CreatingMappingUIPlugin

Your Visited Pages

CreatingMappingUIPlugin
CreatingAlgPlugin
PluginsForPrompt
Prompt

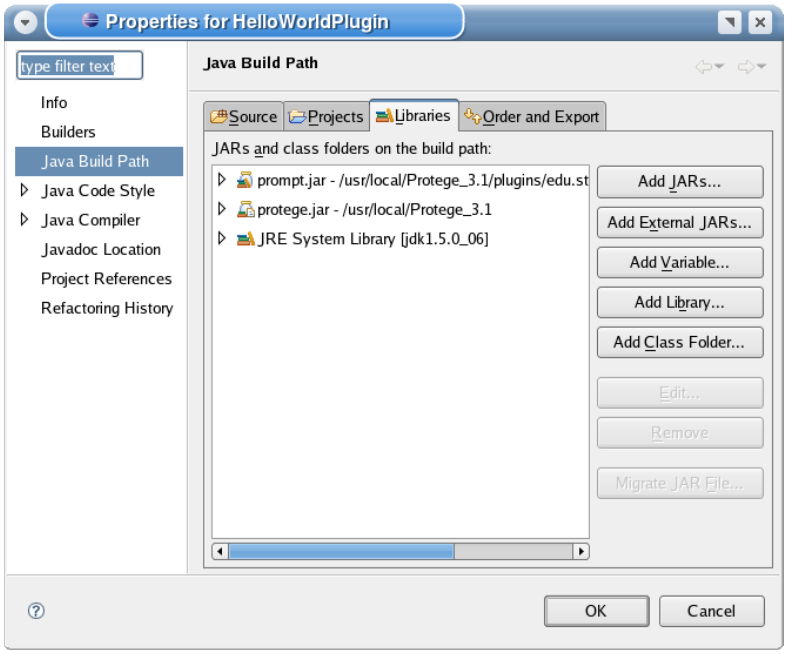
View Backlinks

Search

Creating a Mapping UI Plugin (7F8)

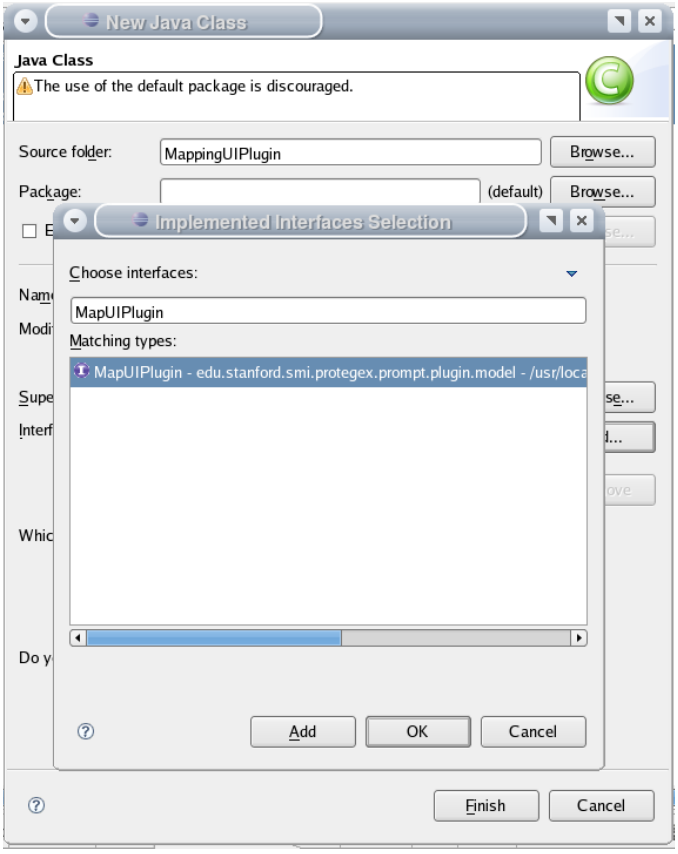
These instructions are written for Eclipse users, but someone not using Eclipse should be able to use them to create their own plugin. You can download the source code here: http://protege.cim3.net/file/pub/files/prompt/MappingUIPlugin.zip. (7FW)

