ExempleCh08_02

Class Loader

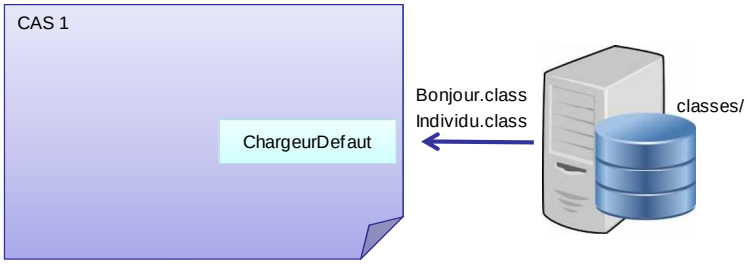
Chargement des classes dynamiquement en JAVA

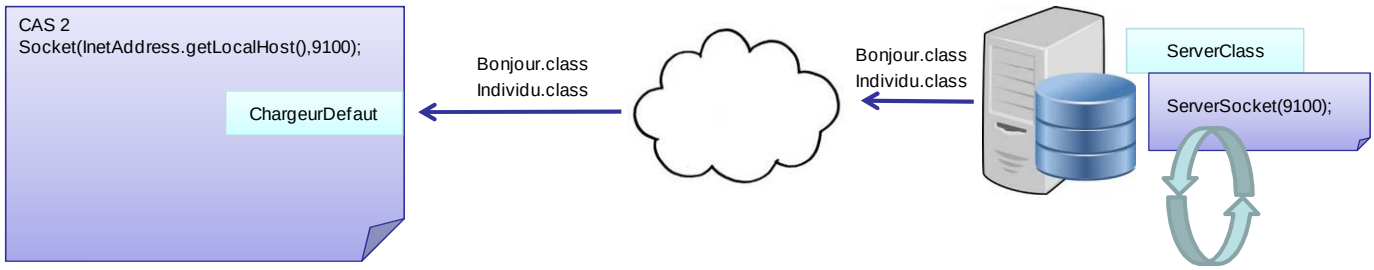
1. Description

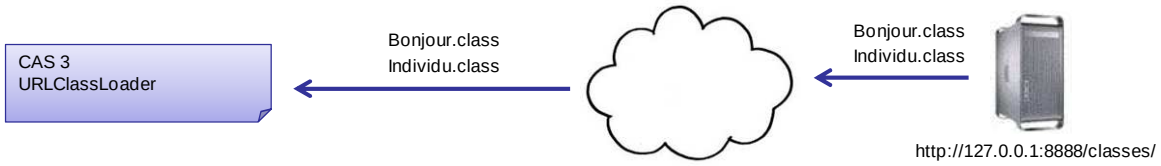
L'objectif de cet exemple est de montrer la mise en œuvre en Java pour le chargement des classes.

Les cas réalisés sont :

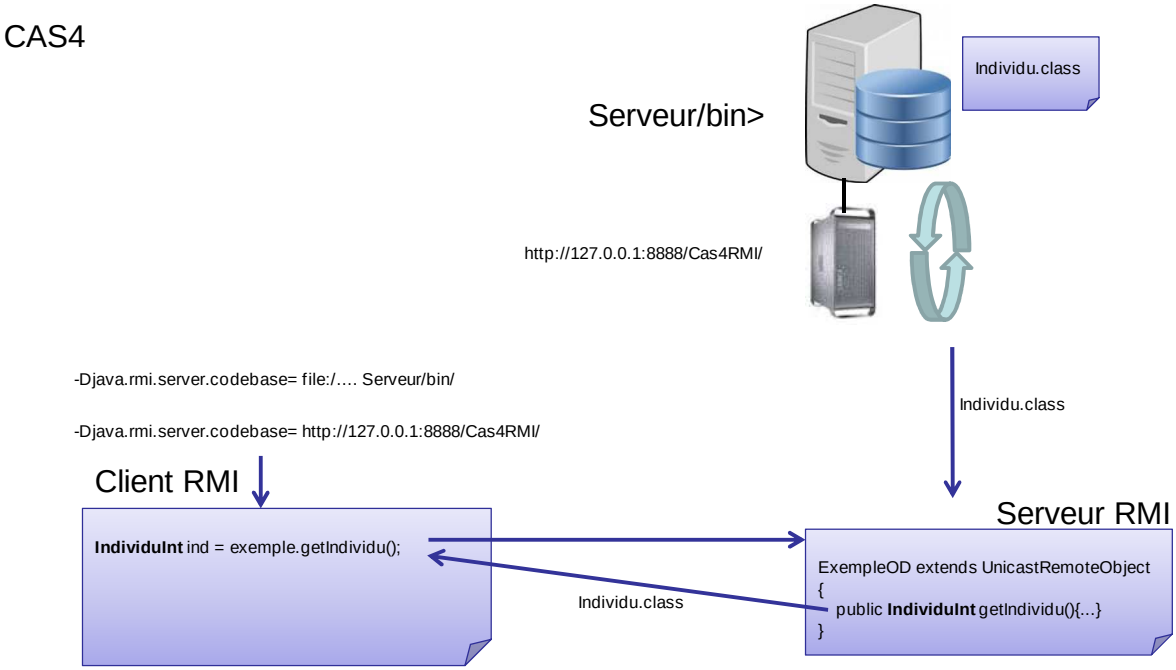
- cas 1 : : le chargement des classes qui se trouvent dans un répertoire
- cas 2 : le chargement des classes qui se trouvent dans une autre JVM à travers un serveur de socket
- cas 3 : le chargement d'une classe qui se trouvent sur un serveur http
- cas 4 : le chargement de classe à travers le protocole RMI. Toutes les classes dynamiques sont sur le serveur
- cas 5 : le chargement de classe à travers le protocole RMI. Les classes dynamiques sont sur le client et sur le serveur
- cas 6 : le chargement de classe sans passer par le protocole RMI. . Les classes dynamiques sont sur le client et sur le serveur

