Exercices OSGi

1. Lancement de la plate-forme

Les plates-formes disponibles sont felix / oscar / knopferlisch / equinox. On travaille sur felix.

Lancer felix : java -jar lib/felix.jar

On obtient un shell de commande permettant d'interroger le framework. Testez les commandes :

ps

ps -l

headers

obr list

...

Arrêter la plate-forme se fait avec la commande shutdown

2. Premier bundle

Réaliser un bundle réalisant HelloWorld dans la console du framework. Il faut pour cela.

1) Déclarer une classe hello.HelloWorld qui sera l'activateur. (Attention une classe possède toujours un nom de package). Pour être activateur il faut qu'elle implante l'interface org.osgi.framework.BundleActivateur. Cette interface impose d'implanter deux méthodes

- `public void start(org.osgi.framework.BundleContext ctx)` et
- `public void stop(org.osgi.framework.BundleContext ctx)`

2) Compilez la classe

```
mkdir classes
javac -d classes -classpath ... src/hello/HelloWorld.java
```

2) Déclarer un fichier `Manifest` (dans `src/hello` par exemple) qui contient au minimum l'indication de la classe d'activation du bundle : `Bundle-Activator: hello.HelloWorld.` D'autres propriété possibles sont :

```
Bundle-Name: Hello World
Bundle-Description: Le Bundle hello
```

3) Fabriquez le bundle en réalisant une archive java contenant votre activateur et prenant en paramètre votre manifest.

```
jar cvfm bundle1.jar src/hello/manifest.mf classes
```

Votre archive doit avoir cette structure : `jar tvf bundle1.jar`

```
META-INF/MANIFEST.MF
hello/HelloWorld.class
```

4) Installez votre archive sur le framework felix

>start file:/XXX/bundle1.jar
Faites un stop/start et observez le cycle de vie de votre composant (bundle).

5*) Réalisez un build.xml qui automatise toutes ces tâches
6*) Modifiez votre helloworld pour lever une exception BundleException au démarrage du bundle.

3. Import / Export

Un des services offferts par OSGi est de garantir que les packages sont disponibles entre bundles.

Installer le bundle export.jar. Il exporte le package insa.export qui contient la classe Server.
Celle ci répond à la méthode statique test

> exports

Réaliser un bundle clientimp.jar qui utilise la méthode statique de l'autre bundle.

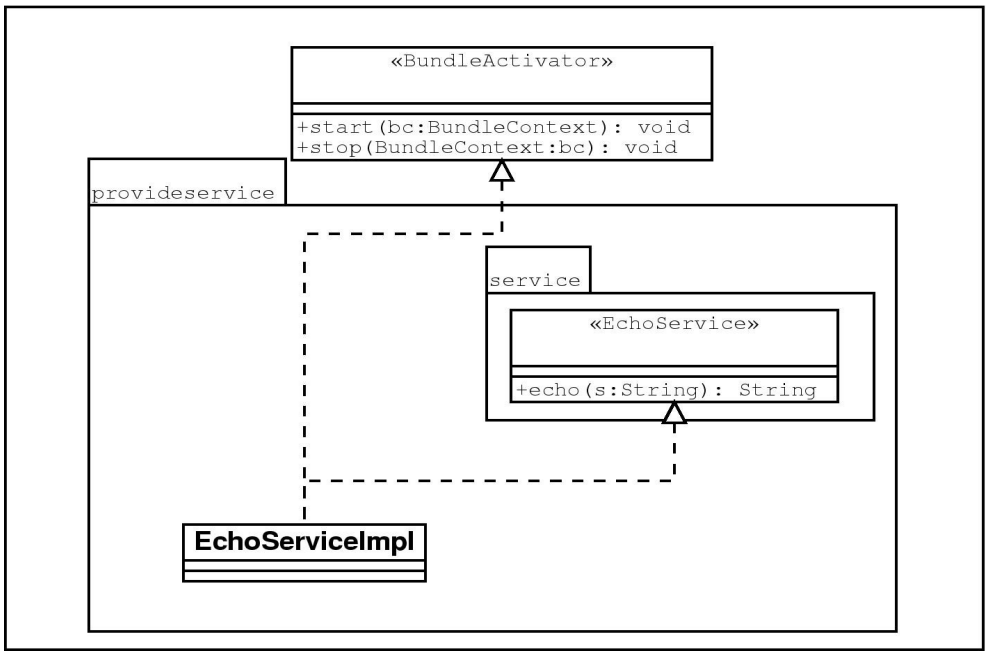
package clientimp
classe Client
Bundle-Activator: clientimp.Client
Import-Package: xxxx

==> Packagez ce bundle installez et lancez.

4. Utilisation d'un service

Vous allez installer et utiliser le service de Log.
> obr start "Log Reader"
Le service installé vous permet d'envoyer des logs à la place de faire System.out.println(...).
-> Trouvez l'interface du service,
-> Trouvez la fonction de log
-> Modifiez votre bundle Hello pour utiliser ce service

5. Fabrication d'un service Echo



Inscription du service
bc.registerService("provideservice.service.EchoService", this, null);
ou
bc.registerService("provideservice.service.EchoService, new EchoImpl(), null)
Fabriquer et mettre à disposition le service Echo sur la plateforme
Tester avec le client fournit !