How to publish a Fairmat plug-in

Revision #2

Date: 2012/07/20

Fairmat plug-ins development

How to publish a plug-in

Index

1 Introduction

This tutorial shows how to build from scratch and how to publish a plug-in for Fairmat Academic.

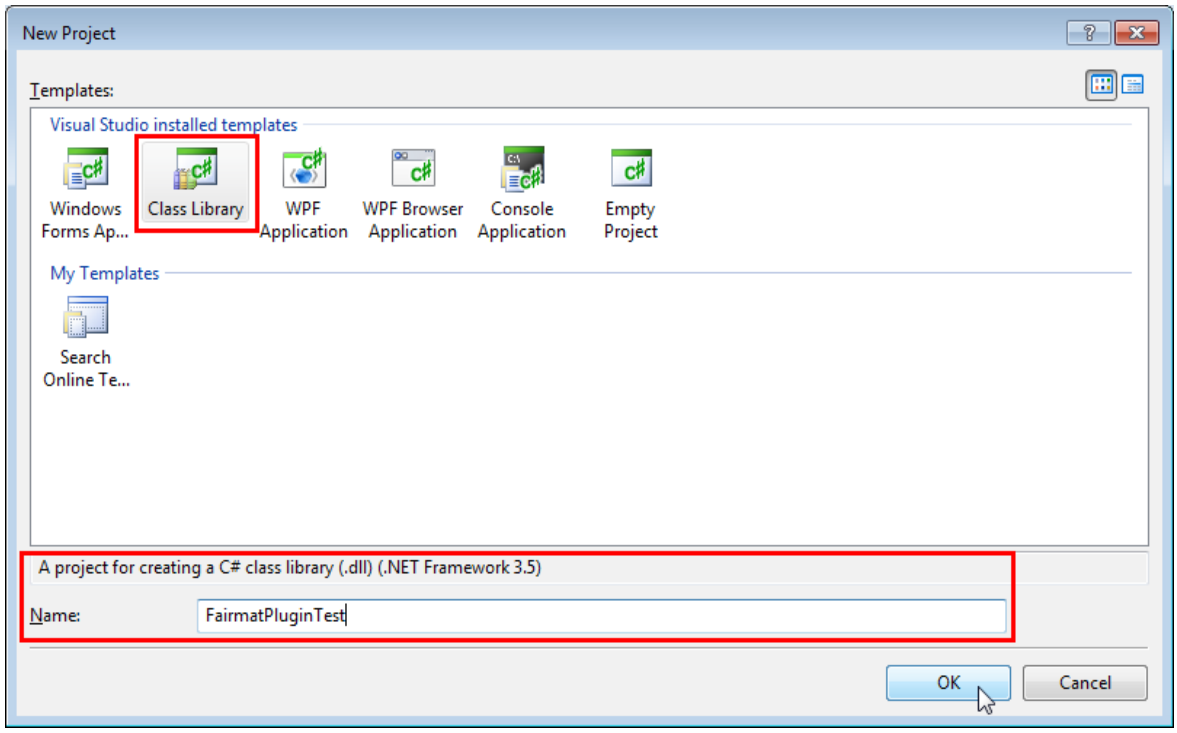
2 Setup a development environment

For this tutorial we used Windows and the Visual C# 2010 Express Edition IDE, but note that the same things can be done using other systems or other IDEs such as Mono Develop (http://www.monodevelop.com) or Sharp Develp (http://www.icsharpcode.net/OpenSource/SD/).

To download *Visual C# 2010 Express Edition*, launch your web browser and go to this address: http://www.microsoft.com/Express/Download/ and choose "Visual C# 2010 Express", download and install it. **You should register your copy** of *Visual C# 2008 Express Edition* within 30 days. The product is free, and registration is free, but Microsoft does require you to register it in order to use it for more than 30 days.

3 Project setup

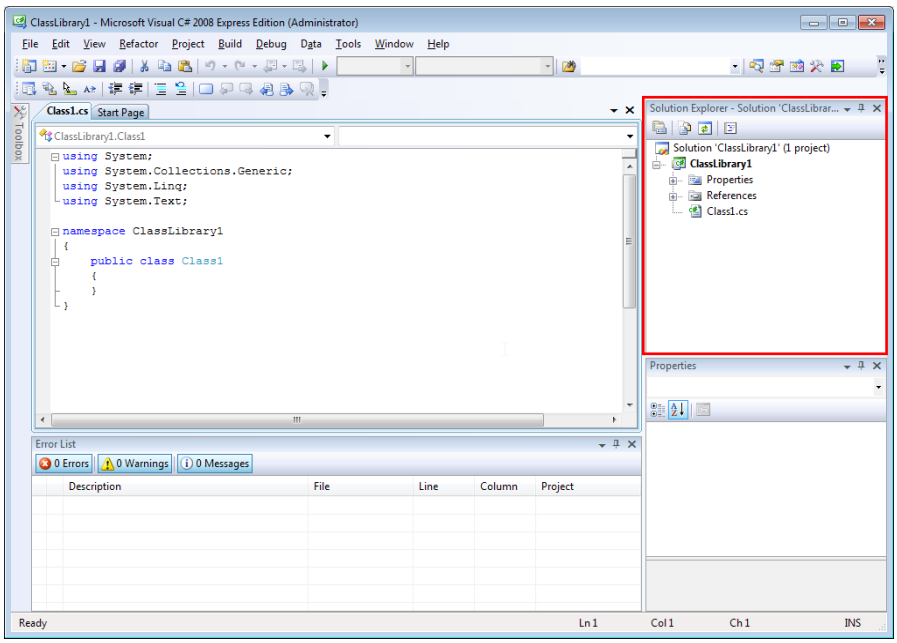
Start Visual C# 2010 Express Edition create new Project and select a Class Library type project, insert the name (eg. "FairmatPluginTest") in the field "Name" and press OK Button



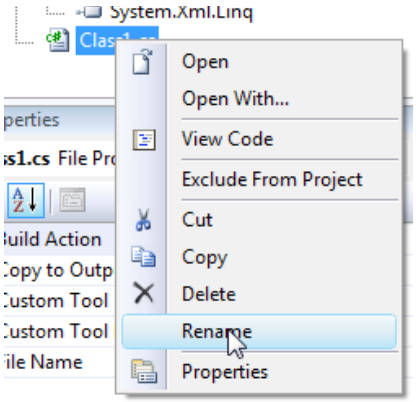
Note:

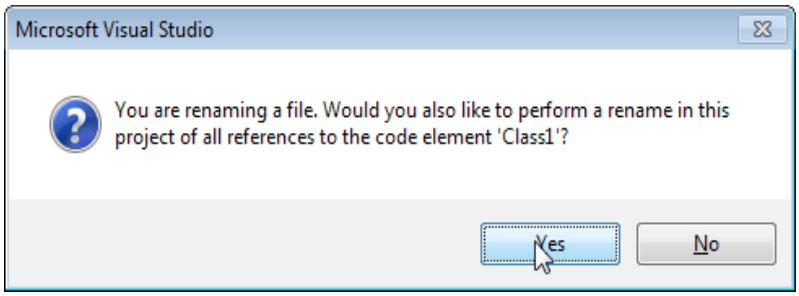
Visual C# 2010 Express Edition uses the .NET Framework 4.0 as default, which is fine, while Visual C# 2012 RC Express Edition uses the .NET Framework 4.5. In this latter case you need to change the target framework to .NET 4.0.