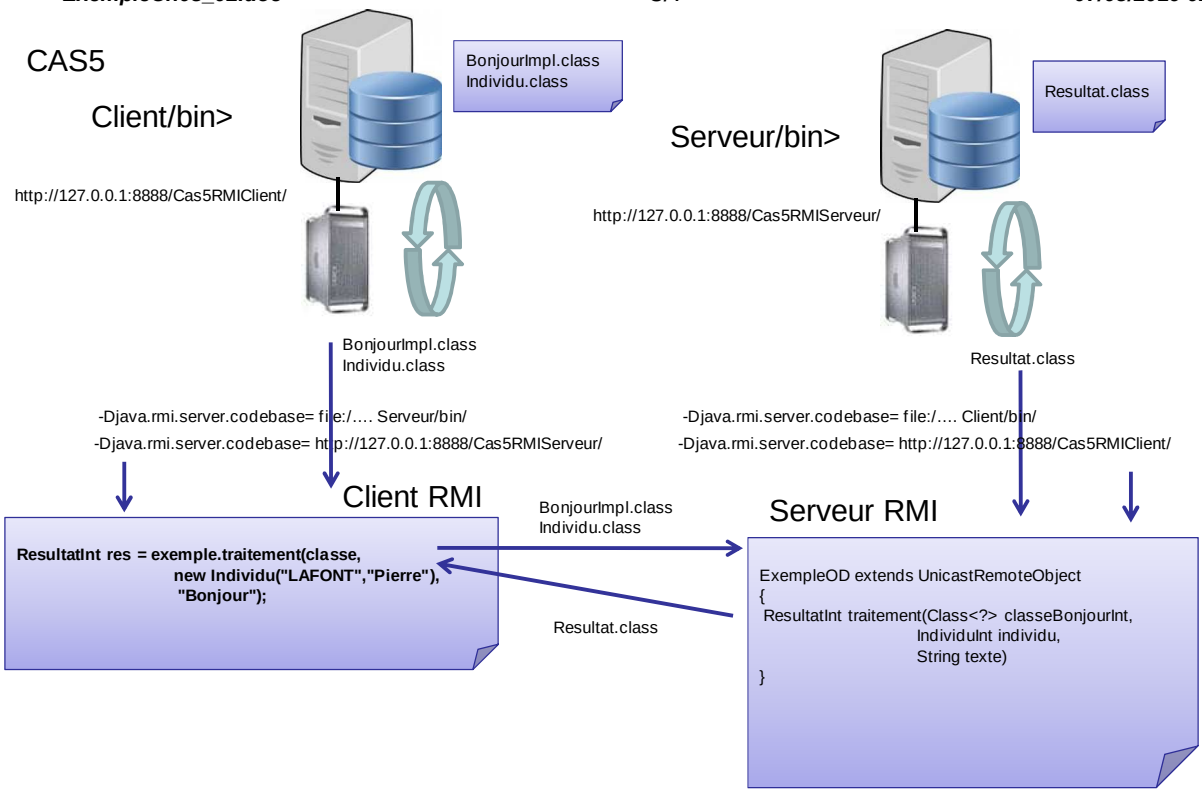

CAS 1
ChargeurDefaut
Bonjour.class
Individu.class
classes/

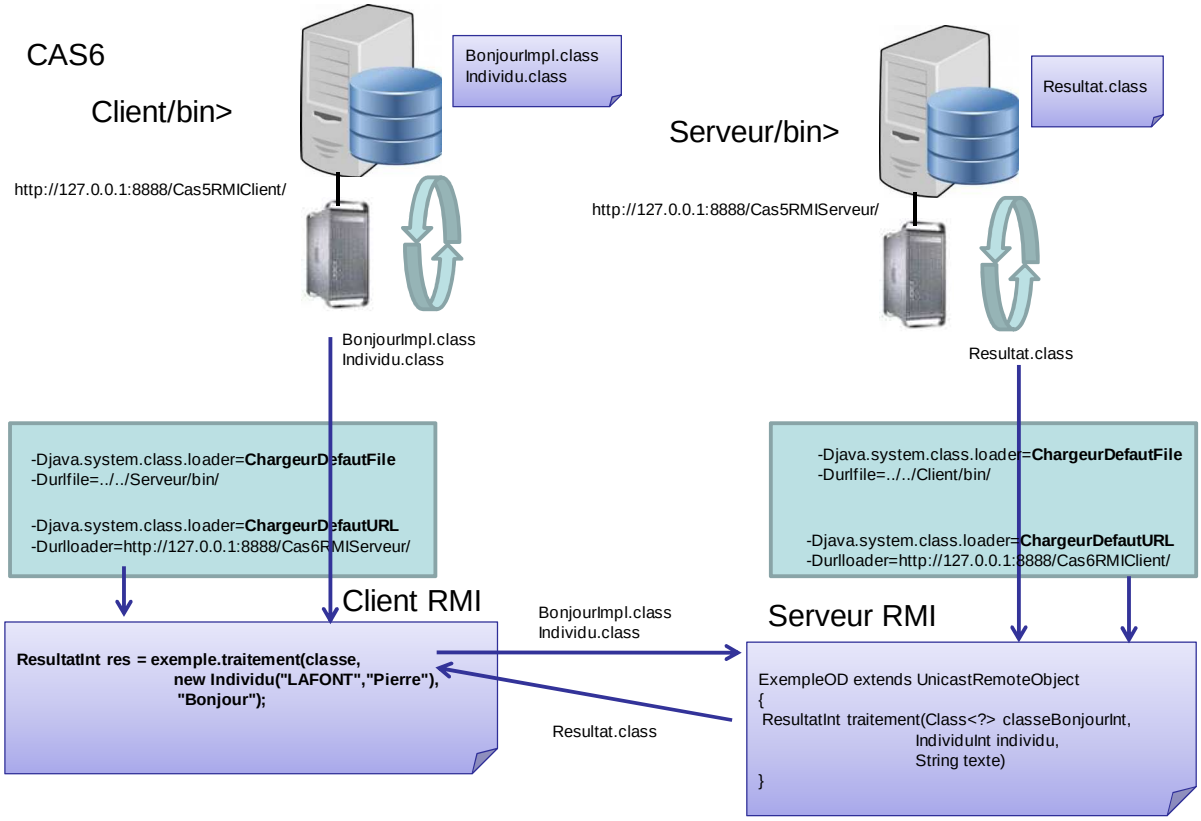

CAS 2
Socket(InetAddress.getLocalHost(),9100);
ChargeurDefaut
Bonjour.class
Individu.class
Bonjour.class
Individu.class
ServerClass
ServerSocket(9100);


CAS 3
URLClassLoader
Bonjour.class
Individu.class
Bonjour.class
Individu.class
http://127.0.0.1:8888/classes/

CAS4


Serveur/bin>
Individu.class
http://127.0.0.1:8888/Cas4RMI/
-Djava.rmi.server.codebase= file:/…. Serveur/bin/
-Djava.rmi.server.codebase= http://127.0.0.1:8888/Cas4RMI/
Individu.class
Client RMI
Serveur RMI
IndividuInt ind = exemple.getIndividu();
Individu.class
ExempleOD extends UnicastRemoteObject
{
public IndividuInt getIndividu(){...}
}





2. Mise en œuvre

Pour les CAS 3, CAS 4 et CAS 5 Il faut exécuter un serveur http.

Pour ma part j'ai choisi EasyPHP comme serveur http (facile à télécharger sur le net et à installer : créer un alias dont le répertoire est le répertoire **classes** de l'arborescence de l'exemple.

Chaque cas, contient un script de compilation **compil.bat** qui compile tous les sources.

Pour chacun des cas, les classes que nous voulons charger dynamiques sont dans le répertoire **classes**.

Alors que les programmes s'exécutent dans le répertoire **bin**. Et bien sur, nous ne mettons pas dans le classpath le répertoire « classes ».

