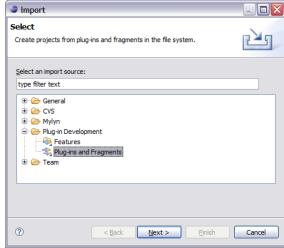
Q: Connection error has occurred

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.

Import Plug-ins and Fragments	_ 🗆 🛛			
Import Plug-ins and Fragments Create projects from plug-ins and fragments in the file system.	-00-			
Create projects from plug-ins and fragments in the file system.				
C Import From				
\fbox{The} The target platform (as specified in the Preferences)	Target Platform			
Plug-in Location: C:\java\sdk\udig 🗸 🗸	Browse			
Locations to be searched for library JARs source attachment:	Source <u>C</u> ode Locations			
Variables in library paths are resolved as specified in the Preferences:	Environment Variables			
C Plug-ins and Fragments to Import				
O Select from all plug-ins and fragments found at the specified location				
O Import plug-ins and fragments required by existing workspace plug-in	IS			
C Import As				
O Bin <u>a</u> ry projects				
O Binary projects with linked content				
● Projects with source folders				
⑦ <back next=""></back>	Einish Cancel			

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

Import Plug-ins and Fragments		
Selection Select plug-ins and fragments to import.		
Locate Plug-ins and Fragments ID (* = any string, ? = any character): Plug-ins and Fragments Found:		Plug-ins and Fragments to Import:
Interrefractions.udg.tutorial.feature.editor (1.C Pret.refractions.udg.tutorials.catalog.property Pret.refractions.udg.tutorials.mapgraphic.coord Pret.refractions.udg.tutorials.render.property (Pret.refractions.udg.tutorials.render.property (Pret.refractions.udg.tutorials.tendate (10.0) Pret.refractions.udg.tutorials.tool.coordinate (1 Pret.refractions.udg.tutorials.tradapter (10.0) Pret.refractions.udg.tutorials.tradapter (10.0) Pret.refractions.udg.tutorials.tradapter (10.0) Pret.refractions.udg.validation (11.0) Pret.refractions.udg.application.source (11.0) Pret.refractions.udg.patform.source (11.0) Pret.refractions.udg.patform.source (11.0)	Existing Plug-ins> Existing Unshared> Add> Add All>	[*] ↓>net.refractions.udig.tutorials.distanceTool (1.0.0)
Include fragments when computing required plug-ins	; 	ack Next > Finish Cancel

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.