Now, in "Solution Explorer" tab

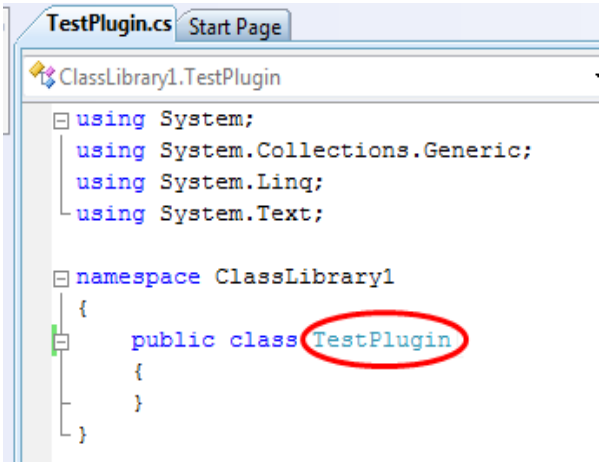


righ-clicking on “Class1.cs”, and selecting “Rename” from the menu, Rename the file (eg. “TestPlugin.cs”), then press Enter key and, when system asks, confirm to change the class name:



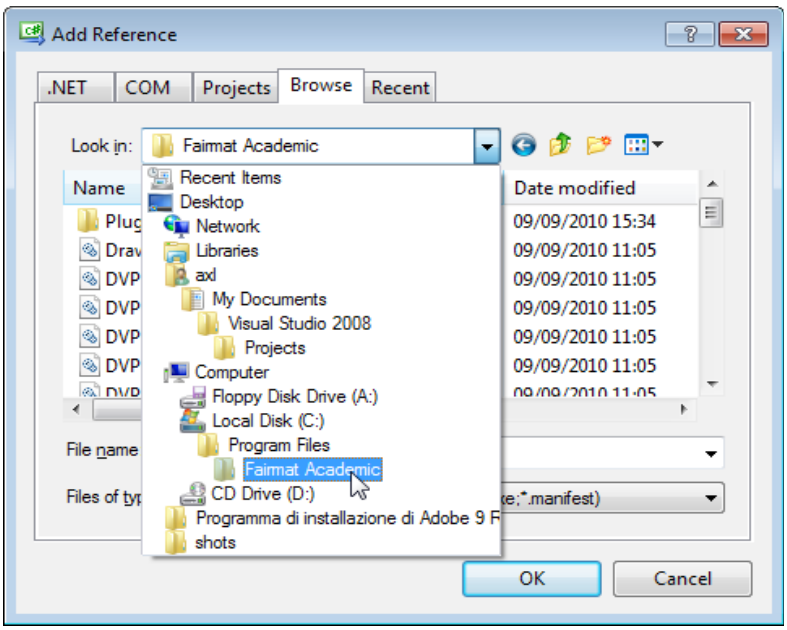


Click OK and you can view the changes in Code tab:

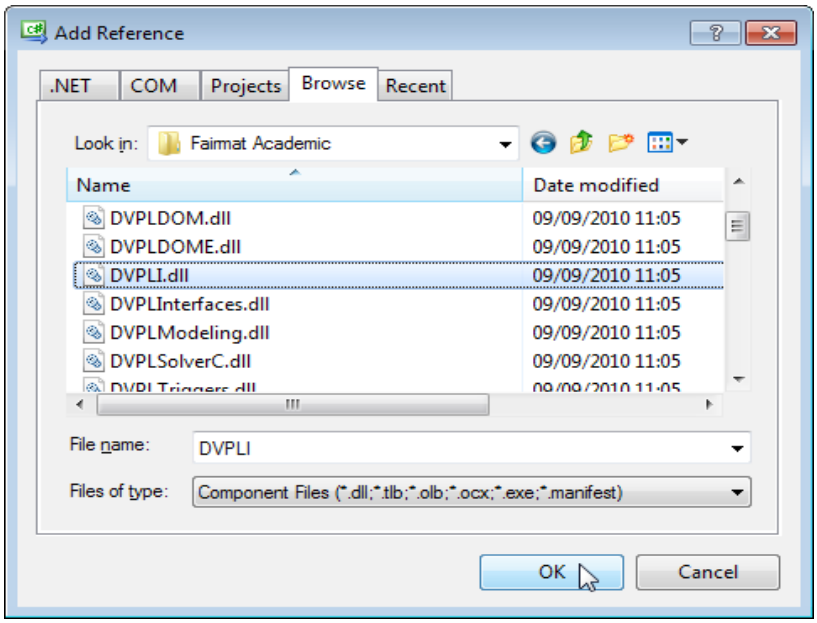


Now you must reference Fairmat Libraries, to do this right-click on “References” in Solution Explorer and select “Add Reference” and then select “Browse” tab and in “Look in” field select

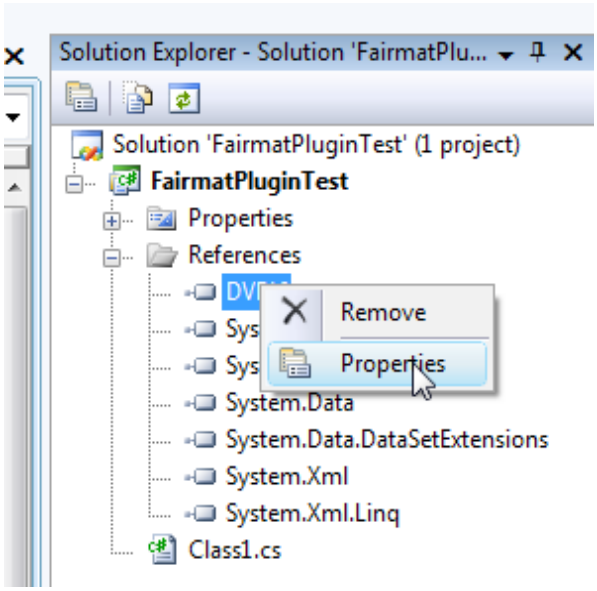
"C:\Program Files\Fairmat Academic" directory

