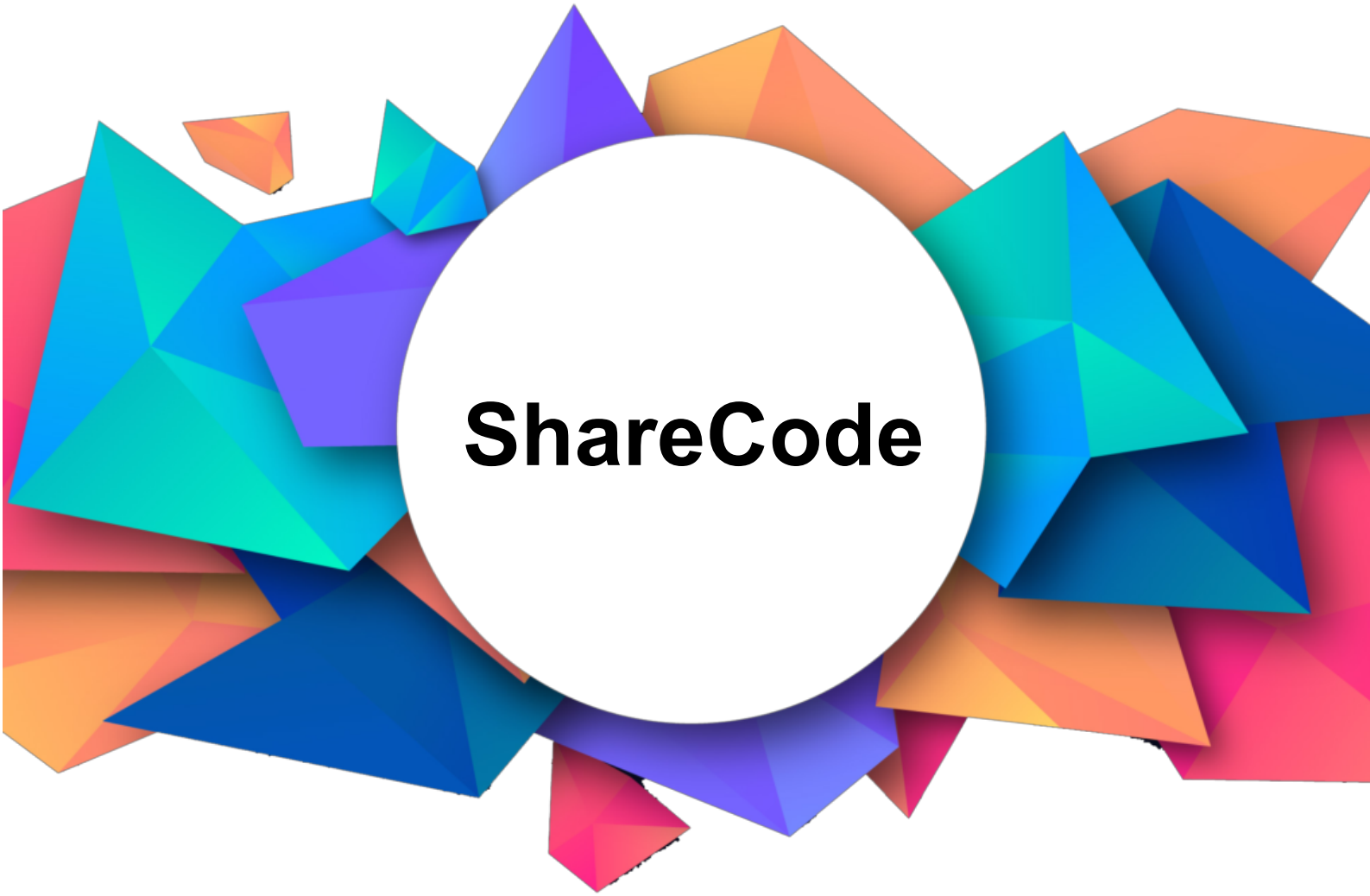
Réflexivité Java



« Une plateforme d'échange de services entre développeurs et amateurs »

Tristan Bilot - 201

Sommaire

I – Présentation

Note : le code et les commentaires du projet ont été fait en Anglais pour m'habituer à utiliser les bonnes pratiques en programmation.

