

# Rapport d'un test de EJB

## GREEN CODE LAB CHALLENGE

GROUP MEMBERS:

CHEN Yiqiao

EZQUERRO GARCIA Xabier

GIMENEZ Nicolas

KETevi Maurice

PAN Jianfei

## Installation de l'environnement de travail

Pour préparer ce concours, nous avons installé les environnements de programmation : eclipse, Jboss, openJDK, Visual VM . Pour cette partie, certains de notre groupe a installé NetBeans.

## Travail préparatoire

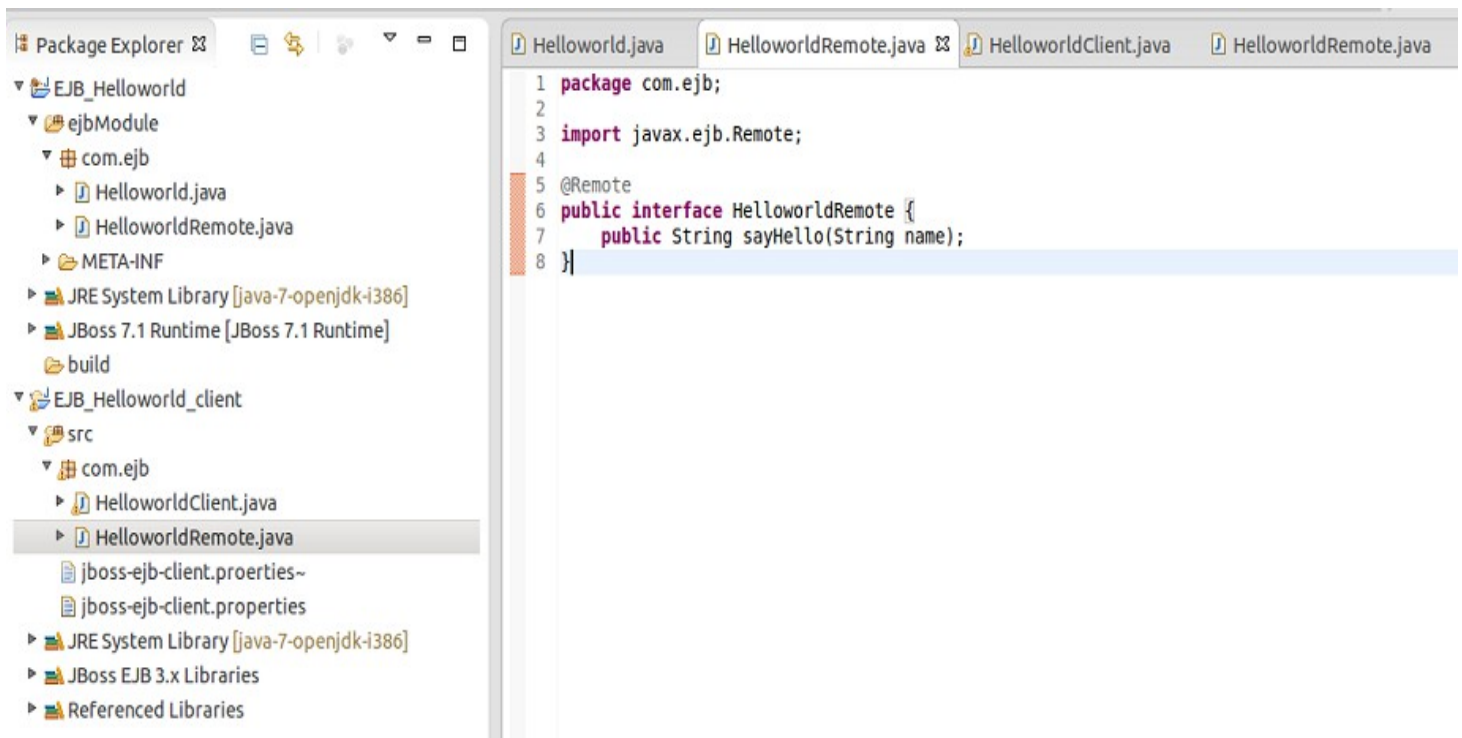
Après avoir installé l'environnement de travail, nous avons fait une recherche documentaire sur EJB, Jboss etc. Cependant nous sommes encore débutant, il reste du travail.