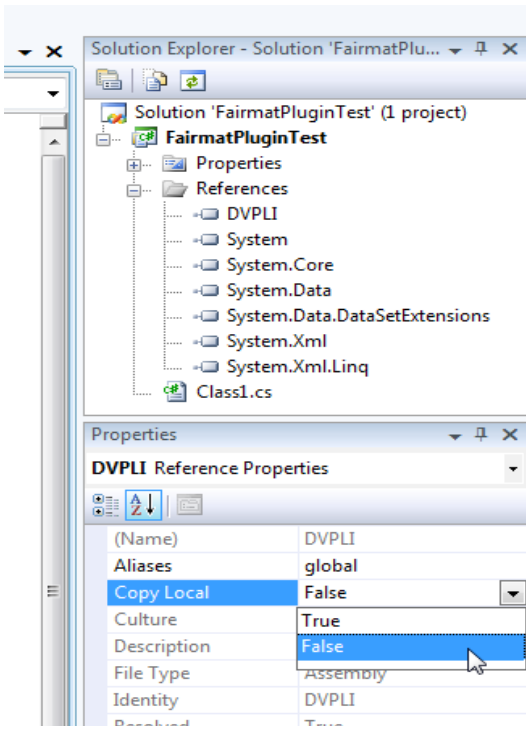


Select DVPLI.dll library and press OK button



In Solution Explorer, right-click on "DVPLI.dll" and select "Properties"





Below Solution Explorer tab you can see the Properties tab that shows properties of DVPLI.dll library, change “CopyLocal” property to False, because this library is already shipped with software (in future versions it will be signed and will be available on the GAC)

As you did with DPLI.dll, you may reference also DVPLDOM.dll, Mono.Addins.dll, DVPLInterfaces.dll libraries.

4 Writing the plug-in

Begin to code a simple plug-in that adds a constant to Fairmat workspace:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using DVPLI;
using DVPLDOM;
using Mono.Addins;
using System.Windows.Forms;

[assembly:Addin("FairmatPluginTest","1.0",Category="Test")]
[assembly:AddinDependency("Fairmat","1.0")]

namespace FairmatPluginTest
{
   [Extension("/Fairmat/UIConstant")]
   public class TestPluginConstant : IUIConstant
   {
      #region IUIConstant Members

      public double Value
      {
         get { return 3.1415; }
      }

      #endregion

      #region ISymbolDefinition Members

      public string Description
      {
         get { return "PI Constant"; }
      }

      public string Name
      {
         get { return "PI"; }
      }

      #endregion
   }

}
```

Now that we have a complete program, let's look at the code to see what each part is for—all of the pieces are things you'll deal with every time you write in C#. Starting from the top, the code has several lines beginning with using:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

These *using directives* help the C# compiler work out what external code this particular source file will use. No code is an island—to get any useful work done, your programs will rely on other code. All C# programs depend on the .NET Framework class library.
Using directives can declare an intent to use classes from any library—yours, Microsoft's, or anyone's. All the directives in our example start with System, which indicates that we want to use something from the .NET Framework. This text that follows the using keyword denotes a *namespace*.

Fairmat plugins need to reference the following libraries:

```
using DVPLI;
using DVPLDOM;
using Mono.Addins;
using System.Windows.Forms;
```

Fairmat extensions (plug-ins) uses Mono.Addins extension model that is based on four concepts:

- **Add-in host**: an application or library which can be extended by add-ins. Extension possibilities are declared using extension points.

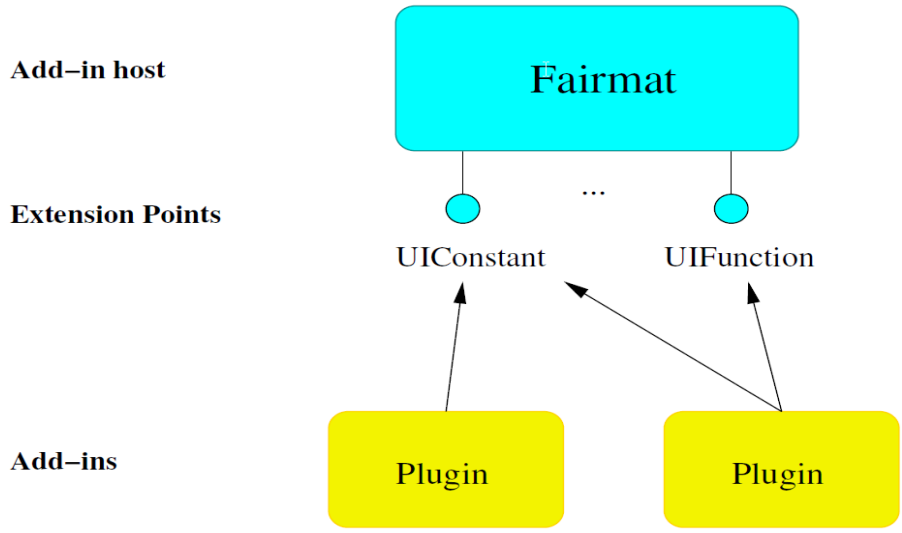
- **Extension point**: a placeholder where add-ins can register extension nodes to provide extra functionality. Extension points are identified using type names or extension paths.

- **Extension node**: an attribute-decorated element that describes an extension. Extension nodes are typed. Extension points may declare which types of extension nodes do they accept.

- **Add-in**: An assembly and/or other files which register new nodes in one or several extension points defined by add-in hosts. An add-in can also act as an add-in host, and as such it can be extended by other add-ins.

Mono.Addin defines an Add-in Description Model, which is used by add-ins and add-in hosts to declare all extensibility information. Add-in descriptions can be represented either using an XML manifest, or by applying custom attributes to assemblies and types.

Mono.Addins also provides an API (implemented in Mono.Addins.dll) which can be used at run-time to query and handle add-in extensions



Add-ins have to be marked with the **[Addin]** attribute to be recognized as add-ins. Add-ins must also declare the add-in hosts they are extending by using the **[AddinDependency]** attribute. Notice that an add-in can extend several add-in hosts, and it can even extend other add-ins which declare their own extension points. The AddinDependency attribute must specify the host/add-in id and its version number.

In our example code we can see the lines:

```
[assembly:Addin("FairmatPluginTest","1.0",Category="Test")]
[assembly:AddinDependency("Fairmat","1.0")]
```

where the first line declares that we are implementing a plug-in (extension) with following properties:

```
Name: "FairmatPluginTest"  (Plugin Name)
Version: "1.0"             (Plugin Version)
Category: "Test"           (Plugin Type)
```

and the second line declares that the plug-in depends on add-in host implemented in Fairmat.

The plug-in's category output can be defined by the user and represents the extension type, actually Fairmat use following values:

- Core [reserved value]
- Modeling
- Stochstic Process
- Calibration

By applying the **[Extension]** attribute to a class we are declaring a new extension node of that type which is added to an extension point. The extension point is determined by looking at the base types and interfaces of the class. So in this example the class implements the ICommand interface, and there is an extension point defined for that interface, so that's the extension point where the type will be registered. If a class implements several interfaces, we can select the extension point by specifying the type name in the attribute.

In our plug-in example we extend "/Fairmat/UIConstant" node using:

