

CUSTOM RCP APPLICATION

Create your own uDig

18 August 2008



Refractions 
RESEARCH
THE GEOSPATIAL EXPERTS

TABLE OF CONTENTS

1 Goals.....	3
2 Branding Plugin.....	4
3 Branding Imagery.....	6
4 Plugin Dependencies.....	9
5 Application.....	11
6 Perspective.....	14
7 Perspective Extensions.....	17
8 Product.....	19
9 Product Configuration.....	21
10 Launching.....	23
11 Splash.....	25
12 Branding.....	26
13 Try It Out.....	27
14 Distribute.....	28
14.1 Include Images in Branding Plug-in Build.....	28
14.2 Product Export.....	29
15 What to Do Next.....	30

1 GOALS

In this workbook we are going to create a custom application based on uDig GIS platform.

We will learn how to:

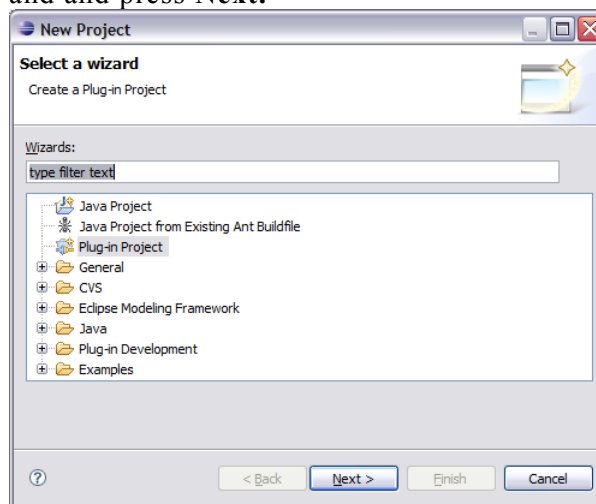
- specify branding elements such as splash screen and title
- Declare how components in the application will be organized
- Package up the application for others to download and use

You should of completed the two previous tutorials, we will require a working uDig SDK Development Environment. If you completed the Distance Tool Plug-in you can include it in your application.

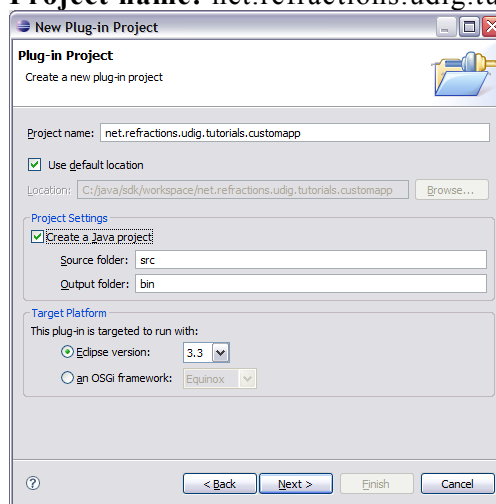
2 BRANDING PLUGIN

In this section we create a plug-in that contains all of the custom “branding” data for our application.

1. Select **File > New > Project...**
2. In the New Project wizard select **Plug-in Project** from the list and press **Next**.

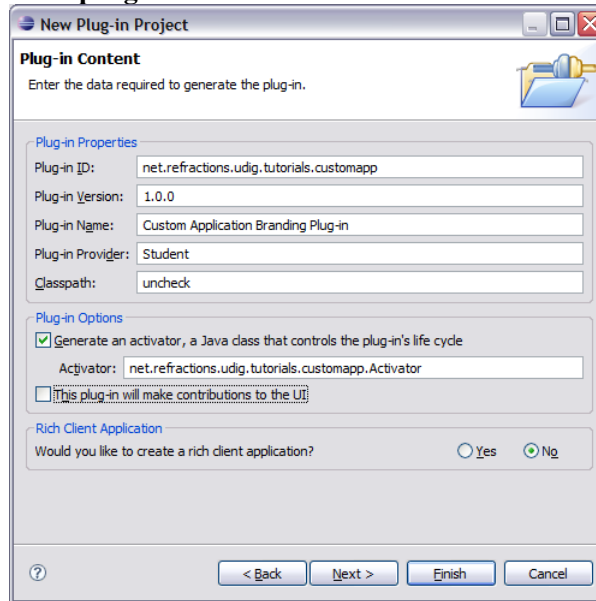


3. On the **Plug-in Project** page enter:
Project name: net.refractions.udig.tutorials.customapp



4. Press the **Next** button

- On the Plug-in Content page provide the following information:
Plug-in Name: Custom Application Branding Plug-in
Plug-in Provider: Student
This plug-in will make contributions to the UI: uncheck



- Press **Finish** to continue (we are not interested in using any of the included templates).
- If the **Open Associated Perspective** dialog pops up you can select **Yes**

Eclipse is switching perspectives to assist you in working on your new plug-in.



- Your new project is added to the Package Explorer, and the **MANIFEST.MF** file will be opened for you to review.

3 BRANDING IMAGERY

Now that your branding plug-in is created we need to provide a little bit of content before we get down to coding and configuration. An important aspect of defining a new product is the all important visual identity.

Fiddling around with drawing programs is a bit beyond the scope of this workbook but we can quickly cover what is needed.

- splash.bmp



Hello World

This product includes software developed by Refrations Reserach, the Eclipse Foundation, the Open Source Geospatial Foundation and others.

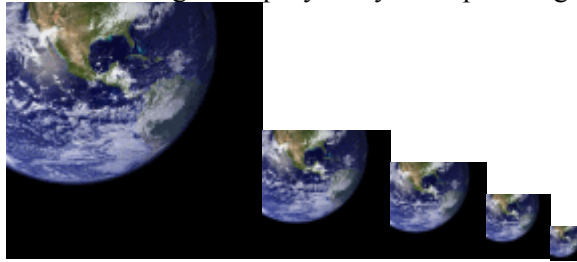
The splash screen must be in bmp format, it is actually read by a C++ program responsible for launching your application. The recommended for this image is 500x330 pixels.

- about.gif



The about image is limited to the gif format, the maximum limit is 500x330, for images smaller than 250x330 you have the option of including your own text and html links in the about dialog.

