

TP : OSGi

Ada Diaconescu

contact : [ada.diaconescu + @ + telecom-paristech.fr](mailto:ada.diaconescu@telecom-paristech.fr)

Objectif

L'objectif de ce TP consiste à se familiariser à la technologie OSGi (www.osgi.org – « The Dynamic Module System for Java »).

Nous utiliserons [Apache Felix](http://apache.felixproject.org) comme plate-forme d'implantation de la spécification OSGi.

Note : les connaissances d'OSGi seront nécessaires par la suite pour suivre la formation introductive sur la technologie à composant orienté service - iPOJO (www.ipoyo.org).

Cas d'Etude

Nous allons voir comment implanter et gérer une application HelloWorld en OSGi.

Dans cette application, un `Client` obtient des messages auprès d'un service de type `Hello`. Le `Client` affiche les messages via une interface graphique en utilisant un service `ClientGUI`. Plusieurs services de type `Hello` seront disponibles, chacun retournant des messages dans une langue différente. Chaque service `Hello` spécifie la langue de ses messages via une propriété « Language » (ex : « Language=French » ou « Language=English »).

Figure 1 montre le diagramme de classes Java « classique » de cette application.

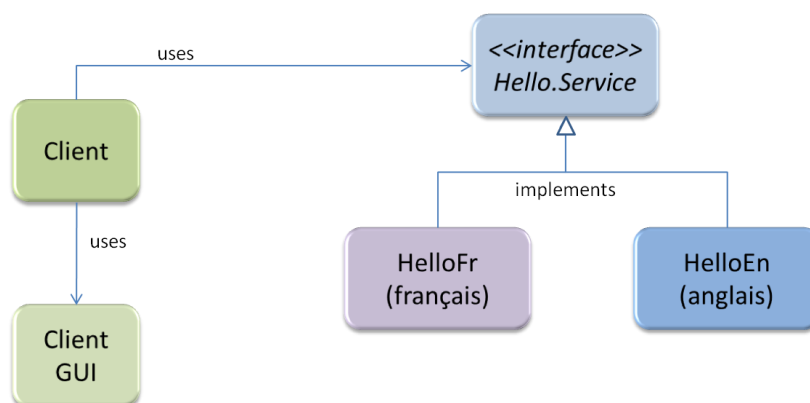


Figure 1: Plain Java class diagram for the HelloWorld example

La Figure 2 indique la façon dont l'application HelloWorld est découpée et organisée en modules OSGi (Bundles) interdépendants.

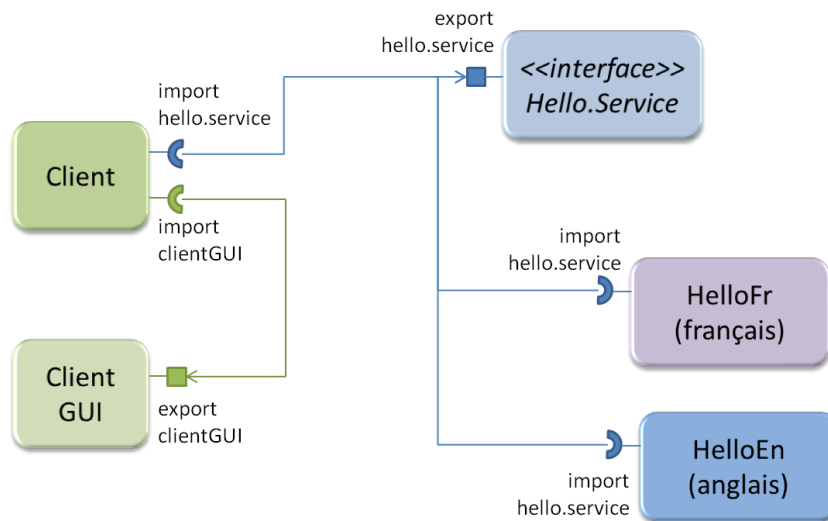


Figure 2: OSGi module (Bundle) view for the HelloWorld example

Enfin, Figure 3 indique la façon dont les services OSGi seront connectés à l'exécution.