## Test « Hello world »

Pour notre premier test « Hello world », nous avons créé la partie serveur et la partie client :

La partie serveur :

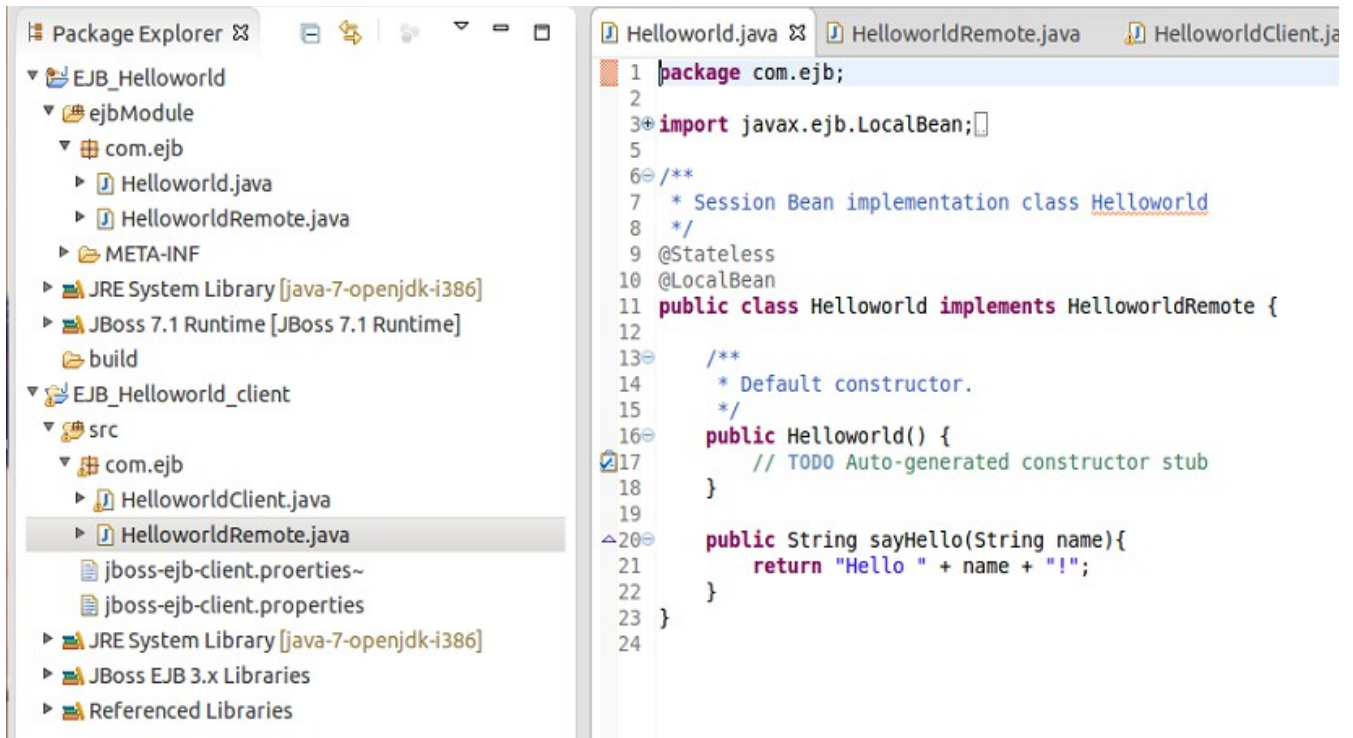
1. Interface « HelloworldRemote » :



Cette interface fait le lien entre le client et le serveur. Elle permet au client de faire aux fonctions se

trouvant dans « HelloWorld.java »