- Window images displayed by the operating system.



gif files sized 128x128,64x64,48x48,32x32,16x16

- Assorted icons

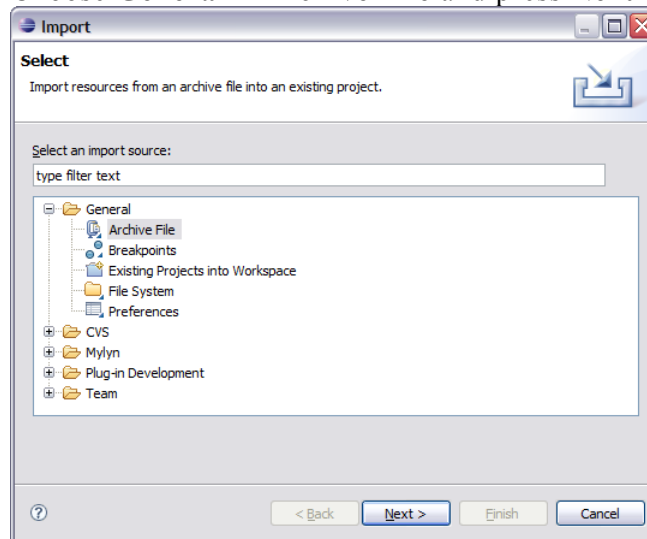
icons/elcl16 icons/dlcl16	Used on a view tool bar actions. (16x16 left and top clear)
icons/etool16 icons/dtool16	Used in application tool bar and menu bar (16x16 left and top clear)
icons/obj16	Used in trees to represent objects or ideas (16x16 centered, bottom clear)
icons/ovr16	Used to decorate a obj16 icon to indicate state (7x8 one pixel white outline)
icons/wizban	Banner used in wizard dialog windows (55x45 bottom left on a blue gradient)

Enough of that - lets download the files we need and keep going. If you have any questions please consult the Eclipse User Interface Guidelines.

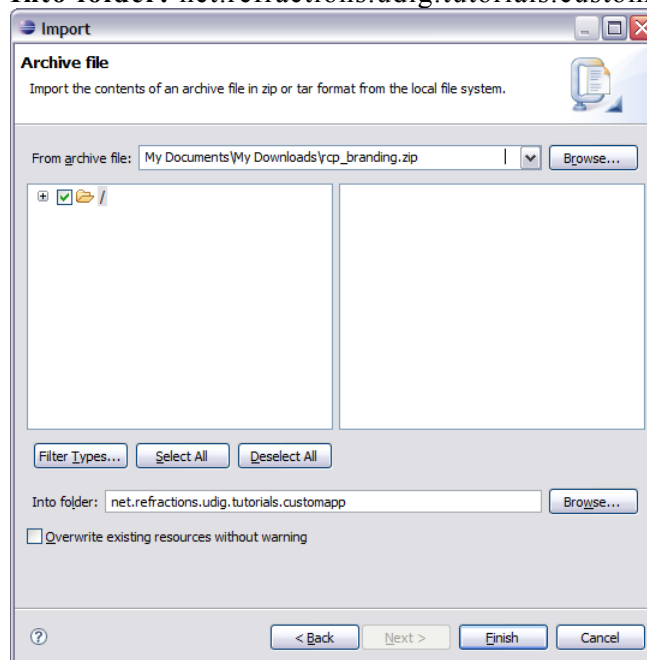
1. Download the following file:
http://udig.refractorions.net/tutorials/rcp_branding.zip
2. Select **File > Import** to open up the Import wizard

If you are using this workbook in a lab setting you will find the file nl.zip in the same folder as this pdf.

3. Choose **General > Archive File** and press **Next**



4. Fill in the following details on the **Archive file** page:
From archive file: rcp_branding.zip
Into folder: net.refractions.udig.tutorials.customapp

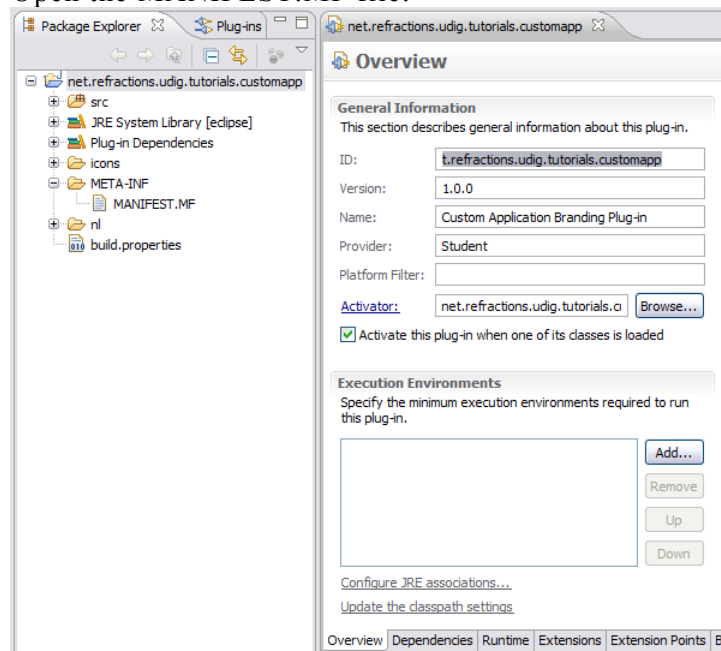


5. Press **Finish**, two folders will be added to your project. There is an “nl” folder with language specific branding; and an icons folder.

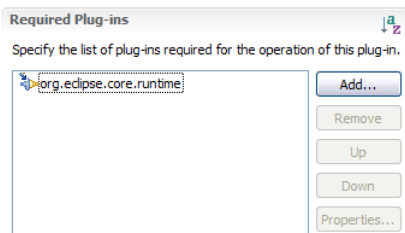
4 PLUGIN DEPENDENCIES

The plug-in MANIFEST.MF file is used to define the requirements and capabilities of any plug-in. You will recognize a lot of the information here from when you defined the plug-in.