```
[Extension("/Fairmat/UIConstant")]
public class TestPluginConstant : IUIConstant
```

Note that the extension class must implement methods and properties defined by IUIConstant interface associated to "/Fairmat/UIConstant" extension node.

In documentation reference (http://www.fairmat.com/software/DVPL.web) you can see that IUIConstant interface defines only the property "Value" that contain the numerical value for the constant symbol we are defining and inherit from interface ISymbolDefinition these properties:

- Description
- Name

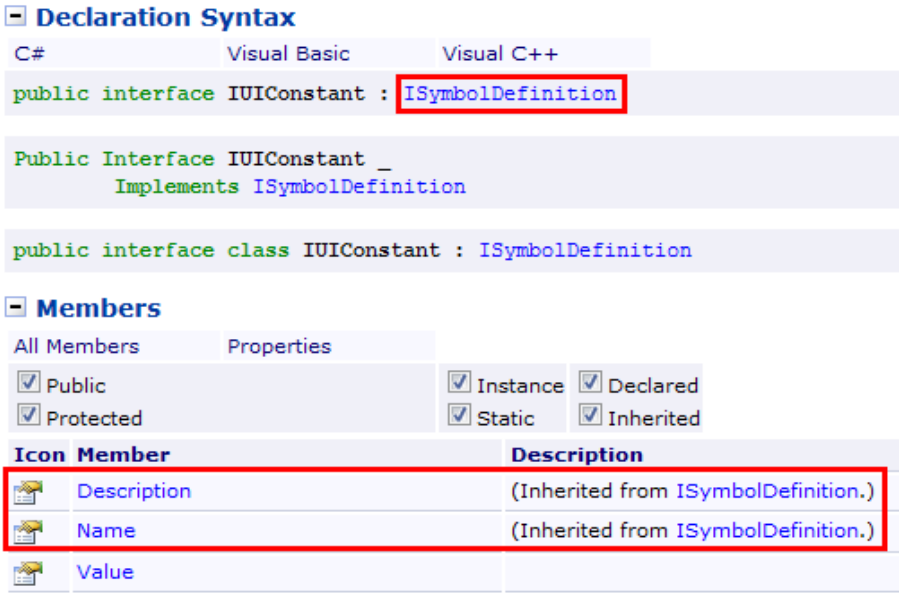
that define name and description of the symbol that represents our constant in the Fairmat workspace.

Note:

You have to add the following lines:

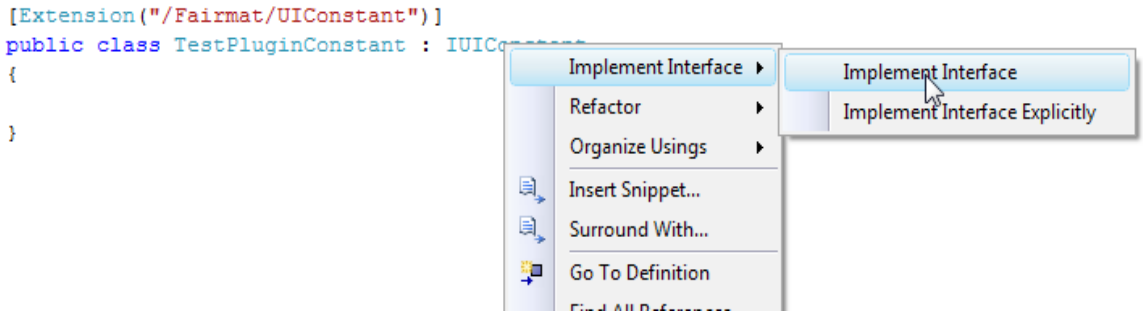
```
[assembly:Addin("FairmatPluginTest","1.0",Category="Test")]
[assembly:AddinDependency("Fairmat","1.0")]
```

once for each assembly, alternatively you can add these to AssemblyInfo.cs file if present.



4.1 Programming Tricks

You can obtain a skeleton implementation of properties and methods defined in an interface by right-clicking on the interface name in the class declaration and selecting "Implement Interface" as depicted in figure:



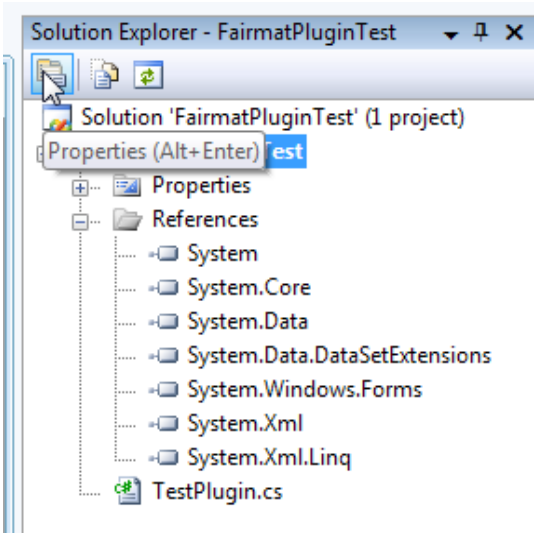
This is the result:

```
[Extension("/Fairmat/UIConstant")]
public class TestPluginConstant : IUIConstant
{

    #region IUIConstant Members

    public double Value
    {
        get { throw new NotImplementedException(); }
    }

    #endregion

    #region ISymbolDefinition Members

    public string Description
    {
        get { throw new NotImplementedException(); }
    }

    public string Name
    {
        get { throw new NotImplementedException(); }
    }