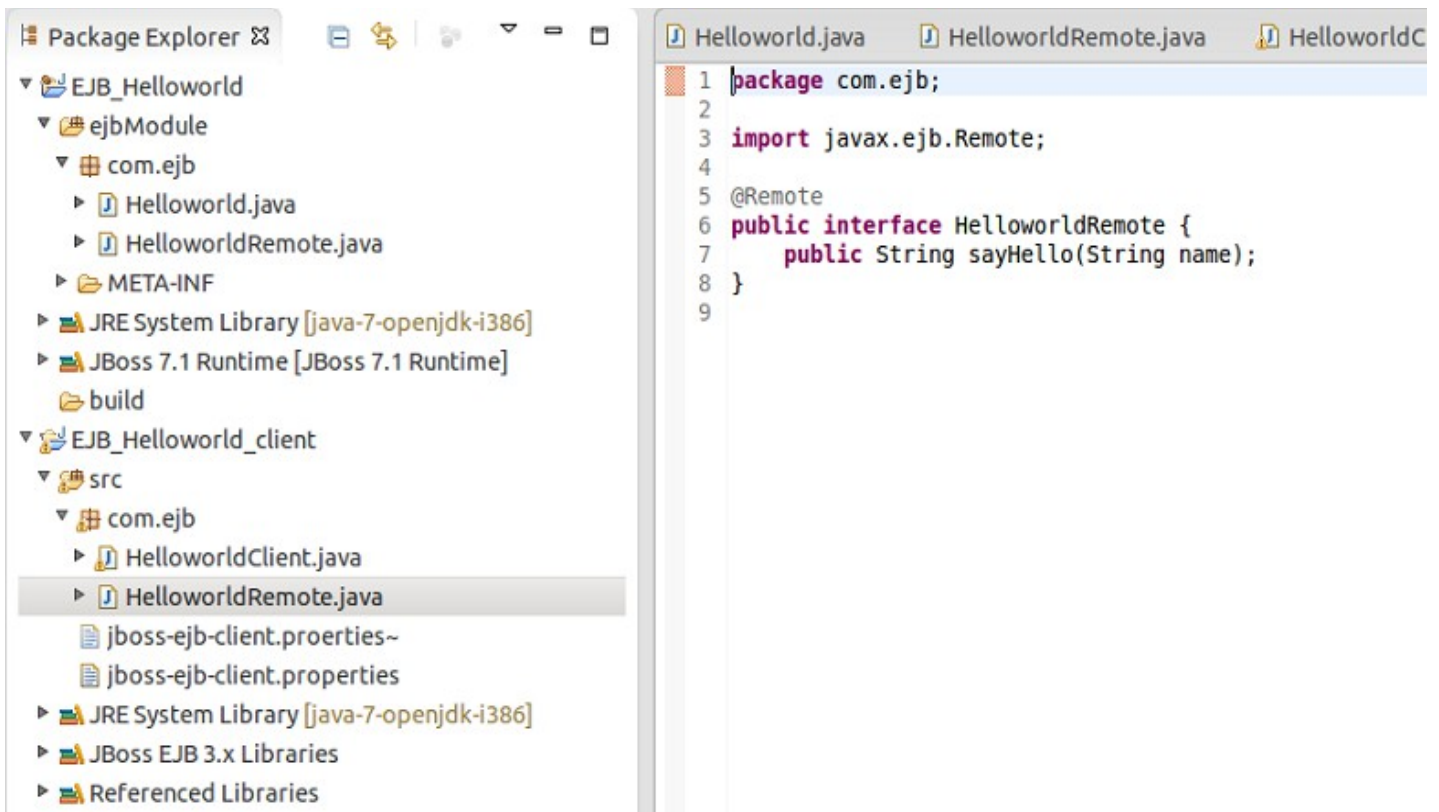
## 2 . Classe « HelloWorld » :



Elle contient toutes les fonctions qui ont été créées pour le client. Elle se situe au niveau du serveur.

La partie client :