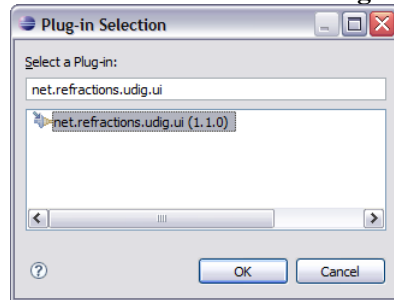
1. Open the MANIFEST.MF file:



2. Switch to the **Dependencies** tab.
3. Press the **Add** button in the **Required Plug-ins** section.



4. Select **net.refractions.udig.ui** from the list and press **OK**



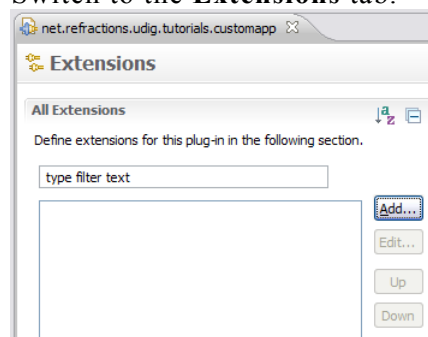
*You can start typing
*.udig.ui to quickly
filter the list.*

5. Select **File > Save** from the menu bar to save the file. This step is important as it gives eclipse a chance to generate:
 - The **.classpath** file used by the Java build system
 - The **.project** file used by the IDE
6. With this information in place the wizards and dialogs we used in the next section will be able to find classed defined by the **net.refractions.udig.ui** plug-in.

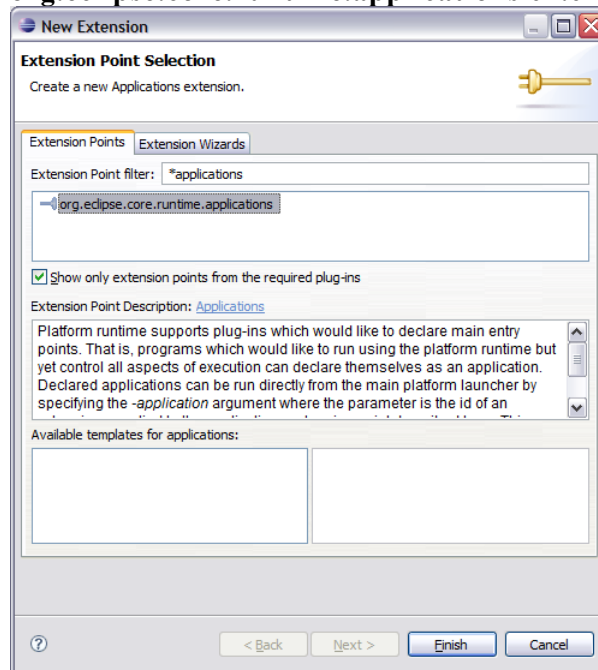
5 APPLICATION

Everything in eclipse is handled through the plug-in extension mechanism, and defining an application is no different. The **org.eclipse.core.runtime** plug-in defines the concept of what it means to be an Application.

1. Switch to the **Extensions** tab.



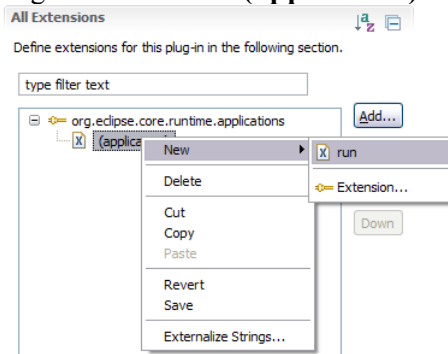
2. Press the **Add** button, and select the **org.eclipse.core.runtime.applications** extension point.



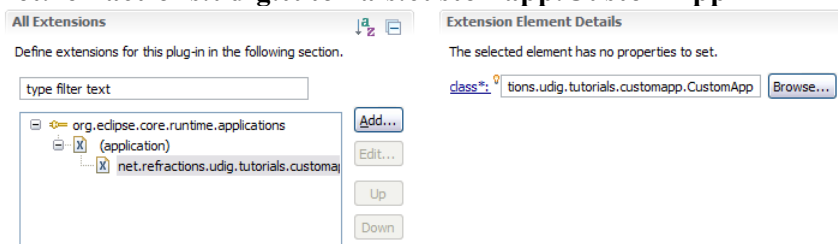
3. Press the **Finish** button.

You can check what xml is being generated at any point by looking at the plugin.xml tab.

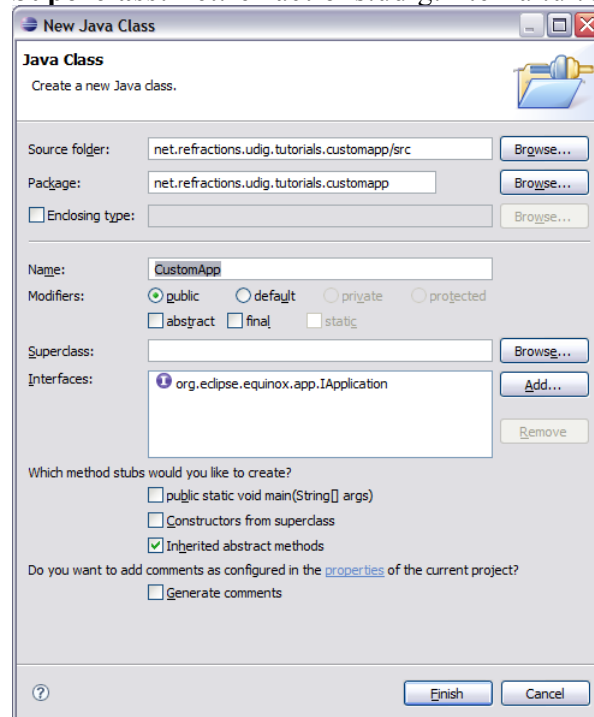
4. Right click on the **(application)** element and select **New > Run**



5. In the **class** text field enter:
net.refrations.udig.tutorials.customapp.CustomApp



6. Save your work, and then press the **class** link.
7. Eclipse will open up a wizard for creating a New Java Class. Enter the following:
Superclass: net.refrations.udig.internal.ui.UDIGApplication



8. Press the **Finish** Button
9. In the newly created file add the following method:

```
@Override
protected WorkbenchAdvisor createWorkbenchAdvisor() {
    return new UDIGWorkbenchAdvisor() {
        @Override
        public String getInitialWindowPerspectiveId() {
            return
"net.refractions.udig.tutorials.customapp.perspective";
        }
    };
}
```

10. If you just cut and paste the above code you will be left with several problems (as UDIGWorkbenchAdvisor is not imported yet).