    #endregion
}
```

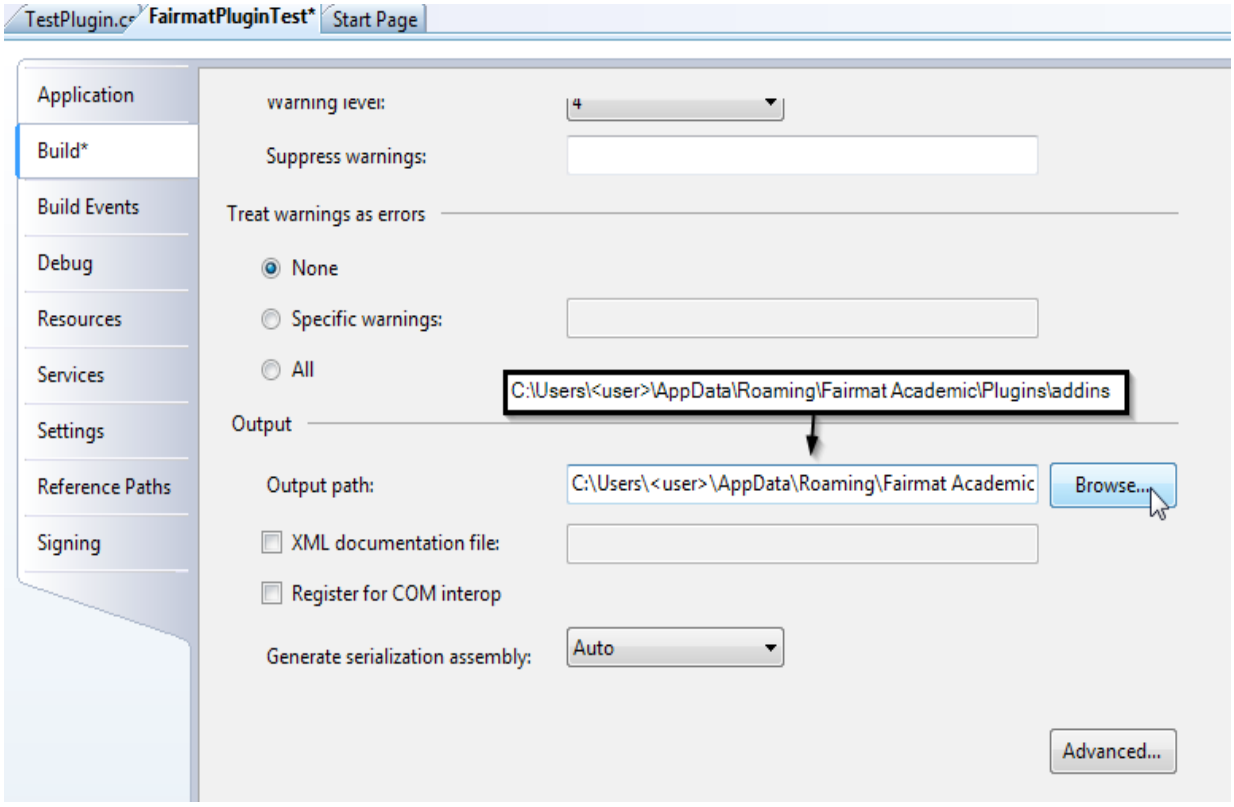
5 Build & Deployment

5.1 Testing setup

To test your plug-in you have to copy the generated plug-in in the Fairmat plug-in directory which is "C:\Program Files\Fairmat Academic\Plugins" or setup your development environment to do this programmatically for you using Project Properties:



Select "Build" tab, set the field "Output path" to:
"C:\Users\<user>\AppData\Roaming\Fairmat Academic\Plugins\addins"
(where <user> is your user name) and save the project.



If you have modified the project output path, you can build it simply by pressing F6 keyboard otherwise you must copy the generated *.dll to path “C:\Users\<user>\AppData\Roaming\Fairmat Academic\Plugins\addins” manually.

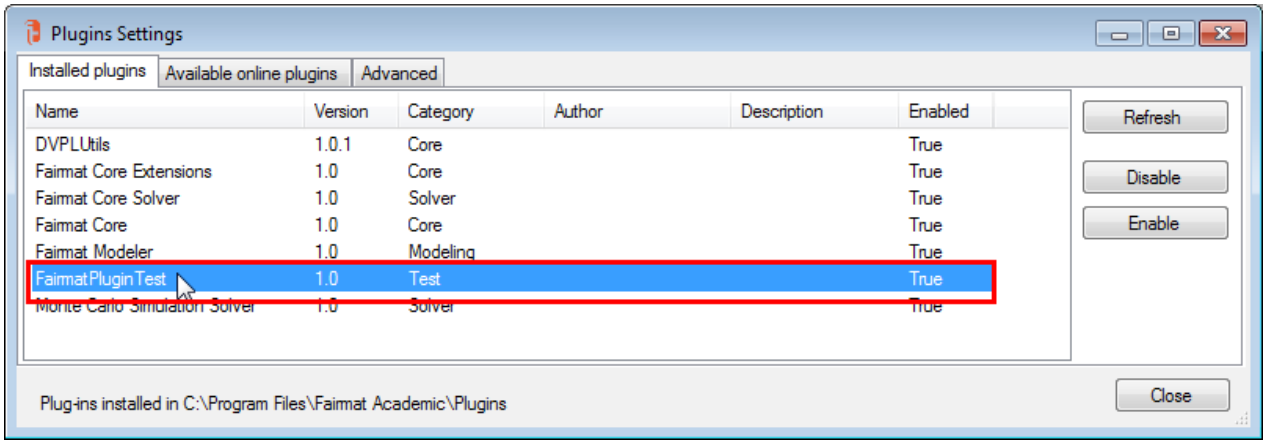
Note that the folder "AppData" is a hidden folder and By default and Windows 7 does not show hidden files.
If you need to manage files under a hidden folder, you'll need to be able to view that folder.

Here's how:

1. Type folder options in the search box after clicking Start.
2. Choose Folder Options under Control Panel from the list of results and
3. Click on the View tab in the Folder Options window.