ShareCode est une plateforme qui permet à des développeurs et des utilisateurs d'interagir entre eux et de se partager des services. Un développeur aura plus de privilèges qu'un amateur (utilisateur) vu qu'il pourra ajouter, supprimer, modifier, activer ou désactiver des services déjà existants sur la plateforme. Les développeurs sont donc vus comme les administrateurs de la plateforme. Les utilisateurs, quant à eux peuvent utiliser les services proposés par les développeurs, à condition que ce service soit activé.
Il est également important de noter qu'un développeur est également considérer comme un amateur car il peut également utiliser les services de la plateforme (heureusement !).

1. Un système client-serveur

Pour permettre des connexions multiples à la plateforme, un serveur est donc lancé. Ce serveur est en permanence à l'écoute de nouveaux clients sur 2 ports :

- Le port 2500 pour les amateurs
- Le port 2600 pour les développeurs

Lors d'une connexion au port 2600, un service de connexion sera invoqué et demandera au développeur d'entrer ses identifiants. Sans cela, n'importe quel utilisateur aurait pu accéder à des sessions administrateurs (à éviter).
En revanche, le port amateur est « libre d'accès », il n'y a donc pas besoin d'avoir d'identifiants pour accéder aux principaux services. Nous verrons par la suite que dans certains cas, l'amateur sera invité à se connecter pour l'utilisation de certains services.

2. Un serveur FTP

Avant tout ajout de services par le logiciel, le développeur devra déposer sa classe sur le serveur FTP associé à son compte. En effet, chaque développeur possède un attribut ftpUrl qui est le chemin vers son répertoire FTP. Au moment de l'ajout, le logiciel va aller vérifier si la classe est bien présente et va ensuite la charger dynamiquement.
Étant sous MacOS, j'ai décidé d'utiliser QuickFTP, un logiciel léger disponible sur l'AppStore. Sous Windows, un serveur FTP comme FilleZilla peut être utilisé.

3. Le lancement de la plateforme

Pour démarrer la plateforme, il suffit de compiler et exécuter « ServerLaunch.java » pour démarrer le serveur en écoute sur les 2 ports. Il suffit ensuite de lancer les clients : « ClientAmateur.java » ou « ClientProgrammer.java » pour interagir avec les services.

II – Les services

Les 2 services majeurs sont « ServiceProgrammer » et « ServiceAmateur », chacun de ces services sera respectivement déclenché lorsque son client sera exécuté.
Un service de connexion est également implanté pour la connexion au port des développeurs et pour certains services des amateurs.

1. La structure d'un service

Lors de l'ajout d'un service, la classe utilisée doit être conforme à certaines règles afin que la communication entre le service et le client se déroule correctement.
Tout d'abord, lorsqu'un développeur veut ajouter un service, il doit s'assurer que son service soit un Runnable, qu'il possède un constructeur avec un paramètre String (qui sera le code input à examiner), la classe devra faire partie du package « services » et devra posséder une méthode getResultOfService() qui retourne un String : le résultat renvoyé par le service. Dans notre cas, les services implantés n'effectueront que des tâches sur des String (des fichiers XML, JSON etc…) à examiner.
Le String retourné sera soit de type booléen converti en String (ex : « Ce fichier est bien un fichier XML ») ou bien un résultat crée par l'algorithme (ex : « Votre chaîne inversée est : ! olleH »).
Un fois la classe (service) ajouté par le développeur, un nouvel objet Service sera créé et recevra la nouvelle classe fraîchement ajoutée. L'objet Service a également un attribut « activated » qui sera utile pour l'activation/désactivation des services.

