Create A PPF Plugin - TODO list

1. Description

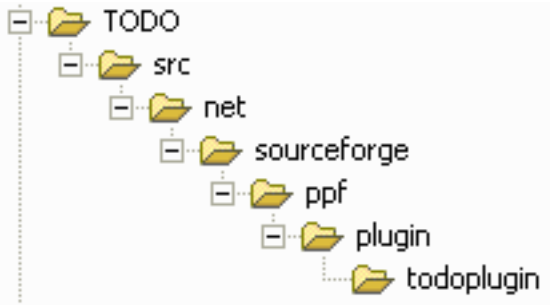
We will create a plugin that will act as a TODO list. Let's call this plugin *TODO*. From this exercise, you will bulid up a plugin that:

- has an XML configuration, allowing configuration of the commands, who can use them and where they output to
- provides version information about itself
- has online, dynamic, help information
- is able to save and retrieve a list of information using XML
- support different languages

The plugin will allow a TODO to be added with the command **!todo**. Remove a TODO with the command **!donetodo**. Show all TODOs with the command **!showtodos**. The TODO list will be stored in an XML file using methods provided by the plugin API.

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 **TODO**. We will also need a place to keep the source file(s) as well, so in the newly created TODO directory, create the following path: **src/net/sourceforge/ppf/plugin/todoplugin**



TODO 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 TODO directory. The basic ANT script for the new plugin:

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

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

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

    <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="${build.system.dir}/plugins/${plugin}">
        <fileset dir="plugins/${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>TODO</name>
    <classname>net.sourceforge.ppf.plugin.todoplugin.TODOPlugin</classname>
  </plugin>
```

5. Create the plugin code

Let's start by creating a config file. We are going to use an XML file for the plugin configuration. We have some idea of the commands we want the plugin to respond to and we can decide for now that only a user authed as admin can manage the TODO list. In the TODO plugin directory create a file named **TODOConfig.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<todoConfig>
  <!-- authLevel: ANY, ADMIN, MASTER, TRUSTED, NONE -->
  <!-- output: CHANNEL, PM, NOTICE -->
  <commands>
    <commandAddTodo authLevel="admin" output="notice">!todo</commandAddTodo>
    <commandRemoveTodo authLevel="admin" output="notice">!donetodo</commandRemoveTodo>
    <commandShowTodos authLevel="admin">!showtodos</commandShowTodos>
  </commands>
</todoConfig>
```

When that's done, also copy the file to **TODOConfig_template.xml**. This is the file that will go into the CVS system and also be used when making a release, leaving your own, personal, config out of the picture.

Now we will start to create the actual code for the plugin. In the *todoplugin* source directory, create a java source file named **TODOPlugin.java**. The plugin extends *PPFPlugin*. From this, we have access to all of the PircBot API methods plus some extra ones just for plugins. We will start with the basic plugin that has version information, help information and reads the config in. This makes the version information available when viewing the loaded plugins. It helps to identify what version you have running, which can then be matched up with the correct documentation and functionality. Reading the config is made easier by methods supplied by the plugin API. Notice the methods **loadDoc()** for reading the config in easily, and then other useful methods for pulling a piece of information from the config file, and even in different forms where needed. The help

provides online help when the bot is running, if the plugin is loaded. The help it shows is related to the auth level of the user asking for help, so in this case, anyone not authed as admin will not be provided with help for the TODO plugin as we are currently configuring it.

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

import java.util.Iterator;
import java.util.List;

import net.sourceforge.ppf.PPFPlugin;
import net.sourceforge.ppf.util.PPFHelp;

import org.jdom.Element;

public class TODOPlugin extends PPFPlugin {

    private static final String PLUGIN_VERSION = "1.0";
    private static final String PLUGIN_ID = "TODO";

    private String commandAddTodo = "";
    private int authLevelAddTodo = 0;
    private int outputAddTodo = 0;
    private String commandRemoveTodo = "";
    private int authLevelRemoveTodo = 0;
    private int outputRemoveTodo = 0;
    private String commandShowTodos = "";
    private int authLevelShowTodos = 0;

    public TODOPlugin() {
        // load the config file as a JDOM document
        loadDoc("TODOConfig.xml");

        // add todo
        commandAddTodo = getTextFromDoc("/todoConfig/commands/commandAddTodo");
        authLevelAddTodo =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandAddTodo", "authLevel"));
        outputAddTodo = outputFromString(getAttributeFromDoc("/todoConfig/commands/commandAddTodo",
"output"));