4. In the Advanced settings: area, locate the Hidden files and folders category.
5. You should also see two options under the folder.
6. Choose the Show hidden files, folders, and drives radio button under the Hidden files and folders category.
7. Click OK at the bottom of the Folder Options window.

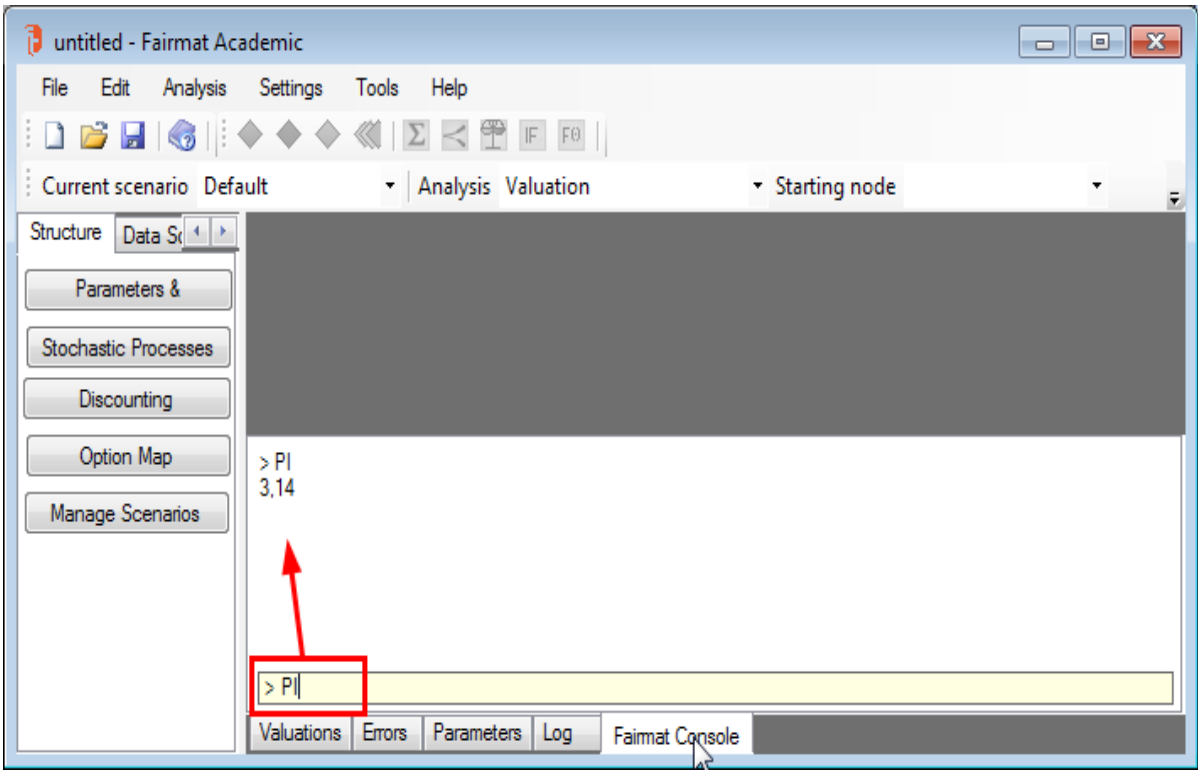
5.2 Testing the plug-in

Run Fairmat, open or create a new document and in the "Settings" menu you select "Plugin Settings", the opened window shows the plugin list where you can check if your new plug-in was loaded



Select the Fairmat Console Tab, insert the name of the constant defined in our plug-in and press the

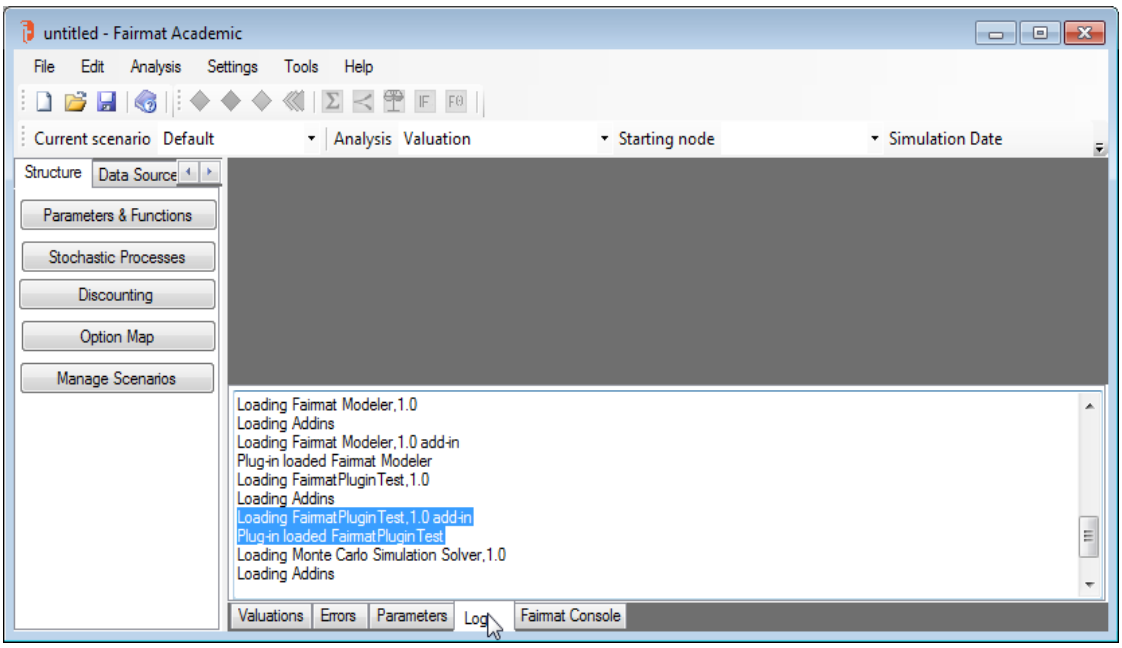
Enter Key.



If all works correctly, you can see on the video the constant value we previouslly defined.

5.3 Notes on Fairmat Log:

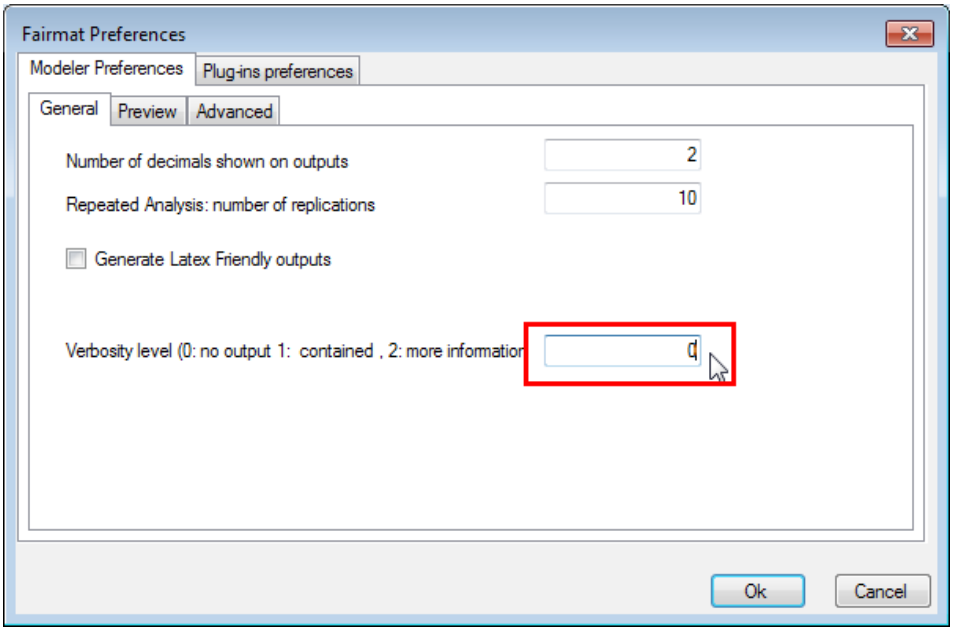
All Fairmat programmatic calls to methods Console.Write or Console.WriteLine are redirected to Fairmat Log tab.



In order to control the amount of outputs a plug-in should write, you can condition quantity of output's to be written by checking the integer value of DVPLI.Engine.Verbosity defined in Fairmat using for example the following code block:

```
if(Engine.Verbose >= threshold)
  {
     Console.Write("Comment...");
  }
```

where "threshold" is an integer value that can be: 0 (no output),1 (contained) or 2 (verbose)
You can control the value of this variable from the user interface, selecting "Fairmat Settings" in "Settings" menu.



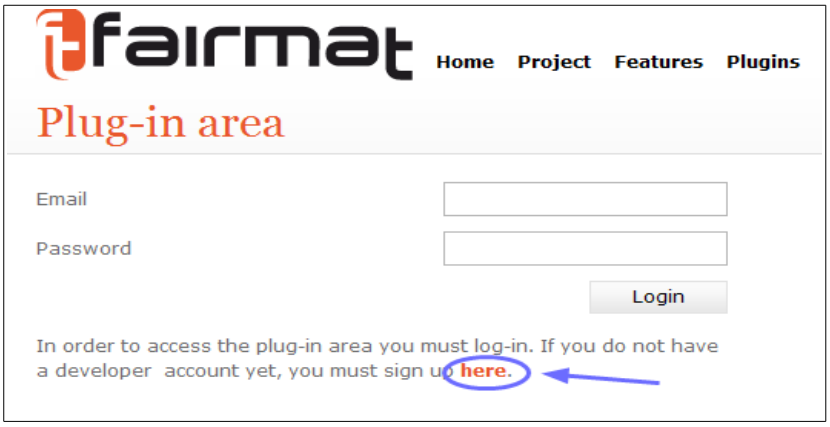
6 Publish the plug-in in the Fairmat website

If you think your plug-in can be useful to others, you can publish it in the Fairmat website and if your plug-in is open-source, you may join the Fairmat revenue sharing program agreement (see http://www.fairmat.com/revenue-sharing/ for more details).

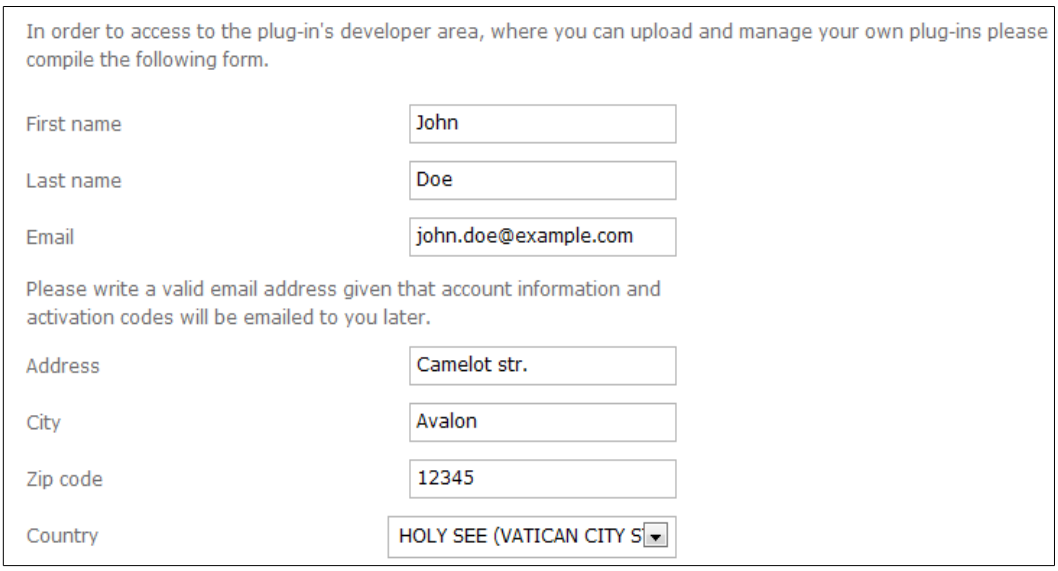
Note that the functionalities described below are not being ported to our new system, if you want to publish a plug-in just contact us.

In order to officially release a plug-in you have to go to Fairmat web site at address: http://www.fairmat.com and register as developer selecting "Plug-ins - area" on the "Developers" menu.

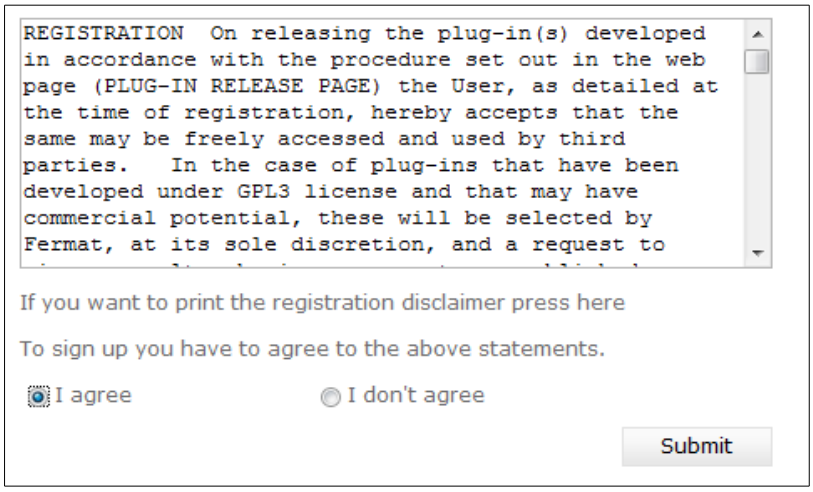
After that you can login to developers area or if you don't have a developer account you have to sign up by clicking the link depicted in figure:



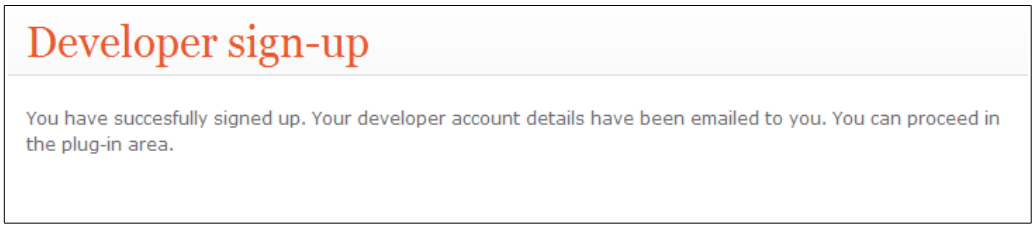
To sign up you have to fill the registration form



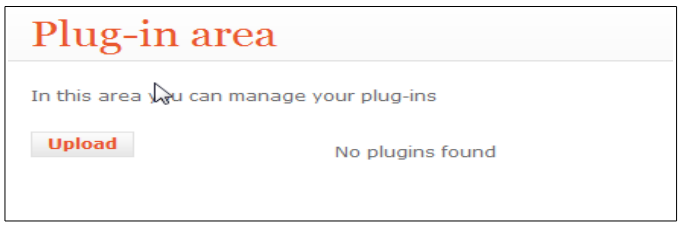
you have to agree on the license and press the "Submit" button



then you can see the registration confirmation and a mail with password is sent automatically to the mail address you provided in registration form.



Once you are logged in you can manage and upload your plug-ins