- First begin by downloading Prompt, if you do not already have the latest version. (7FX)
- Open Eclipse and create a new Java Project, by clicking File - New - Project. Give the project the name, MappingUIPlugin. (7FY)
- Right click on your MappingUIPlugin in the Package Exlorer and click Properties. In the dialog that opens, click the Libraries tab, then click Add External JARs. In the file open dialog, find where you have prompt.jar, select it, and click Ok. Do the same for your protege.jar file. (7F9)



(7FA)

- Right click on your MappingUIPlugin in the Package Exlorer and create a new Class file. Call it, MappingUIExtension and in the Interfaces section click the Add button. In the Choose interfaces text box, type in MapUIPlugin and click Add, then click Ok. Finally, click the Finish button. (7FB)



(7FC)

- In your MappingUIExtension.java file, add the following methods if they were not created automatically by Eclipse. (7FD)

```
public String getPluginName() {nid 790}
public void invokePlugin() {nid 791}
public void afterLoad()
public void beforeClose()    (7FE)
```

Implementing the extension (7FF)

- Modify getPluginName. This function must return a valid string in order for your plugin to show up in Prompt. (7FG)

```
public String getPluginName() {
   return "Map UI Extension Tutorial";
}    (7FH)
```

- Modify afterLoad. This function is called after the mapping user interface is drawn. This is where you can add your custom UI components. In the following code, we add two tabs (one to the source panel and one to the target panel) to the interface, a button to the suggestion list, and a selection listener on the suggestion list. (7FI)

```
public void afterLoad() {
   // create a new source tab containing a JPanel object with the title "Our first tab"
   MapPluginFacade.addSourceDisplayTab(new JPanel(), "Our first tab");

   // create a new target tab containing a JPanel object with the title "Our second tab"
   MapPluginFacade.addTargetDisplayTab(new JPanel(), "Our second tab");

   // get the suggestion list container object
   SelectableContainer suggestionContainer = MapPluginFacade.getSelectableContainer(MapPluginFacade.UI_UTILITY_SUGGESTION);

   // add a new button to the suggestion list header with the tooltip "Our action button"
   suggestionContainer.addHeaderButton(new CustomAction("Our action button"));

   // add a custom selection listener to the suggestion container's list
   suggestionContainer.addSelectionListener(new CustomSelectionListener());
}    (7FJ)
```

- The above code will most likely cause Eclipse to complain about some compile errors and that is because we have not yet implemented our CustomAction or CustomSelectionListener classes. Let's begin by creating the CustomAction class. At the bottom of your MappingUIExtension.java file, after the last curly brace, copy and paste the following code. (7FK)

```
class CustomAction extends AbstractAction {
   public CustomAction(String name) {
      super(name);
   }

   public void actionPerformed(ActionEvent e) {
      // say hello when this action takes place
```

```
      JOptionPane.showMessageDialog(null, "Hello, world!");
   }
}    (7FL)
```

- Below the CustomAction class, copy and paste the following to create the CustomSelectionListener class. (7FM)

```
class CustomSelectionListener implements SelectionListener {
   /**
    * Called when a suggestion is selected in Prompt
   */
   public void selectionPerformed(Object arg0) {
      // convert the selected object to an Operation
      Operation selectedOperation = (Operation)arg0;

      if (selectedOperation != null) {
         // get the arguments of the operation
         ActionArgs args = selectedOperation.getArgs();
         if (args != null) {
            Object o1 = args.getArg(0);
            Object o2 = args.getArg(1);

            // if the arguments are of type Cls, display their names
            if(o1 instanceof Cls && o2 instanceof Cls) {
               Cls c1 = (Cls)o1;
               Cls c2 = (Cls)o2;

               JOptionPane.showMessageDialog(null, c1.getName() + " mapped to " + c2.getName());
            }
         }
      }
   }
}    (7FN)
```

Creating a manifest file (7FO)

- The manifest file is used by Prompt in order to load your plugin. If this file does not exist, the plugin will not be loaded. (7FZ)
- In Eclipse, right click on your HelloWorldPlugin project in the Package Explorer and create a new folder called META-INF. (7G0)
- Right click on the newly created META-INF folder, and create a new file called MANIFEST.MF. (7G1)
- Open MANIFEST.MF and add the following lines: (7G2)

