Creating Plug-ins Using NetBeans - a Quick-Start Guide

Introduction

NetBeans 7.1 was used here. Other versions may do things just a bit differently.

This document assumes that you already have IGB sources checked out and NetBeans set up. If not, go see Compile and run IGB from the command line.

Create a new plug-in

- Set IGB as your NetBeans Main Project by right clicking on IGB in your projects panel.
- File > New Project... and then In the ensuing wizard:
    1. select NetBeans Modules | Module, enter a description and then (Next>)
    2. Choose a name etc. and then (Next>)
    3. In this third step of the wizard check the "Generate OSGi Bundle" checkbox and then (Finish).
- File > New File... and then in the ensuing wizard:
    1. Category: Module Development; File Type: Installer / Activator and then (Next>)
    2. (Finish) and Installer.java will be created.
- Open Installer.java and make it look like this (You could make it even smaller by leaving out the context stuff and the printf statements.):

```
package com.gene.osgi;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

/**
 * Say hello upon starting, and goodbye when stopping.
 */
public class Installer implements BundleActivator {

    static BundleContext context;

    @Override
    public void start(BundleContext c) throws Exception {
        context = c;
        System.out.printf("%s: starting...\n", this.getClass().getSimpleName());
        javax.swing.JOptionPane.showMessageDialog(null,
            "Hello, World.",
            this.getClass().getSimpleName(),
            javax.swing.JOptionPane.INFORMATION_MESSAGE);
    }

    @Override
    public void stop(BundleContext c) throws Exception {
        javax.swing.JOptionPane.showMessageDialog(null,
            "Goodby, World.",
            this.getClass().getSimpleName(),
            javax.swing.JOptionPane.INFORMATION_MESSAGE);
        System.out.printf("%s: stopping...\n", this.getClass().getSimpleName());
        context = null;
    }

}
```

Notice that I use a Swing dialog box. Some frameworks (notably Felix) do not automatically import javax classes. So to get Swing imported I had to edit the module manifest. (NetBeans will put this in META-INF/MANIFEST.MF.) Under the Projects tab, open "Module Manifest" in the "Important Files" folder. Add "`, javax.swing`" to the `Import-Package:` line like so:

```
Manifest-Version: 1.0
Bundle-Activator: com.gene.osgi.Installer
Bundle-Localization: com/gene/osgi/Bundle
Bundle-Name: %OpenIDE-Module-Name
Bundle-SymbolicName: com.gene.osgi
Bundle-Version: 1.0
Import-Package: org.osgi.framework, javax.swing
```

At this point you should be able to run it from NetBeans, but I had trouble doing so. There were a few errors and things hung when I tried to quit, but I did see the "Hello, World" greeting dialog. Also it *does* work with felix and with IGB.

To get it to work with IGB you will have to use the command line with the install bundle option. For example:

```
$ ./run_igb.sh -install_bundle file:../../NetBeansProjects/HelloOSGi/build/cluster/modules/com-gene-osgi.jar
```

Registering

To make a plug in that IGB can actually use we will try to add a track operator similar to `MyTransformer` in Bundles and Plug-Ins. To do this we will need the com.affymetrix.genometryImpl.operator package which contains an Operator class. So open a new project and open it's properties. You will add genometry.jar as a wrapped jar.

So we create an Activator (which NetBeans likes to name Installer.java) as before, but we add registration code and a new file which extends Operator. It is this new file that needs to be registered and will do the work. The new file is created as a regular java class (not a Module Activator). Let's call it MinOperator.

External Reference

- OSGi and NetBeans on the NetBeans Wiki