Here are two ways to fix this:

- Type **control-shift-o** this keyboard short cut will try and fix as many import statements as it can. Save the file after this change and the correct import statements should be there.
- Or you can just type them in at the top of the file:

```
import net.refractions.udig.internal.ui.UDIGWorkbenchAdvisor;
```

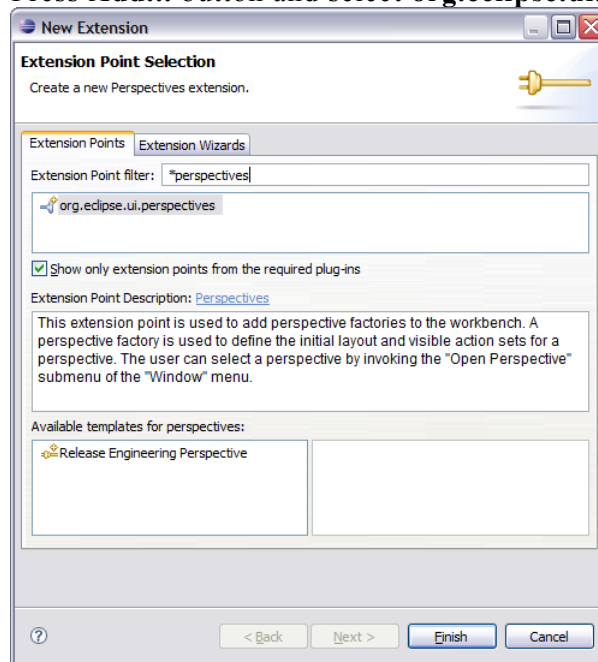
11. Save your work and lets move on.

6 PERSPECTIVE

Perspectives are used to gather together a screen layout (usually for a specific task).

In the previous section we saw that an application can list a default perspective to use when it starts up – this is only used the first time the user runs the application. On subsequent runs the application will maintain the arrangement as the user left them.

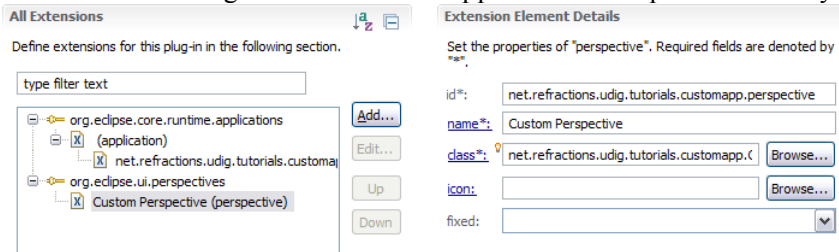
1. Return to the Extension editor. It is the editor titled: **net.refraction.udig.tutorials.custom.app**, make sure the editor is still on the Extensions tab.
2. Press **Add...** button and select **org.eclipse.ui.perspectives**



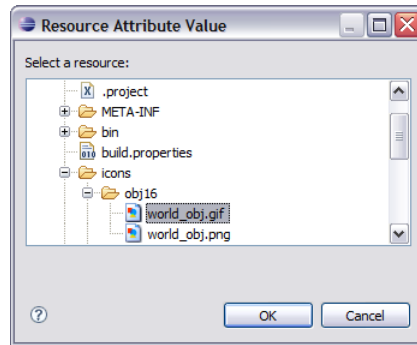
3. Press **Finish**

If the name item was not created for you right click on org.eclipse.ui.perspective and add the name element yourself.

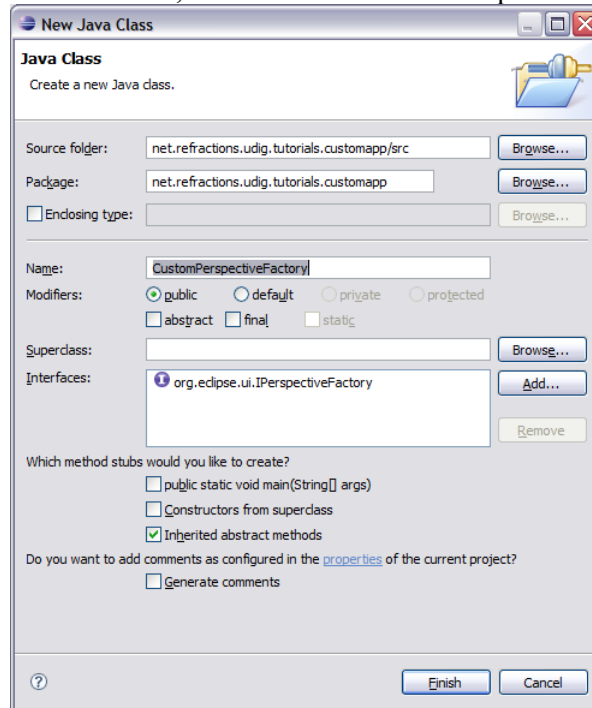
- Select the **name** element.
id: net.refractions.udig.tutorials.customapp.perspective
name: Custom Perspective
class:
net.refractions.udig.tutorials.customapp.CustomPerspectiveFactory



- Click the **Browse** button after **icon** and select **world_obj.gif**



- Save the file, click the **class** link to open the Java Class dialog.



- Press **Finish** to create the class.
- Add the following static constants:

```
private static final String BOOKMARKS =  
    "org.tcat.citd.sim.udig.bookmarks.internal.ui.BookmarksView";  
private static final String PROJECTS =  
    "net.refractions.udig.project.ui.projectExplorer";  
private static final String LAYERS =  
    "net.refractions.udig.project.ui.layerManager";
```

9. In the **createInitialLayout** method, add the following code:

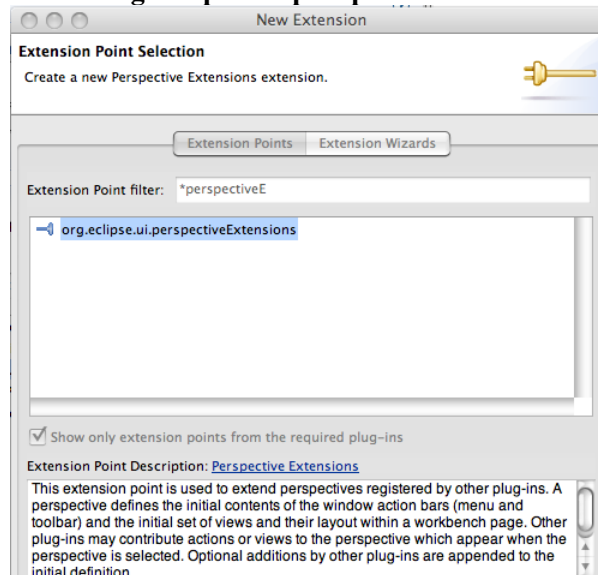
```
layout.addFastView(PROJECTS);  
layout.addView(LAYERS, IPageLayout.LEFT, 0.3f,  
    IPageLayout.ID_EDITOR_AREA);  
layout.addView(BOOKMARKS, IPageLayout.BOTTOM, 0.7f, LAYERS);
```


7 PERSPECTIVE EXTENSIONS

In the previous section we used one method for defining a perspective – we used code to manually add views to the perspective. We will now use a second technique – using a plugin.xml extension to declare what a perspective is composed of.