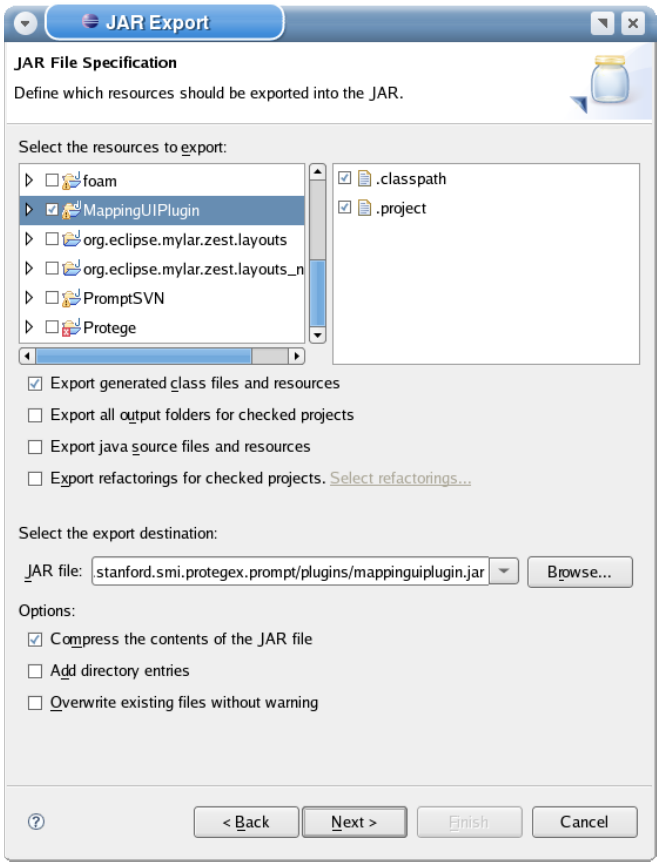
```
Manifest-Version: 1.0 {nid 79F}

Name: MappingUIExtension.class
Map-Extension: True    (7FP)
```

- This will tell Prompt the name of your plugin class and also the type of extension this plugin is. (7G3)

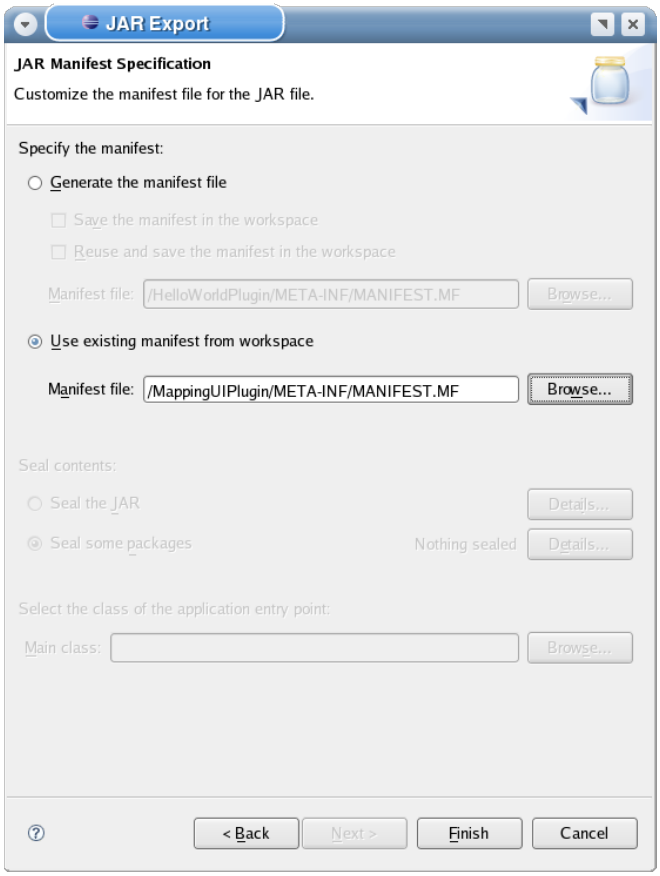
Installing the plugin (7FQ)

- Right click on your MappingUIPlugin project in the Package Explorer and choose Export. (7G4)
- Select under Java the JAR file option. (7G5)
- Select the check mark by the MappingUIPlugin project if it is not already selected. (7G6)
- Click Browse, and navigate to your Prompt install directory. Create a new directory and call it plugins if this directory does not already exist. In the Name field, call your jar file mappinguiplugin.jar. (7FR)