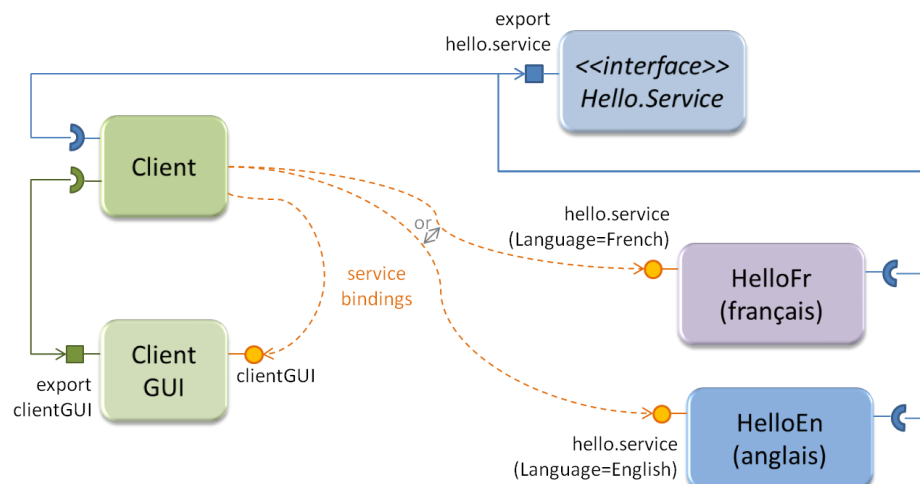


Figure 3: OSGi Service view for the HelloWorld example

Travaux Pratiques

Installation et exécution de Felix

- Télécharger la dernière distribution de Felix à partir de la page de [téléchargement](#) du projet (ex : Felix Framework Distribution 2.0.4 – archives felix-framework-2.0.4 .tar.gz ou zip). Autrement, la version 2.0.4 est également disponible [ici](#).
- Désarchiver la distribution
 - o `unzip <distribution_felix>.zip`, ou
 - o `tar -xzf <distribution_felix>.tar.gz`
- Pour la simplicité, renommer le répertoire de la distribution Felix : « felix »
 - o `mv <distribution_felix> felix`
 - o ex : `mv felix-framework-2.0.4 felix`
- Pour démarrer Felix : à partir du répertoire d'installation (<felix_home>) - ex : <répertoire_du_TP>/felix :
 - o `java -jar bin/felix.jar`
- Pour voir les Bundles disponibles et leurs états, depuis la console de commande de Felix :
 - o `ps`
- Afin d'obtenir la liste de commandes Felix disponibles, utiliser :
 - o `help`, ou `help <nom_commande>`
- Pour arrêter Felix, depuis sa console de commande :
 - o `shutdown`

Code du TP

- Télécharger et désarchiver le code source du TP – [distrib](#) (dans le même répertoire dans lequel vous avez téléchargé la distribution de Felix => dans le <répertoire_du_TP> vous aurez deux répertoires : *felix* et *distrib*).
- Compiler les classes java – à partir du répertoire *distrib* :
 - o `javac -d bin -classpath ../felix/bin/felix.jar src/client/*.java src/clientGUI/*.java src/hello/service/*.java src/helloEn/*.java src/helloFr/*.java`

Ou plus simple, utiliser le fichier build.xml fourni avec la distribution du TP :

- o `ant compile`

(les classes compilées seront mises dans le répertoire *distr/bin*)

Création, installation et démarrage des Bundles OSGi

- Créer les Bundles OSGi – à partir du répertoire *distrib/bin* :
 - o Pour le client : `jar cfm ../jars/helloEn.jar ../src/client/manifest.mf client`
 - o Pour le clientGUI : `jar cfm ../jars/clientGUI.jar ../src/clientGUI/manifest.mf clientGUI`
 - o Pour le hello.service : `jar cfm ../jars/hello.jar ../src/hello/service/manifest.mf hello/service`
 - o Pour le helloEn : `jar cfm ../jars/helloEn.jar ../src/helloEn/manifest.mf bin/helloEn`