2. Gestion des services

Toutes les méthodes concernant la gestion des services se trouvent dans une classe statique « ServicesDatabase ». C'est donc dans cette classe que tous les services sont stockés et qu'ils peuvent être manipulés.

3. Ajout et chargement d'un service

L'ajout de service est une tâche strictement réservée aux développeurs.

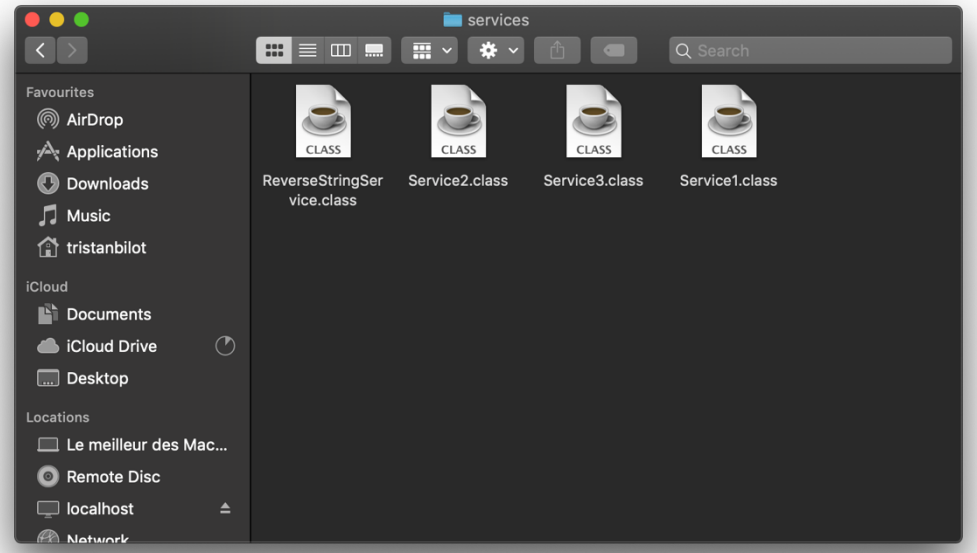
- Lancement du client pour développeurs

```
[+] Your are now connected to the server /127.0.0.1:2600
Your login:
a
Your password:
a
[+] Connection OK !

1
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
```

Le développeur est invité à se connecter, un utilisateur root a déjà été inscrit dans le projet :

- Login : a
- Password : a
- Ftp Url : ftp://localhost:4446



- *Choix du service*

```
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
1
[+] Connection to your FTP server: ftp://localhost:4446/
Please, tap the name of the class file -->
ReverseStringService
[+] Class ReverseStringService.class loaded with success !
```

Dans notre cas, nous choisissions l'ajout de service (1).
Il faut ensuite indiquer le nom de la classe à ajouter (sans le .class), dans mon cas, mes services se trouvent dans un répertoire services sur mon serveur FTP.
La classe est chargée avec succès si le nom et le chemin sont corrects. Il sera nécessaire de changer l'adresse FTP de l'utilisateur selon la machine utilisée.

A partir du moment, ou la classe est chargée avec succès, le nouveau service créé sera ajouté à l'ArrayList de services dans la classe ServicesDatabase.

4. Modification d'un service

```
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
2
[+] Connection to your FTP server: ftp://localhost:4446/
==> Please, tap the name of the class file:
ReverseStringService
[+] Class found !
--> Enter the name of the new service:
Service2
[+] Service updated correctly !
```

Pour modifier un service, le développeur doit d'abord entrer le nom du service à modifier. Attention, ce service doit avoir été préalablement ajouté en utilisant l'option 1). Il suffit ensuite d'indiquer le nom de la classe qui va remplacer l'enseigne, cette classe doit bien évidemment se situer sur le serveur FTP.