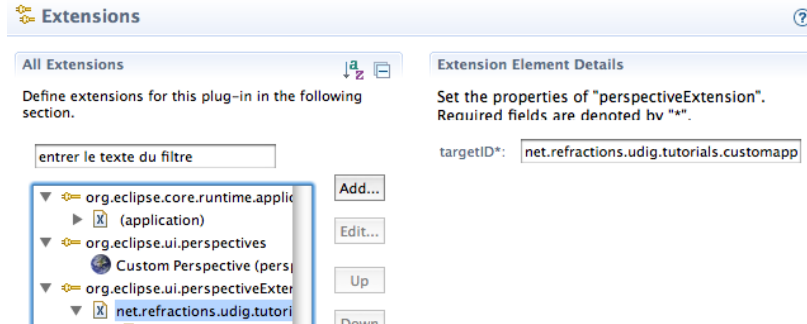
To control what menus are shown we will use an extension to add an Action Set to a perspective. The “Perspective Extensions” extension point can be used to control what actions are available, what views are displayed and so forth.

1. Open the **plugin.xml** file.
2. Switch to the **Extensions** tab
3. Click the **Add...** Button
4. Select **org.eclipse.ui.perspectiveExtensions**

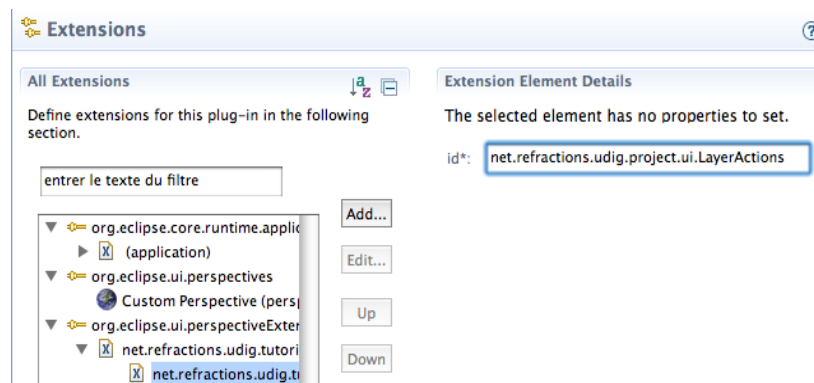


5. Press **Finish**

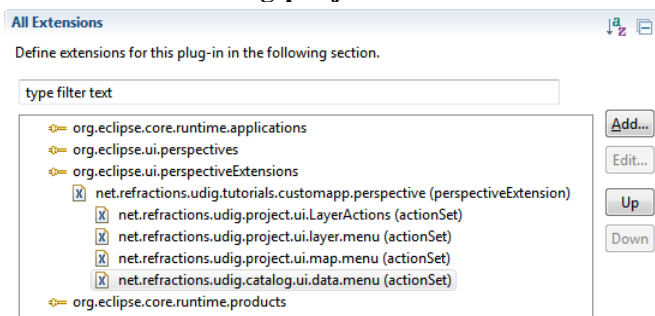
6. Select the **targetID** element and enter **net.refractions.udig.tutorials.customapp.perspective**



1. Right click on **net.refractions.udig.tutorials.customapp.perspective** and select **New > Action Set**
2. Select the **id*** field and enter: **net.refractions.udig.project.ui.LayerActions**



3. Repeat this process for the following additional entries:
net.refractions.udig.project.ui.layer.menu
net.refractions.udig.project.ui.map.menu
net.refractions.udig.project.ui.data.menu

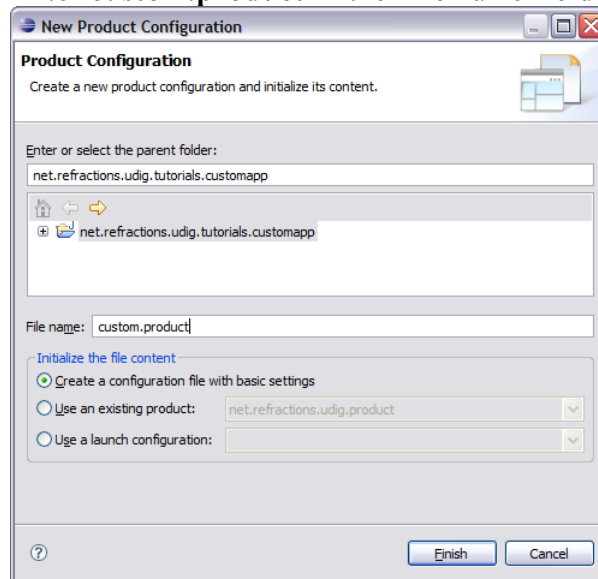


8 PRODUCT

Now that we have an application we can bundle it up as a product for others, and how it is to be run.

