Auteurs : Christophe ROGER & Michèle BARRE
Version : 1.0

IMPLÉMENTATION D'UN PLUGIN POUR JIVE WILDFIRE 2.5

1. Utilisation du script de Wildfire

1.1 Principe

Le script de Wildfire doit vous aider à développer vos plugins. Pour qu'on puisse l'utiliser on doit respecter l'arborescence suivante :

```
myplugin/
 |- plugin.xml      <- Plugin definition file
 |- readme.html     <- Optional readme file for plugin
 |- changelog.html  <- Optional changelog file for plugin
 |- icon_small.gif  <- Optional small (16x16) icon associated with the plugin (can also be a .png file)
 |- icon_large.gif  <- Optional large (32x32) icon associated with the plugin (can also be a .png file)
 |- classes/        <- Resources your plugin needs (i.e., a properties file)
 |- lib/            <- Libraries your plugin needs
 |- src/
     |- java        <- Java source code for your plugin
     |   |- com
     |       |- mycompany
     |           |- *.java
     |- web
         |- *.jsp      <- JSPs your plugin uses for the admin console
         |- images/    <- Any images your JSP pages need (optional)
         |- WEB-INF
             |- web.xml    <- Optional file where custom servlets can be registered
```

Le script compile les fichiers sources et JSP et crée une structure de plugin valide ainsi qu'un fichier JAR. Mettez votre répertoire de plugin dans le dossier src/plugins des sources de Wildfire et utilisez « **ant plugins** » pour fabriquer vos plugins.

Si le plugin a besoin de JAR suplémentaires, ils doivent être placés dans le répertoire lib. Ces JAR seront également copiés dans le répertoire lib du plugin généré.
.
Si vous créez un fichier src/web/WEB-INF/web.xml, chaque servlet figurant dans ce fichier sera initialisée au lancement du plugin.

1.2 Description des fichiers à créer

- plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
  <!-- Main plugin class -->
  <class>org.example.ExamplePlugin</class>

  <!-- Plugin meta-data -->
  <name>Example Plugin</name>
  <description>This is an example plugin.</description>
```

```
  <author>Jive Software</author>
  <version>1.0</version>
  <date>07/15/2005</date>
  <url>http://www.jivesoftware.org/pluginX</url>
  <minServerVersion>2.2.0</minServerVersion>

  <!-- Admin console entries : optional -->
  <adminconsole>
    <!-- More on this below -->
  </adminconsole>
</plugin>
```

Les champs de meta-data qui peuvent être utilisés dans le fichier plugin.xml sont :

name – le nom du plugin.
description -- la description du plugin.
author -- l'auteur du plugin.
version -- la version du plugin.
date – la date de création du plugin. Elle doit être de la forme mm/jj/aaaa, exemple 07/15/2005.
url -- une URL où des informations supplémentaires sur le plugin sont disponibles.
minServerVersion – la version minimum de Wildfire requise pour exécuter le plugin. Si la version du serveur est inferieur à celle requise, le plugin ne sera pas démarré.
parentPlugin – le nom du plugin parent (mettre "foo" pour le plugin "foo.jar"). Quand un plugin a un parent, c'est le chargeur de classe du parent qui sera utilisé au lieu de créer un nouveau chargeur de classe. Cela permet aux plugins de mieux fonctionner ensemble. Un plugin fils ne fonctionnera pas sans la présence de son père.

Quelques fichiers supplémentaires peuvent être présents dans le plugin pour apporter des informations additionnelles pour les utilisateurs finaux (tous sont à placer à la racine du plugin)

readme.html – Optionel, qui sera affiché pour les utilisateurs finaux.
changelog.html -- Optionel, qui sera affiché pour les utilisateurs finaux.
icon_small.png – Optionel, petite icone (16x16) associée au plugin. Peut aussi être un fichier .gif.
icon_large.png – Optionel, grande icone (32x32) associée au plugin. Peut aussi être un fichier .gif.

2. Implémentation du plugin

La classe de votre plugin doit implémenter l'interface Plugin de l'API de Wildfire et avoir un contructeur par défaut (sans arguments). L'interface Plugin contient des méthodes pour initialiser et détruire le plugin.

```
package org.example;

import org.jivesoftware.wildfire.container.Plugin;
import org.jivesoftware.wildfire.container.PluginManager;

import java.io.File;

/**
 * A sample plugin for Wildfire.
 */
public class ExamplePlugin implements Plugin {

    public void initializePlugin(PluginManager manager, File pluginDirectory) {
        // Your code goes here
```

```
    }

    public void destroyPlugin() {
        // Your code goes here
    }
}
```

Les plugins ont un accès complet à l'API de Wildfire. Ce qui donne aux plugins plus de flexibilité dans ce qu'ils peuvent accomplir. Il existe différentes manières d'implémenter les plugins :

- L'intégration en tant que composant : Les composants reçoivent tous les paquets adressés à un sous-domaine particulier. Donc un paquet adresse à joe@composant.exemple.com sera délivré au composant « composant ». La notion des sous-domaines abordé ici n'a rien à voir avec les sous domaines définis sur le DNS. Le routage des paquets au niveau des sockets s'effectue en le domaine du serveur (ici exemple.com), les sous domaines ne sont utilisés que pour le routage des paquets à l'intérieur du serveur XMPP.

- L'intégration en tant que *IQHandler*. *IQ handlers* répondent aux paquets *IQ* qui ont un nom et un espace de nom particulier. Voilà comment enregistre un *IQHandler*