        // remove todo
        commandRemoveTodo = getTextFromDoc("/todoConfig/commands/commandRemoveTodo");
        authLevelRemoveTodo =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandRemoveTodo", "authLevel"));
        outputRemoveTodo =
outputFromString(getAttributeFromDoc("/todoConfig/commands/commandRemoveTodo", "output"));

        // show all todos
        commandShowTodos = getTextFromDoc("/todoConfig/commands/commandShowTodos");
        authLevelShowTodos =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandShowTodos", "authLevel"));

        // get the existing list of TODOs (of which there won't be any at this point as we are
        // just starting.  This reads in the list of todos.
        List todoList = getListFromDoc("/todoConfig/todos");
        if(!(todoList.isEmpty())) {
                Element allTodos = (Element) todoList.get(0);
                Iterator iter = allTodos.getChildren("todo").iterator();
                while(iter.hasNext()) {
                    Element todo = (Element) iter.next();
                    addToItemList(todo.getAttributeValue("info"),
todo.getAttributeValue("description"));
                }
        }

        // plugin version information
        setVersion(PLUGIN_VERSION);
```

```
        // online plugin help
        setHelp(commandAddTodo, new PPFHelp(commandAddTodo,
            "Add a new item to the TODO list", authLevelAddTodo));
        setHelp(commandRemoveTodo, new PPFHelp(commandRemoveTodo,
            "Remove an item frmo the TODO list", authLevelRemoveTodo));
        setHelp(commandShowTodos, new PPFHelp(commandShowTodos,
            "Show all of the items on the TODO list", authLevelShowTodos));
    }

}
```

Now we can add in some handling for those managing a persistent list of TODOs. The *!todo* and *!donetodo* are dealt with in the channel, so we override the onMessage() method from PircBot. We check for the configured command, and then check that the user has the correct (configured) auth level to use it. After that, the command is stripped away from the message and the message is parsed into the way we want to handle it. Notice the very easy list management calls **addToItemList()**, **isInItemList()**, **removeFromItemList()**, and **viewItemList()**. The *!showtodos* is dealt with as a private message so we override the PircBot method onPrivateMessage().

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

        // convert both message and command to uppercase for a startsWith comparison
        if(message.toUpperCase().startsWith(commandAddTodo.toUpperCase() + " ")) {
            // is the user allowed to perform this command?
            if(getBot().isAuthed(sender, authLevelAddTodo)) {
                // get the text after the command
                String realMsg = message.substring(commandAddTodo.length()).trim();
                // split the parts needed from the message
                String info = realMsg.substring(0, realMsg.indexOf(" ")).trim();
                String description = realMsg.substring(info.length()).trim();

                if(isInItemList(info)) {
                    sendOutput(outputAddTodo, channel, sender,
                        "\""+ info +"\" TODO already exists.  You must remove "+
                        "it first: "+ commandRemoveTodo +" "+ info);
                } else {
                    Element newTodo = new Element("todo");
                                newTodo.setAttribute("info", info);
                                newTodo.setAttribute("description", description);

                    if(addToItemList("todos", newTodo, info, description)) {
                        sendOutput(outputAddTodo, channel, sender,
                            "Added TODO \""+ info +"\" \""+ description +"\"");
                    }
                }
            }
        } else if(message.toUpperCase().startsWith(commandRemoveTodo.toUpperCase() + " ")) {
            if(getBot().isAuthed(sender, authLevelRemoveTodo)) {
                String realMsg = message.substring(commandRemoveTodo.length()).trim();

                if(removeFromItemList("todos", "//todo[@info='"+ realMsg +"']", realMsg)) {
                    sendOutput(outputRemoveTodo, channel, sender,
                        "Removed TODO \""+ realMsg +"\"");
                }
            }
        }
    }

    protected void onPrivateMessage(String sender, String login, String hostname,
        String message) {

        if(message.trim().equalsIgnoreCase(commandShowTodos)) {
            if(getBot().isAuthed(sender, authLevelShowTodos)) {
                viewItemList(sender);
            }
        }
```

```
    }
```

Now that is done, we could use the plugin already, but let's add some multilingual capabilities. Each user using the bot can set their preferred language from the languages that are available to the core. If desired, the plugin can also support these same languages, or just some of them. What we need to do is to strip out any display texts from the code and put it into property files. Let's make English, German, and Finnish property files for our plugin. The country code representation uses the codes defined at http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt. In the TODO plugin directory, create the three property files we need with any plain text editor:

todo_en.properties

```
exists = TODO already exists.  You must remove it first:
added = Added TODO
removed = Removed TODO
```

todo_de.properties

```
exists = Eintrag existiert bereits
added = Eintrag hinzugefügt
removed = Eintrag gelöscht
```

todo_fi.properties

```
exists = Kohta jo TODO-listassa. Sitä pitää ensin poistaa:
added = Kohta lisätty
removed = Kohta poistettu
```

