Developer documentation

Wollenburger Antoine

March 31, 2013

Chapter 1

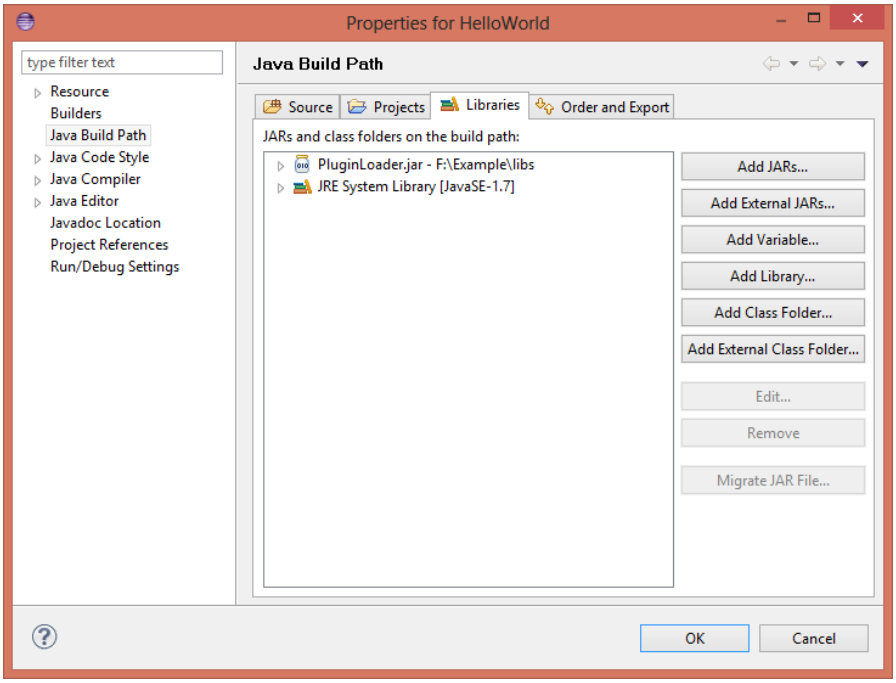
Create your first plugin

The first thing you need is a new project. Name it at your conveniance. For the example, we name it "HelloWorld".

1.1 Fix the Java Build Path

Secondly add "PluginLoader.jar" to the Java Build Path (right clic on th project, Properties, Java Build Path).

Figure 1.1: Fixing Java Build Path



1.2 Saying "Hello World"

Before creating a new class, you must create a new package. In this example we take "com.ornicare.helloworld". Then create a new class and make it an extend of PluginRunnable and create a run method : it's the hook for the platform. It must look like this :

```
package com.ornicare.helloworld;

import com.space.plugin.PluginRunnable;

public class MainClass extends PluginRunnable {

        @Override
        public void run() {
                System.out.println("Hello world !");
        }
}
```

1.3 Generate the propertie file

Create a new file named "plugin.properties" into the project. In this file add :

- The plugin name : "name = HelloWorld"
- The hook's class path : "main = com.ornicare.helloworld.MainClass"
- The runnable attribut : "runnable = true"

1.4 Exporting your first plugin

To run this, you need to create a jar of your plugin : "File > Export > Java/JAR File". Put the created jar into the plugin directory and run the plugin loader.

Chapter 2

Intricated plugins

2.1 Using another plugin

For this example, we are going to use the "CConsole" plugin : it's just a plugin to display the java console (and few other things). So put "CConsole.jar" into the plugin directory. Add it to your project Java Build Path. And use it. For example :

```
package com.ornicare.helloworld;

import cconsole.CConsole;

import com.space.plugin.PluginRunnable;

public class MainClass extends PluginRunnable {

        @Override
        public void run() {
                CConsole.load();
                CConsole.println("Hello world !");
        }
}
```

Then you need to add the following line to your properties file : "depend = CConsole".

Chapter 3

Make a regular “plugin.properties”

You are allowed to use the following attributes :

- The plugin name : “name = your_plugin_name”
- The hook’s class path : “main = package.hook_class_name”
- The runnable attribut : “runnable = true\false”
- If you want a single instance : “singleton = true\false”
- Make your plugin lazy : “lazy = true\false”
- Indicate dependencies to others plugins : “depend = plugin1, plugin2, ...”

Chapter 4

Advanced functions

4.1 Advanced functions for all plugins

4.1.1 getPlugin

4.1.2 getPluginList

4.2 Advanced functions for a runnable plugin

4.3 Advanced functions for a content provide plugin

4.3.1 getObject