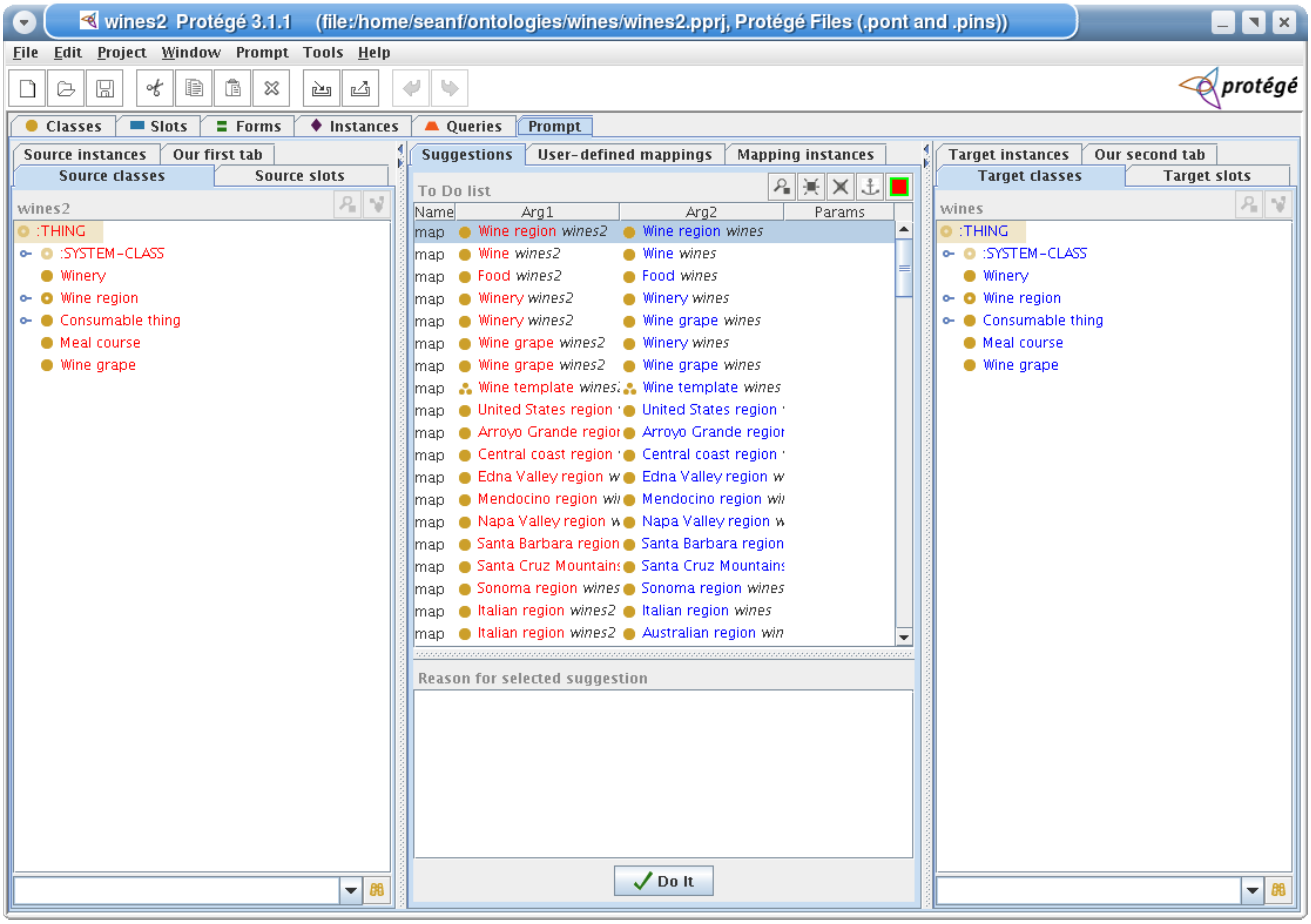
(7FS)

- Click Next twice on the JAR Export dialog. (7G7)
- You should now see a JAR Manifest Specification screen. Select the Use existing manifestor from workspace radio button and Browse to your MANIFEST.MF file. (7G8)
- Finally click Finish. (7G9)



(7FT)

- You should now be able to run Protege, open a project, open Prompt and select the Map option. Select a target ontology and click the "Click here to begin" button. This will take you to the mapping interface where you should see an image similar to the one below. (7FU)



(7FV)

View other revisions
Last edited August 1, 2006 10:57 (diff)

RSS