In order to upload your plug-in, you have to prepare the following packages:

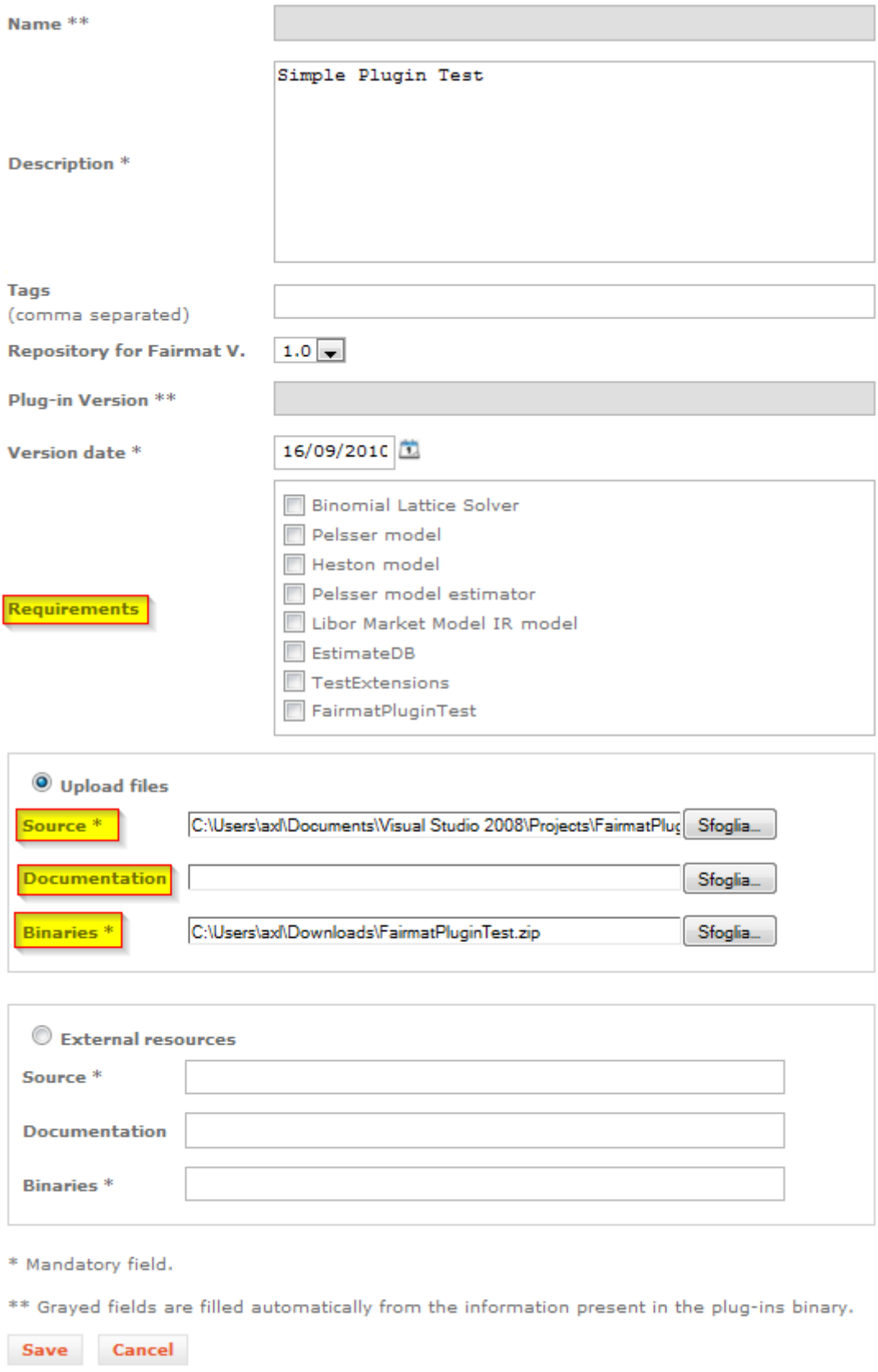
- Binary Package: after compilation, go to your project's binary folder and create a zip file containing the plug-in libraries and its dependencies. Note that the system will not accept zip archives containing more than on plug-in. This means that if you have more than one assembly containing the attribute assembly:Addin you have to separate them in different archives.

- Source Package: clean the project and create a zip file containing the entire solution.
  If you're using VC# 2008 Express Edition, then the Clean commands are not on any menus by default. You can customize your menus by right-clicking on empty space on either the menu bar or toolbar area, and selecting "Customize...". You'll be able to search for the Clean commands in the dialog that appears, and then you can drag-and-drop it to the menu of your choice.

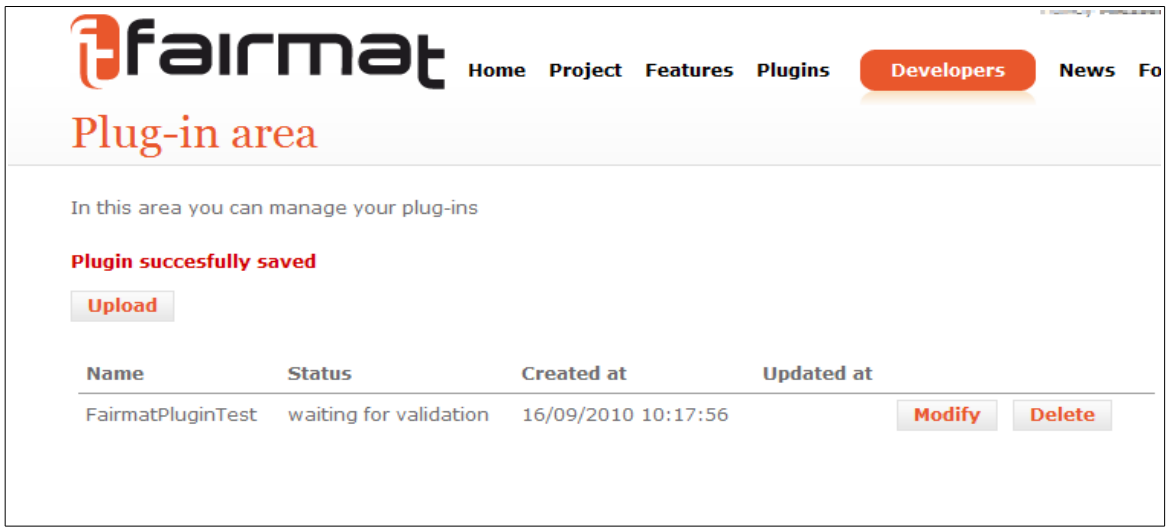
- Documentation: you can add a Pdf or a text file which explains how to use your plug-in in Fairmat.

In the “Requirements” area you must select the packages that are required for the execution of your plugin and that will be automatically provided when download it.

Press upload button, then compile the fields “Description” and “Version Date” of the resulting form and press "Save" button.



When your plug-in is uploaded correctly, you can see the confirmation:



Notes:

Your plug-in have to wait for validation from Fairmat Srl before it would be available to the others. You can check this on the plug-in status area in your personal page.