3. Exécution pour le cas 1

cas1Repertoire
- bin
- classes
  - Bonjour.class
  - Individu.class
- compil.bat
- ExempleCh08_02Cas1.java
- ChargeurDefaut.java
- Bonjour.java
- Individu.java
- run1SansClassLoader.bat
- run2AvecClassLoader.bat

Pour désigner son chargeur de classe :
run2AvecClassLoader.bat

java **-Djava.system.class.loader=ChargeurDefaut** -classpath "." ExempleCh08_02Cas1 Paul Auchon

Le chargeur de classe :

```
public class ChargeurDefaut extends ClassLoader
{

    public ChargeurDefaut(ClassLoader parent)
    {
        super(parent);
    }

    public Class findClass(String nom) throws ClassNotFoundException
    {
        File fic;
        FileInputStream fis;
        Class c=null;
        byte[] donnees;

        try{
            // Lecture du fichier de classe
            fic = new File("../classes",
                           nom+".class");
            fis = new FileInputStream(fic);
```

```
              donnees = new byte[(int) fic.length()];
              int octetslus=fis.read(donnees);
              fis.close();

              // Résolution de la classe
              c = (Class)(defineClass(nom,
                                      donnees,
                                      0,donnees.length));

          }catch(Exception           ex){System.out.println("ChargeurDefaut
findClass> "+ex);}
          return c;
      }
```

Un chargeur de classe hérite de ClassLoader.
La méthode findClass est appelée quand le parent ne trouve pas la classe demandée.
Elle lit le fichier .class dans le répertoire classes et retourne la classe.

Le script run1SansClassLoader.bat permet de vérifier que sans le chargeur de classe le programme ne trouve pas la classe Bonjour.

4. Exécution pour le cas 2

cas2Socket
- bin
- classes
  - Bonjour.class
  - Individu.class
- compil.bat
- ExempleCh08_02Cas2.java
- ChargeurDefaut.java
- Bonjour.java
- Individu.java
- run.bat

<u>Pour le serveur de classe</u>
- ServerClass.java
- runServerClass.bat
- file.policy

Le programme « client » (ExempleCh08_02Cas2.java) utilise un chargeur de classe ChargeurDefaut qui interroge un serveur de classe ServerClass.

On exécute d'abord **runServerClass.bat** qui met en place une politique de sécurité :

```
java -Djava.security.manager -Djava.security.policy=../file.policy
                                                        ServerClass
```

file.policy :

```
grant{
  permission java.security.AllPermission;
  permission java.net.SocketPermission "*:9100","connect,accept,resolve";
  permission java.security.policy "write";
};
```

```
public class ServerClass
{
    public static void main(String... args) throws Exception
    {
        System.setSecurityManager(new SecurityManager());
```

Ensuite on execute : run.bat

5. Exécution pour le cas 3

- Lancer EasyPHP qui est configuré avec un alias sur le répertoire classes du cas 3.
- Lancer run.bat

Ici, le chargeur de classe est un chargeur de classe prédéfini : URLClassLoader

```
URL[] urls = new URL[2];
urls[0] = new URL("http://127.0.0.1:8888/ExempleCh08_02Cas3/");
urls[1] = new URL("http://127.0.0.1:80/ExempleCh08_02Cas3/");

URLClassLoader classLoader = URLClassLoader.newInstance(urls);

Class<?> classe = Class.forName("Bonjour", true, classLoader);
```

De plus, on ne surcharge pas le chargeur de classe par défaut car on veut être dynamique sur un éventuel changement de classe **pendant l'exécution**.

Ainsi, pendant l'exécution on peut modifier le code de Individu.java puis le recompiler avec compilIndividu.bat.
On voit bien que la classe qui est utilisée est maintenant la nouvelle.

Ici, le programme boucle sur l'utilisation de la classe Bonjour.

ATTENTION : IL FAUT NE PAS OUBLIER de mettre un "/" à la fin des path de répertoire et d'url.

6. Exécution pour le cas 4

On a séparé le serveur RMI et le client RMI.
Chacun a son script de compilation.

runServeur.bat lance l'OD Exemple OD

runFile.bat utilise un chargement dynamique par fichier.
runURL.bat utilise un chargement dynamique par http. Pour cela, il faut lancer avant EasyPHP avec l'accès au répertoire Serveur/bin

7. Exécution pour les cas 5 et 6

On a séparé le serveur RMI et le client RMI.
Chacun a son script de compilation.

runServeurFile.bat lance l'OD Exemple OD et charge dynamiquement les classes du client par fichier

runServeurURL.bat lance l'OD Exemple OD et charge dynamiquement les classes du client par URL. Pour cela, il faut lancer avant EasyPHP avec l'accès au répertoire Client/bin

runClientFile.bat se connecte à l'OD Exemple OD et charge dynamiquement les classes du serveur par fichier

runClientURL.bat se connecte à l'OD Exemple OD et charge dynamiquement les classes du serveur par URL. Pour cela, il faut lancer avant EasyPHP avec l'accès au répertoire Serveur/bin.

On ne peut lancer qu'un serveur et qu'un client en même temps.