To start lets add a product file to our branding plugin:

1. Select **net.refractions.udig.tutorials.customapp** in the Project Explorer
2. Selection **File > New > Product Configuration**
3. Select **net.refractions.udig.tutorials.customapp**
4. Enter **custom.product** in the File name field

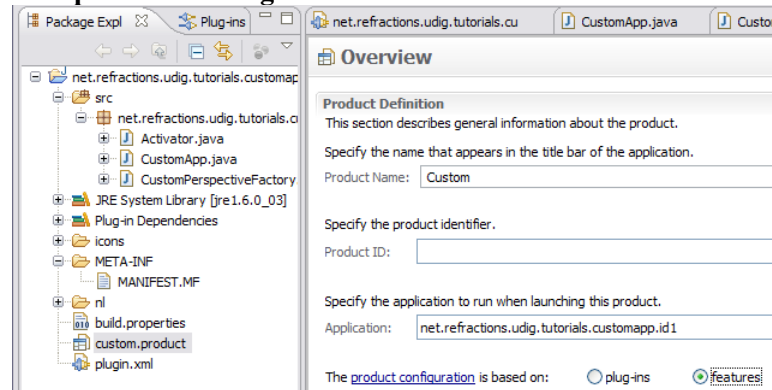


5. Press the **Finish** button.
6. The **custom.product** file is created and opened for your review.

7. We are going to fill in the blanks defining our product.

Product Name: Custom

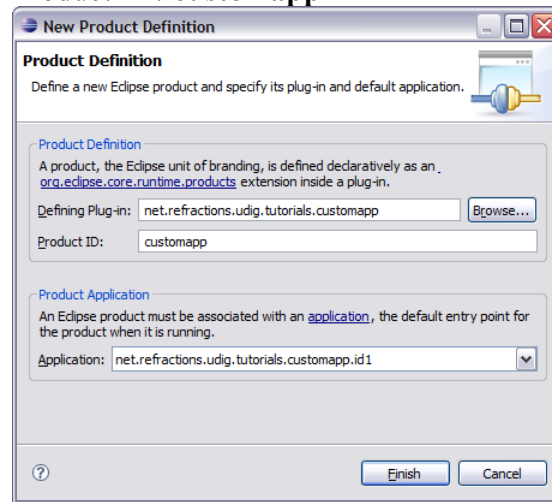
The product configuration is based on: features



8. Save your work and hit the **New...** button next to **Product ID**.

9. In the Product Definition wizard enter:

Product ID: customapp

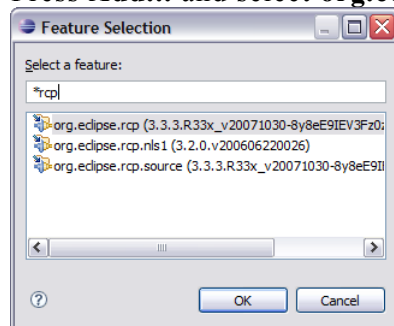


10. Press the **Finish** button and save your work.

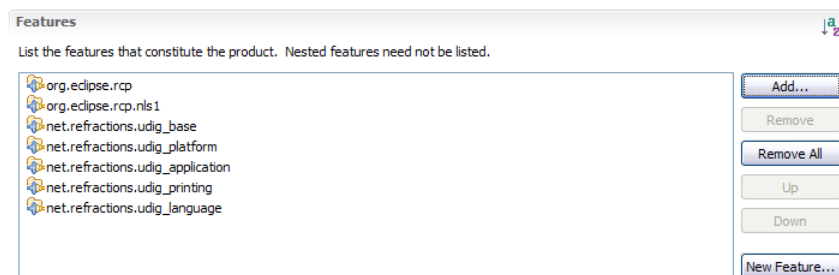
9 PRODUCT CONFIGURATION

We are going to use “features” (ie groups of plug-ins) to define what is included in our application. We will making use of features provided as part of eclipse rcp, features provided as part of the uDig SDK and we will make one feature to hold our branding plugin.

1. Change to the **Configuration** tab, we are going to be defining this product in terms of the features we require to be present.
2. Press **Add...** and select **org.eclipse.rcp** from the list.

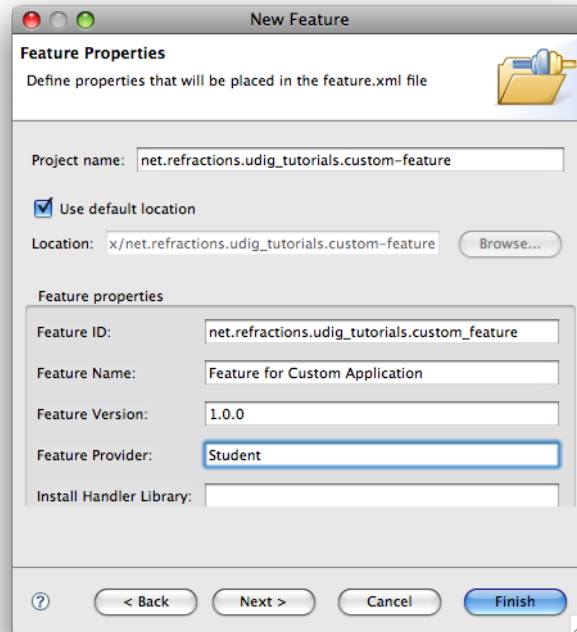


3. Press the **OK** button to add the feature.
4. In the same manner add the following features:
org.eclipse.rcp.nls1
net.refractions.udig_base
net.refractions.udig_platform
net.refractions.udig_application
net.refractions.udig_printing
net.refractions.udig_language



5. We now need to define a feature to hold our branding plugin (so it is included in the product!).
6. Pressing the **New Feature...** button.

- In the New Feature wizard enter the following values:
Project name: net.refractions.udig_tutorials.custom-feature
Feature Name: Feature for Custom Application
Feature Provider: Student



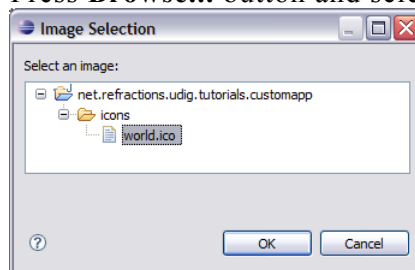
- Press the **Next** button.
- On the Referenced Plug-ins and Fragments page select:
 - net.refractions.udig.tutorials.customapp
 - net.refractions.udig.tutorials.distancetool (optional)
- Press **Finish**

10 LAUNCHING

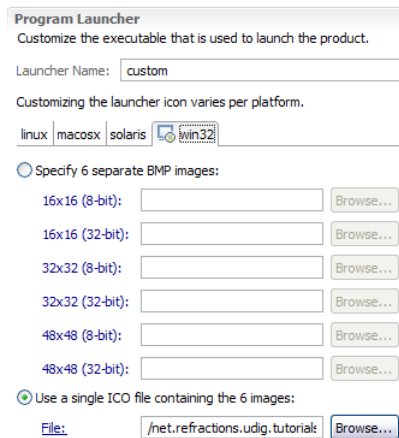
Eclipse RCP applications are bundled up with a platform specific launcher. You can run your application as a normal Java application but this is more fun.