5. *Modification de l'Url du serveur FTP*

```
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
3
==> Enter your new url:
ftp://localhost:4446/
[+] Great, your FTP server is now: ftp://localhost:4446/
```

Le développeur peut modifier son url de serveur FTP, il lui suffit de rentrer sa nouvelle adresse et le changement sera immédiat.

6. *Arrêt / démarrage d'un service*

```
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
4
<==> Choose an action (1/2) <==>
(1) Start a service
(2) Stop a service
2
==> Enter the name of the service:
ReverseStringService
[+] Class found.
The service has been stopped correctly.
```

Un fois qu'un service est ajouté, il peut être dans 2 états : activé (par défaut) ou désactivé. Cet état est une alternative à la suppression complète d'un service comme nous le verrons dans la prochaine section. Après avoir choisi 1) pour démarrer ou 2) pour éteindre, vous devrez indiquer le nom du service et l'opération sera exécutée.

7. *Supprimer un service*

```
Which service do you want to use ?
--> (1) Put a new service
--> (2) Update a service
--> (3) Change your FTP server IP
--> (4) Start/stop a service
--> (5) Delete a service
5
==> Enter the name of the service to delete:
ReverseStringService
[+] Class found.
The service has been removed correctly.
```

La méthode la plus radicale est la suppression d'un service. Cette méthode aura pour effet de supprimer le service de la liste des services, il ne sera donc plus possible d'y accéder à moins de recharger la classe.

III – Différents services

Malgré un choix plus limité, les amateurs peuvent tout de même utiliser les services proposés par les programmeurs.

L'utilisateur est invité à choisir un service parmi ceux qui sont considérés comme « activés ». La classe chargée dynamiquement est ensuite récupérée et utilisée.

1. *Service d'inversion de String*

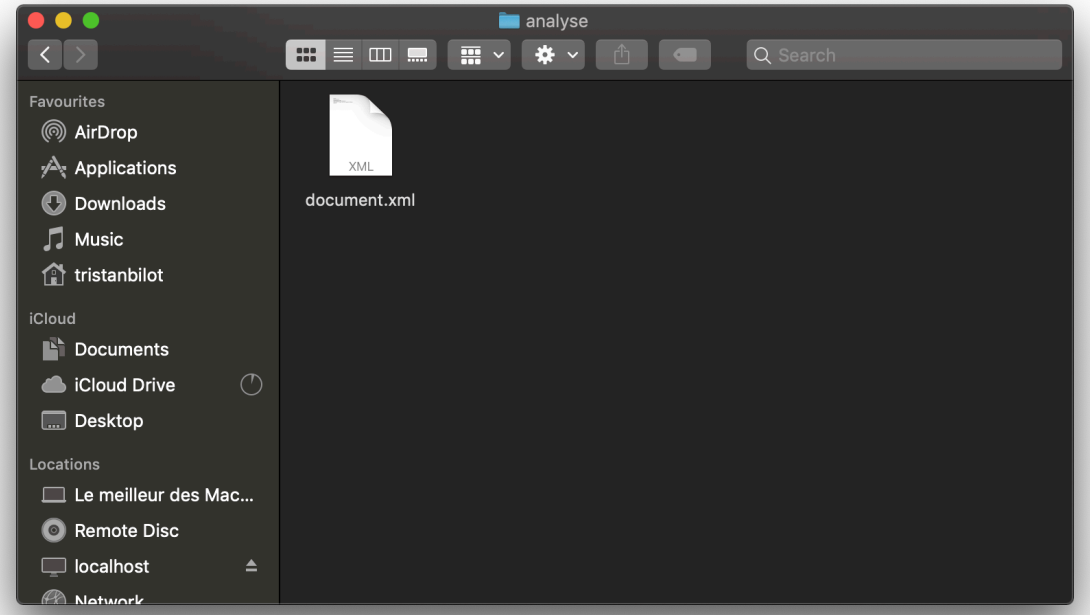
Ce premier service assez simple permet à l'utilisateur d'inverser une phrase ou un texte.

```
[+] You are now connected to the server: /127.0.0.1:2500

==> Choose a service to start:
(1) ReverseStringService
(2) CheckJsonFileService
(3) CheckXmlFileService
1
Please, enter the input code you want to analyse:
My super code
==> Here is the response of the service:
edoc repus yM
```

2. *Service de reconnaissance de fichier XML*

Ce service permet d'analyser un fichier XML posé sur le serveur FTP et d'en déduire s'il s'agit bien de la syntaxe d'un fichier XML ou non.