Now we will load the properties files in the code and use the values from these, while providing default text in the code. This is made simple by calling the **readProperties()** method, passing in the root name of the properties files. Add this to the *TODOPlugin* constructor:

```
    readProperties("todo");
```

Now we can change the static texts in the code to use the values taken from these property files. Let's start with the help texts. Change:

```
  "\""+ info +"\" TODO already exists.  You must remove "+
  "it first: "+ commandRemoveTodo +" "+ info);
...
  "Added TODO \""+ info +"\" \""+ description +"\"");
...
  "Removed TODO \""+ realMsg +"\"");
```

to:

```
  "\""+ info +"\" "+ getPPFResourceBundle().getString(getBot().getUserLang(sender),
  "exists", "TODO already exists.  You must remove it first:")
  + " "+ commandRemoveTodo +" "+ info);
...
  getPPFResourceBundle().getString(getBot().getUserLang(sender),
  "added", "Added TODO") +"\""+ info +"\" \""+ description +"\"");
...
  getPPFResourceBundle().getString(getBot().getUserLang(sender),
  "removed", "Removed TODO") +" \""+ realMsg +"\"");
```

The final plugin code is:

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

import java.util.Iterator;
import java.util.List;

import net.sourceforge.ppf.PPFPlugin;
import net.sourceforge.ppf.util.PPFHelp;

import org.jdom.Element;

public class TODOPlugin extends PPFPlugin {

    private static final String PLUGIN_VERSION = "1.0";
    private static final String PLUGIN_ID = "TODO";

    private String commandAddTodo = "";
    private int authLevelAddTodo = 0;
    private int outputAddTodo = 0;
    private String commandRemoveTodo = "";
    private int authLevelRemoveTodo = 0;
    private int outputRemoveTodo = 0;
    private String commandShowTodos = "";
    private int authLevelShowTodos = 0;

    public TODOPlugin() {
        // load the config file as a JDOM document
        loadDoc("TODOConfig.xml");

        // read language property files
        readProperties("todo");

        // add todo
        commandAddTodo = getTextFromDoc("/todoConfig/commands/commandAddTodo");
        authLevelAddTodo =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandAddTodo", "authLevel"));
        outputAddTodo = outputFromString(getAttributeFromDoc("/todoConfig/commands/commandAddTodo",
"output"));

        // remove todo
        commandRemoveTodo = getTextFromDoc("/todoConfig/commands/commandRemoveTodo");
        authLevelRemoveTodo =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandRemoveTodo", "authLevel"));
        outputRemoveTodo =
outputFromString(getAttributeFromDoc("/todoConfig/commands/commandRemoveTodo", "output"));

        // show all todos
        commandShowTodos = getTextFromDoc("/todoConfig/commands/commandShowTodos");
        authLevelShowTodos =
authLevelFromString(getAttributeFromDoc("/todoConfig/commands/commandShowTodos", "authLevel"));

        // get the existing list of TODOs
        List todoList = getListFromDoc("/todoConfig/todos");
        if(!(todoList.isEmpty())) {
                Element allTodos = (Element) todoList.get(0);
                Iterator iter = allTodos.getChildren("todo").iterator();
                while(iter.hasNext()) {
                    Element todo = (Element) iter.next();
                    addToItemList(todo.getAttributeValue("info"),
todo.getAttributeValue("description"));
                }
        }

        // plugin version information
        setVersion(PLUGIN_VERSION);

        // online plugin help
        setHelp(commandAddTodo, new PPFHelp(commandAddTodo,
            "Add a new item to the TODO list", authLevelAddTodo));
        setHelp(commandRemoveTodo, new PPFHelp(commandRemoveTodo,
```

```
            "Remove an item frmo the TODO list", authLevelRemoveTodo));
        setHelp(commandShowTodos, new PPFHelp(commandShowTodos,
            "Show all of the items on the TODO list", authLevelShowTodos));
    }