The `osgi.parentClassloader=ext` argument is used to let runtime system know that we want access to JRE extensions (specifically JAI and ImageIO).

1. Return to the **custom.product** editor, and change to the **Launching** tab.
2. We are going to fill in the the Program Launcher section:
 - Enter **custom** into the Launcher Name field
 - Under Customizing the launcher icon varies per platform select **win32**
 - Select the **Use a single ICO file containing the 6 images:** radio button
 - Press **Browse...** button and select **icons/world.ico** for the file



3. Here is what this looks like when you are done:



The screenshot shows the 'Program Launcher' configuration window. The title bar reads 'Program Launcher'. Below the title bar, it says 'Customize the executable that is used to launch the product.' There is a text field for 'Launcher Name' containing the word 'custom'. Below that, it says 'Customizing the launcher icon varies per platform.' There are four tabs: 'linux', 'macosx', 'solaris', and 'win32'. The 'win32' tab is selected. Underneath the tabs, there are two radio button options. The first is 'Specify 6 separate BMP images:' which is currently unselected. It has six rows, each with a label (e.g., '16x16 (8-bit):'), a text input field, and a 'Browse...' button. The second option is 'Use a single ICO file containing the 6 images:' which is selected. It has a 'File:' label, a text input field containing the path '/net.refractions.udig.tutorial!', and a 'Browse...' button.

4. Now lets work on the **Launching Arguments**

- Select the **All Platforms** tab, and fill the the following **VM Arguments**:
-Xmx386M -Dosgi.parentClassloader=ext
- Select the **macosx** tab and add the following to the existing **VM Arguments**:
-Djava.awt.headless=true

5. That is it; we have enough information to move along with.

11SPASH

The splash screen is displayed by the launcher to give the user something to watch as Java loads up; later on in the start up process a progress bar is drawn over top of the splash screen.



When making your own splash screen you will want to plan out the location of the progress bar and progress message as shown above.

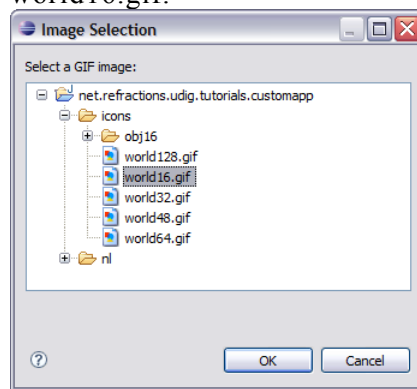
1. Select the **Splash** tab
2. Check the **Add a progress bar** check box and enter:
 - Enter **0** in the x-offset field
 - Enter **196** into the y-offset field
 - Enter **500** in the width field
 - Enter **10** in the height field
3. Check the **Add a progress message** check box and enter:
 - Enter **180** into the y-offset field
 - Enter **16** into the height field
4. Change the Text color to **white**

A screenshot of the "Customization" dialog box in an IDE. The title bar says "Customization". Below the title bar, it says "Create a custom splash screen using one of the provided templates or by adding a progress bar and message." There is a "Template:" dropdown menu with "<none>" selected. Below that, it says "Specify the geometry and color of the progress bar and message." There are two checked checkboxes: "Add a progress bar" and "Add a progress message". For "Add a progress bar", the fields are: x-offset: 0, y-offset: 196, width: 500, height: 10. For "Add a progress message", the fields are: x-offset: 7, y-offset: 180, width: 445, height: 16. There is a "Text Color:" field with a white color swatch.

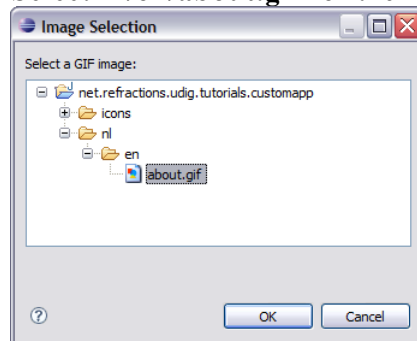
12BRANDING

We are now down to the final branding information, the icons used for windows and the image displayed in Help>About.

1. Click the **Branding** tab
2. Click on the browse button next to 16x16 image, select **world16.gif**.



3. In a similar manner take care of:
icons/world16.gif
icons/world32.gif
world48.gif
world64.gif
world128.gif
4. Select **nl/en/about.gif** for the **About Dialog** image:



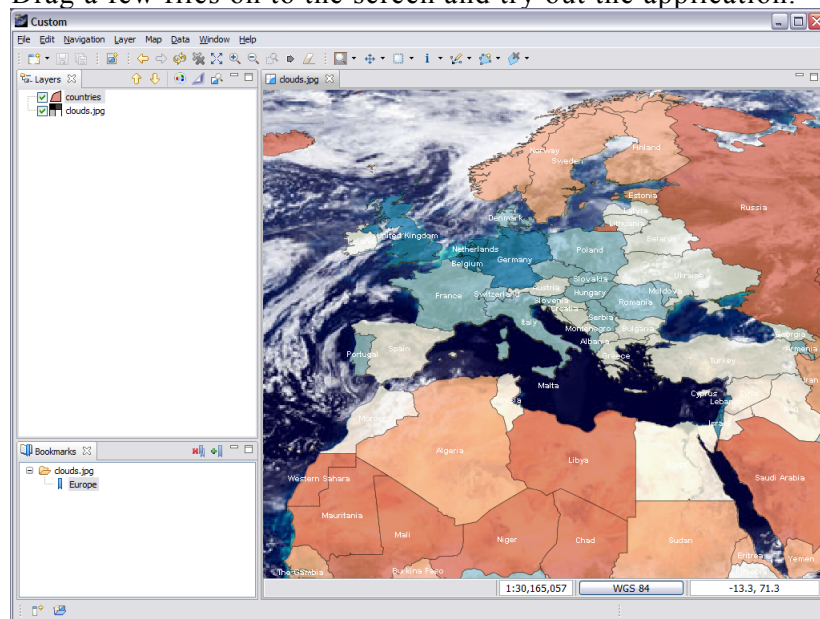
5. And enter the following Text:

```
Hello World

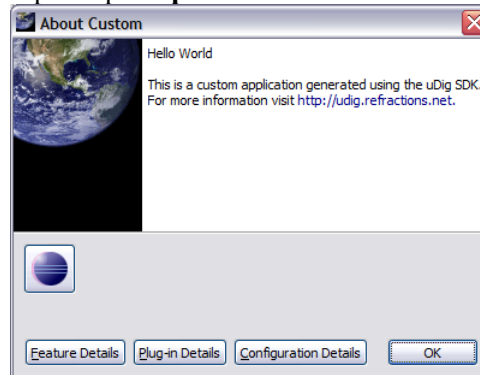
This is a custom application generated using the uDig SDK.
For more information visit http://udig.refractions.net.
```