/ Fichier document.xml */*
<note>
<to>Hey</to>
<from>From </from>
<heading>Paris</heading>
<body></body>
</note>

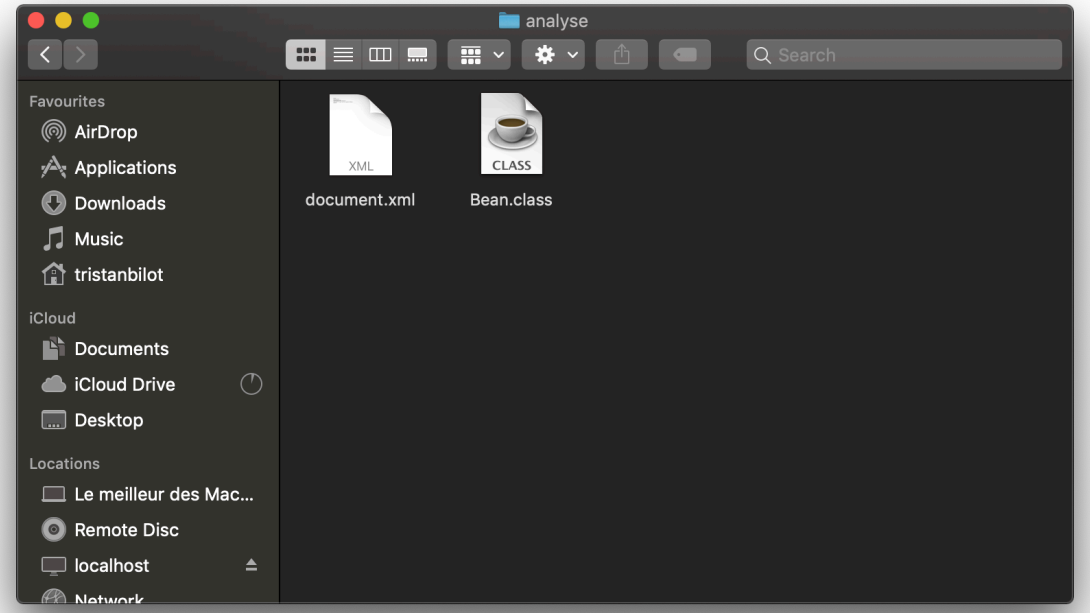
```
[+] You are now connected to the server: /127.0.0.1:2500

==> Choose a service to start:
(1) ReverseStringService
(2) CheckJsonFileService
(3) CheckXmlFileService
3
Please, enter the input code you want to analyse:
document
==> Here is the response of the service:
This document is an XML file.
```

Dans ce cas, il faut indiquer le nom du fichier XML (sans l'extension) et non le code.

3. *Service de reconnaissance de fichier JSON*
4. *Service de reconnaissance de classe JavaBean*

Ce service détermine si une classe est un JavaBean ou non. Grâce aux méthodes de réflexivité, il est assez simple de charger la classe à partir de son nom et d'examiner chaque méthode, attribut ou constructeur afin de déterminer s'il s'agit d'un Bean.




```
==> Choose a service to start:
(1) ReverseStringService
(2) CheckJsonFileService
(3) CheckXmlFileService
(4) CheckBeanClassService
4
Please, enter the input code you want to analyse:
Bean
==> Here is the response of the service:
[+] This class is a bean.
```

III – Amélioration possible

La plateforme pourrait accepter des services de toutes catégories et non seulement des services appliqués à des Strings.