- Pour le helloFr : `jar cfm ../jars/helloFr.jar ../src/helloFr/manifest.mf bin/helloFr`

Ou plus simple, utiliser le fichier build.xml fourni avec la distribution du TP :

- `ant jars`

(les Bundles – archives *.jar*, seront mis dans le répertoire *distr/jars*)

- Installer et démarrer les Bundles OSGi : démarrer Felix et depuis sa console de commande :
 - `start`
file:<chemin_vers_le_répertoire_du_TP>/distrib/jars/hello.jar
 - `start`
file:<chemin_vers_le_répertoire_du_TP>/distrib/jars/helloEn.jar
 - `start`
file:<chemin_vers_le_répertoire_du_TP>/distrib/jars/helloFr.jar
 - `start`
file:<chemin_vers_le_répertoire_du_TP>/distrib/jars/clientGUI.jar
 - `install`
file:<chemin_vers_le_répertoire_du_TP>/distrib/jars/client.jar

NOTE : Felix garde les Bundles installés dans un cache - répertoire <felix_home>/felix-cache, ce qui permet d'éviter à devoir les installer (et les démarrer) à chaque redémarrage de Felix. Par conséquent, pour toute modification d'un bundle (.jar) déjà installé, il faudrait effacer le bundle correspondant du cache de Felix.

- Vérifier les bundles disponibles et leur état (toujours dans la console de commande de Felix) :
 - `ps`

Felix affichera la liste de bundles installés, sous la forme :

```
-> ps
START LEVEL 1
ID    State      Level Name
[ 0] [Active]    [ 0] System Bundle (2.0.4)
[ 1] [Active]    [ 1] Apache Felix Bundle Repository (1.4.3)
[ 2] [Active]    [ 1] Apache Felix Shell Service (1.4.2)
[ 3] [Active]    [ 1] Apache Felix Shell TUI (1.4.1)
[ 4] [Active]    [ 1] Hello Service (1.0.0)
[ 5] [Active]    [ 1] English Hello (1.0.0)
[ 6] [Active]    [ 1] French Hello (1.0.0)
[ 7] [Active]    [ 1] Hello GUI Client (1.0.0)
[ 8] [Installed] [ 1] Dynamic Hello Client (1.0.0)
->
```

Les informations affichées pour chaque Bundle :

- ID – identifiant du Bundle (à utiliser pour les commandes start, update, stop, ...)
- State – état du Bundle (Installed, Resolved, Active, ...)
- Name – le nom du Bundle, tel que spécifié dans son manifest.mf

Exécution du cas d'utilisation

- Démarrer le bundle client :
 - `start <ID_du_bundle_client>` (ex : pour le cas de la figure ci-dessus : `start 8`)

Une fenêtre graphique affichera le message « Hello World »

- Que se passe-t-il lors de l'arrêt du service HelloEn ? (`stop <ID_du_bundle_English_Hello>` - ex : `stop 5`)
- Que se passe-t-il lors de l'arrêt du service HelloFr ? (`stop <ID_du_bundle_French_Hello>`)
- Que se passe-t-il lors du redémarrage du service HelloFr ? (`start <ID_du_bundle_French_Hello>`)
- Que se passe-t-il lors du redémarrage du service HelloEn ? (`start <ID_du_bundle_English_Hello>`)
- Comment s'explique ce comportement ?
- Pour arrêter le client il suffit d'arrêter son Bundle.

Exercices

1. Modifier le client de façon à changer sa langue préférée.
2. Modifier le client de façon qu'il puisse réutiliser le service préféré lors de son enregistrement, même si ce service s'enregistre après que client ait déjà trouvé un autre service Hello (ex : HelloFr).
3. Ajouter un service Hello<NewLanguage>, envoyant des messages dans une autre langue.