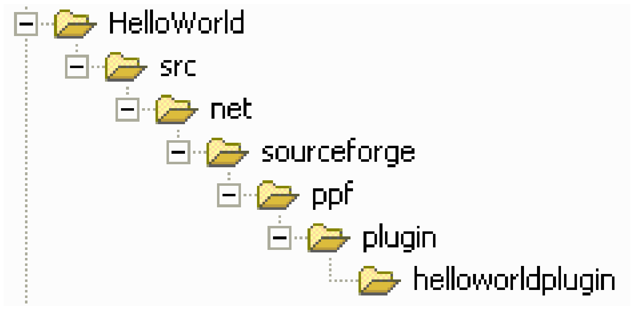
Create A PPF Plugin - Hello World

1. Description

We will create a plugin that will respond to someone typing *hello* in a channel with the famous reply of *Hello World!*. Let's call this plugin *HelloWorld*. From this exercise, you will see how to create the basic parts of a plugin, and learn how to use the dynamic reloading of plugins for a quicker development cycle. It is assumed that your PPF is currently running.

2. Create the directory structure

All plugins are created in the system directory under the main PPF install directory. So, in system, create a directory named **HelloWorld**. We will also need a place to keep the source file(s) as well, so in the newly created HelloWorld directory, create the following path: **src/net/sourceforge/ppf/plugin/helloworldplugin**



Hello World directory structure

3. Create the build file

Compiling and packaging the plugin is controlled using an ANT build script. This script is called **build.xml** and is placed in the HelloWorld directory.

```
<?xml version="1.0"?>

<project basedir="." default="deploy">

    <property name="plugin.name" value="HelloWorld"/>

    <path id="plugin.classpath">
      <fileset dir=".">
        <include name="*.jar"/>
      </fileset>
    </path>

    <!-- Call the PPF build script to perform the build.  Can set the plugin specific
      details here and keep the PPF classpath in one place -->
    <target name="compile">
      <ant antfile="../PPF/build.xml" target="compile.plugin" inheritRefs="true"/>
    </target>

    <target name="deploy" depends="compile">
      <jar destfile="./${plugin.name}.jar" basedir="bin"/>
    </target>

    <target name="release">
```

```
      <!-- Copy plugin JAR and required libs -->
      <copy todir="${build.system.dir}/${plugin}">
        <fileset dir="../${plugin}">
          <include name="*.jar"/>
        </fileset>
      </copy>
    </target>
</project>
```

4. Configure the plugin to be available to PPF

The core must be made aware of the plugin, so we need to add a new *plugin* section to PPFConfig.xml.

```
  <plugin load="yes">
    <name>HelloWorld</name>
    <classname>net.sourceforge.ppf.plugin.helloworldplugin.HelloWorldPlugin</classname>
  </plugin>
```

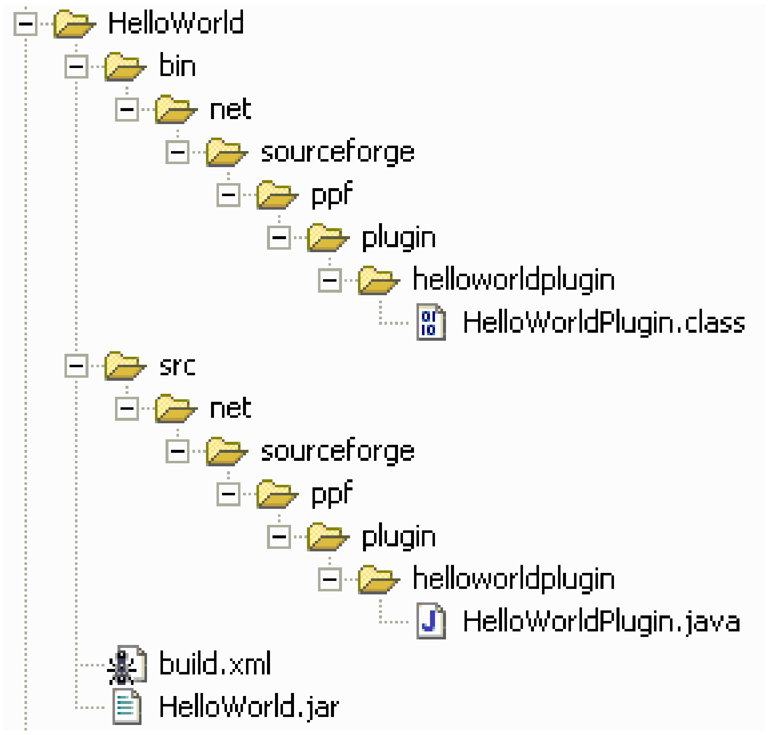
5. Create the plugin code

Now we will create the actual code for the plugin. In the *helloworldplugin* directory, create a java source file named **HelloWorldPlugin.java**. The plugin extends *PPFPlugin*. From this, we have access to all of the PircBot API methods plus some extra ones just for plugins. In the *onMessage()* method we check when someone says *hello* and then send back to the same channel the text *Hello World!*.

```
package net.sourceforge.ppf.plugin.helloworldplugin;

import net.sourceforge.ppf.PPFPlugin;

public class HelloWorldPlugin extends PPFPlugin {

    public HelloWorldPlugin() {
    }

    public void onMessage(String channel, String sender, String login,
        String hostname, String message) {

        if(message.equalsIgnoreCase("hello")) {
                getBot().sendMessage(channel, "Hello World!");
        }
    }

}
```

6. Deploy the new plugin

Now the code is written, it needs to be compiled. Run ANT with the build.xml file you created, calling the *deploy* target. This will compile the code and create a JAR file ready for loading into PPF.



Completed HelloWorld Plugin

7. Use the plugin

After a successful build, the plugin can now be loaded into PPF. Run these commands, using your own bots config details:

- **/msg BotEd auth fubar** - auth as an admin to your bot
- **/msg BotEd plugins** - check that HelloWorld is listed as a plugin that is *not* loaded
- **/msg BotEd loadplugin helloworld** - load the plugin
- **/msg BotEd plugins** - check that HelloWorld is listed as a plugin that *is* loaded

Now in a channel with the bot, type the message **hello** and the bot will respond with **Hello World!**. That is your first PPF plugin completed!

8. Change the plugin, re-deploy, and reload

Now to demonstrate the dynamic reloading, we will make a change to the plugin, rebuild it, and reload it. Let's change it a little so that it responds to the person that said *hello*. Change this line:

```
    getBot().sendMessage(channel, "Hello World!");
```

to include the sender as well:

```
    getBot().sendMessage(channel, sender +", Hello World!");
```

Now follow the step above to deploy the plugin and then type:

- **/msg BotEd reloadplugin helloworld** - reload the plugin

Now type *hello* for the bot again and you can see that the new plugin was loaded as the message now starts with the name of the person that said hello.