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

        // convert both message and command to uppercase for a startsWith comparison
        if(message.toUpperCase().startsWith(commandAddTodo.toUpperCase() + " ")) {
            // is the user allowed to perform this command?
            if(getBot().isAuthed(sender, authLevelAddTodo)) {
                // get the text after the command
                String realMsg = message.substring(commandAddTodo.length()).trim();
                // split the parts needed from the message
                String info = realMsg.substring(0, realMsg.indexOf(" ")).trim();
                String description = realMsg.substring(info.length()).trim();

                if(isInItemList(info)) {
                    sendOutput(outputAddTodo, channel, sender,
                        "\""+ info +"\" "+
getPPFResourceBundle().getString(getBot().getUserLang(sender),
                        "exists", "TODO already exists.  You must remove it first:")
                        +" "+ commandRemoveTodo +" "+ info);
                } else {
                    Element newTodo = new Element("todo");
                                newTodo.setAttribute("info", info);
                                newTodo.setAttribute("description", description);

                    if(addToItemList("todos", newTodo, info, description)) {
                        sendOutput(outputAddTodo, channel, sender,
                            getPPFResourceBundle().getString(getBot().getUserLang(sender),
                            "added", "Added TODO") +"\""+ info +"\" \""+ description +"\"");
                    }
                }
            }
        } else if(message.toUpperCase().startsWith(commandRemoveTodo.toUpperCase() + " ")) {
            if(getBot().isAuthed(sender, authLevelRemoveTodo)) {
                String realMsg = message.substring(commandRemoveTodo.length()).trim();

                if(removeFromItemList("todos", "//todo[@info='"+ realMsg +"']", realMsg)) {
                    sendOutput(outputRemoveTodo, channel, sender,
                        getPPFResourceBundle().getString(getBot().getUserLang(sender),
                        "removed", "Removed TODO") +" \""+ realMsg +"\"");
                }
            }
        }
    }

    protected void onPrivateMessage(String sender, String login, String hostname,
        String message) {

        if(message.trim().equalsIgnoreCase(commandShowTodos)) {
            if(getBot().isAuthed(sender, authLevelShowTodos)) {
                viewItemList(sender);
            }
        }
    }

}
```

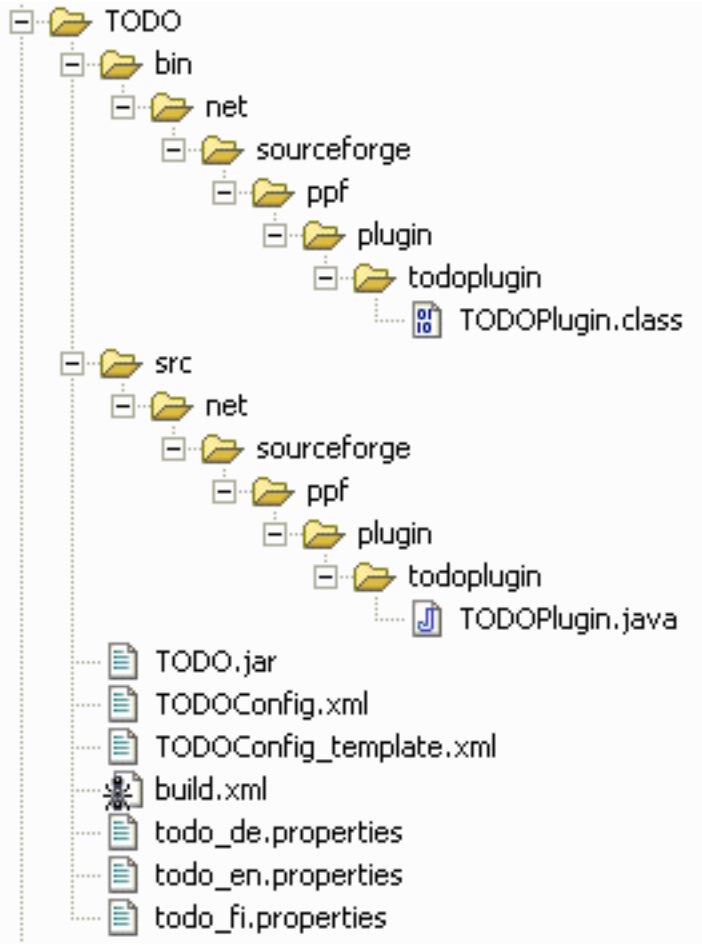
Now, due to the addition of a config file and the properties file, the properties files have to be copied and the template config has to be renamed when a full release is made, so the build.xml file changes to:

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

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

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

    <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="${build.system.dir}/plugins/${plugin}">
        <fileset dir="plugins/${plugin}">
          <include name="*.jar"/>
          <include name="*.properties"/>
        </fileset>
      </copy>
      <!-- Copy config from template to used location and name -->
      <copy file="../${plugin}/${plugin}Config_template.xml"
        tofile="${build.system.dir}/${plugin}/${plugin}Config.xml"/>
    </target>

</project>
```

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

Now in a channel with the bot, type the message:

- !todo shopping marg, bread, cheese
- /msg BotNick !showtodos
- !donetodo shopping