```
IQHandler myHandler = new MyIQHander();
IQRouter iqRouter = XMPPServer.getInstance().getIQRouter();
iqRouter.addHandler(myHandler);
```

- L'intégration en tant que *PacketInterceptor* pour recevoir tous les paquets qui sont envoyés à travers le système, et éventuellement les rejeter.

3. Modification de la Console d'Administration

Les plugins peuvent ajouter des onglets, des sections et des pages à la console d'administration. Il y a quelques manipulations à faire pour obtenir cela:

Une section <adminconsole/> doit être ajoutée au fichier plugin.xml.
Des fichiers JSP doivent être compilés et placés dans le classpath du plugin. Un fichier web.xml contenant les entrées des servlets des JSP compilés doit être placé dans le répartoire web/ du plugin.
Note: le script de Wildfire peut compiler les JSP et créer le fichier web.xml. Celui-ci est détaillé plus bas
Les images dont les JSP ont besoin doivent être dans le dossier web/images/. Seul les images GIF et PNG sont supportées.

La section <adminconsole/> de plugin.xml définie les onglets, les sections et entrée ajoutée dans la console d'administration.
Un fichier plugin.xml file devrait ressembler à cela :

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
    <!-- Main plugin class -->
    <class>org.example.ExamplePlugin</class>

    <!-- Admin console entries -->
    <adminconsole>
        <tab id="mytab" name="Example" url="my-plugin-admin.jsp" description="Click
      to manage...">
            <sidebar id="mysidebar" name="My Plugin">
```

```
                <item id="my-plugin" name="My Plugin Admin"
                    url="my-plugin-admin.jsp"
                    description="Click to administer settings for my plugin" />
            </sidebar>
        </tab>
    </adminconsole>
</plugin>
```

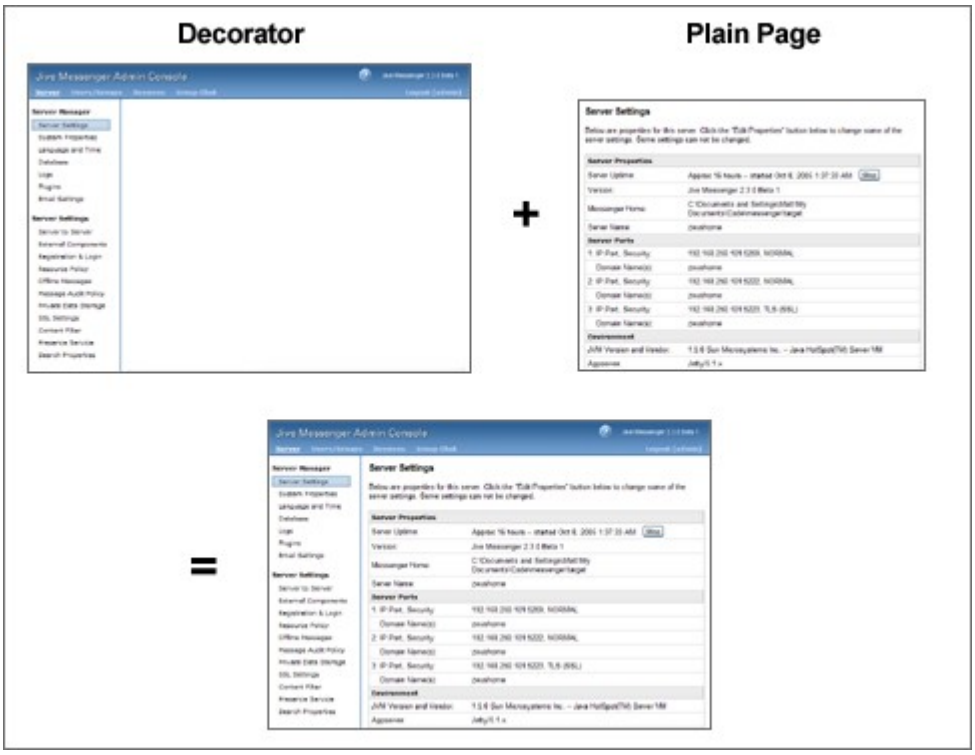
Dans cet exemple, nous avons défini un nouvel onglet « Example », une section « My Plugin » dans la barre latérale et une page « My plugin Admin ». Nous avons enregistré my-plugin-admin.jsp comme page principale. Vous pouvez redéfinir les onglets, les sections et items existants en utilisant les valeurs d'attributs exitant dans votre propre définition d'e <adminconsole>.

3.1 Recommandation pour la console d'administration

Il y a quelques bonnes pratiques à considérer lorsque vous faites des changement sur la console d'administration de Wildfire via un plugin. L'idée générale est que les plugins doivent :

- Intégrer dans des onglets, section de barre latérale existant autant que possible plutot que de créer les votre. Ne créer de nouveau onglet que pour des fonctionnalités très significatives.
- Ne pas utiliser le mot « plugin » dans les noms d'onglets et d'items. Par exemple, au lieu d'appeler un item « Gateway Plugin », l'appeler « Gateway Setting ».
- Essayer de respecter l'UI de la console d'administration dans les pages de votre plugin.
- Vous n'avez pas besoin de créer une entrée sur la console d'administration pour montrer les meta-data de votre plugin. Laissez Wildfire informer l'utilisateur des plugins installés et permettre leur gestion.

Wildfire utilise le Sitemesh framework pour décorer les pages de la console d'administration. Un décorateur s'applique à chaque page pour obtenir l'aspect suivant :



A compléter après première création de page d'administration