13 TRY IT OUT

1. Select the **Overview** tab
2. Click the **synchronize** link in the **Testing** section
3. Click the **Launch an Eclipse application** link, check that your splash screen with progress bar and progress message works
4. Drag a few files on to the screen and try out the application.



5. The application defaults to our **Custom perspective**, to switch perspectives use **Window > Open Perspective > Other**
6. Open up **help > About** and have a look at the branding info.



7. **Alt-tab** between windows to look at the window icon.

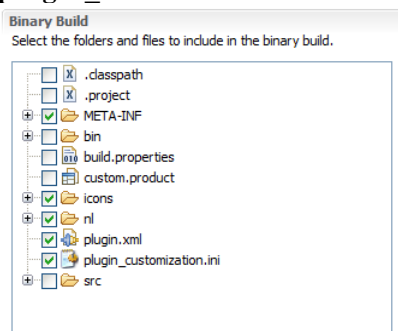
14DISTRIBUTE

This section shows how to export your product for others. We have a couple of things to do before we get down to running to product export wizard.

14.1INCLUDE IMAGES IN BRANDING PLUG-IN BUILD

Everything that you wish bundled into a plug-in's jar needs to be ticked off as part of the **build.properties** file.

1. We need to make all the files we need are included in our branding plug-in.
2. Open up **MANIFEST.MF** and switch to the **Build** tab.
3. Check of the following files in the **Binary Build**:
icons
nl
plugin_customization.ini



4. Save the file and we can go on to the next section.

14.2 PRODUCT EXPORT

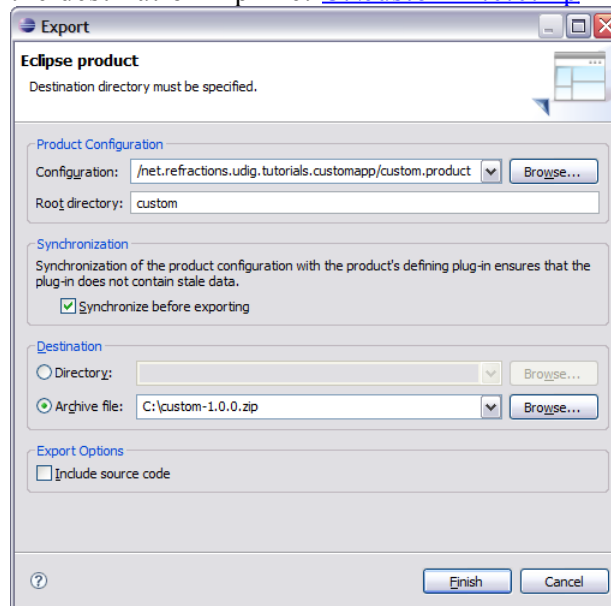
And now we are in a good position to export.

1. Open up a file browser and delete the “runtime directory”. This directory was when you were trying out the application earlier.
C:\java\runtime-custom.product
2. Return to the **custom.product** editor. Switch to the **Overview** tab
3. Click the **Eclipse Product export wizard** link to open up the Export wizard.
4. Change the **Root directory** field to custom.
5. Select the **Archive file** radio button and enter the destination zip file: [C:\custom-1.0.0.zip](#)

The runtime directory was created when you were trying out the application earlier.

Removing this directory has the same effect as clearing the configuration in the run dialog.

If the wizard fails a log file will be saved to the directory you indicate here.



6. Press the **Finish** button
7. Please wait: when finished custom-1.0.0.zip will be created.
8. Now unzip that folder and run your new application.

To use this application your end users will need Java installed along with the JAI and ImageIO extentions.

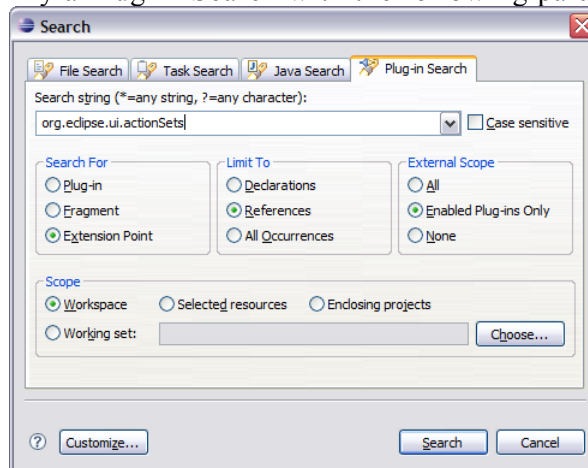
15WHAT TO DO NEXT

When searching for documentation please start with the Eclipse Help > Help Contents menu bar.

This Help Contents always matches the version of Eclipse you are using; and includes the uDig Developers Guide.

Here are some additional challenges for you to try. Each of these ideas results in a more professional looking application.

- Try a wider **about.gif** image and see what that looks like.
- Try making window icons with a transparent section.
- The eclipse Search (**Control-Shift-H**) can be used find all the available action sets.
Try a Plug-in Search with the following parameters.



- Add an “intro part” to your application to welcome new users. Remember to search in the eclipse help menu for documentation on how to do this – yes it is even better than trying to use google or koders.
- Add branding to the net.refractorions.udig_tutorials-feature feature. This is how you can get your name included in the about box.
- Externalize all the translatable strings in the **plugin.xml** and the **custom.product**. You can also create language specific **splash.bmp** and **about.gif** images.
- Manually add the **jre** folder from the SDK Quickstart to the generated custom-1.0.0.zip file. This will let users of your application unzip and run.
- Add platform specific icons for Macintosh and Linux