### 1. Interface « HelloWorldRemote » :

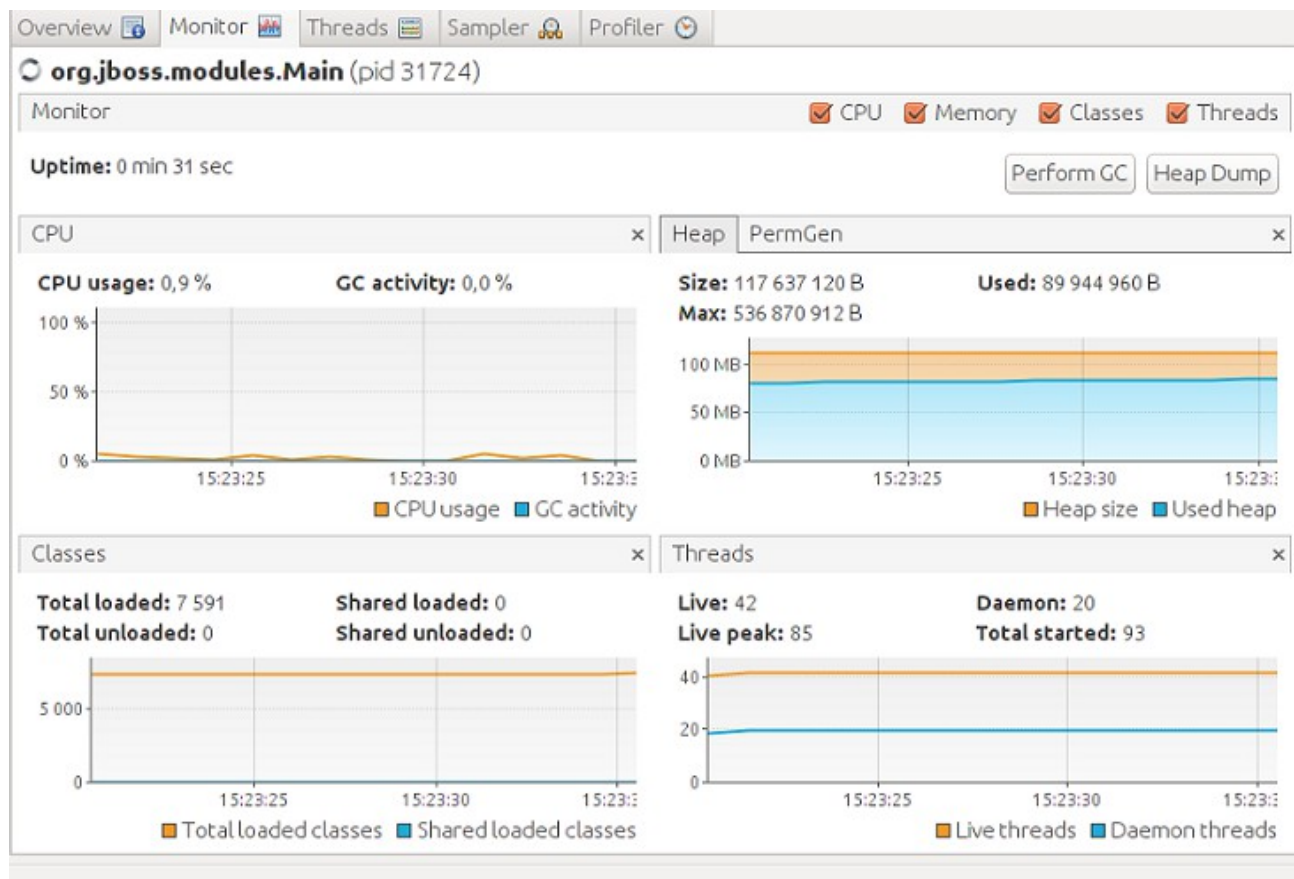


## 2 . Classe « HelloworldClient » :

```
1 package com.ejb;
2
3 import java.util.Hashtable;
4
5
6
7
8
9
10 public class HelloworldClient {
11     public static void main(String[] args) {
12         Properties jndiProperties = new Properties();
13         // Hashtable jndiProperties = new Hashtable();
14         jndiProperties.put("jboss.naming.client.ejb.context", true);
15         jndiProperties.put(Context.URL_PKG_PREFIXES, "org.jboss.ejb.client.naming");
16         try {
17             Context context = new InitialContext(jndiProperties);
18             final String appName = "";
19             final String moduleName = "EJB_Helloworld";
20             final String distinctName = "";
21
22             Object obj = context.lookup("ejb:" + appName + "/" + moduleName + "/" + distinctName + "/" + H
23             System.out.println(obj);
24             HelloworldRemote hwr = (HelloworldRemote) obj;
25             String say = hwr.sayHello("Yiqiao");
26             System.out.println(say);
27         } catch (NamingException e) {
28             e.printStackTrace();
29         }
30     }
31 }
```

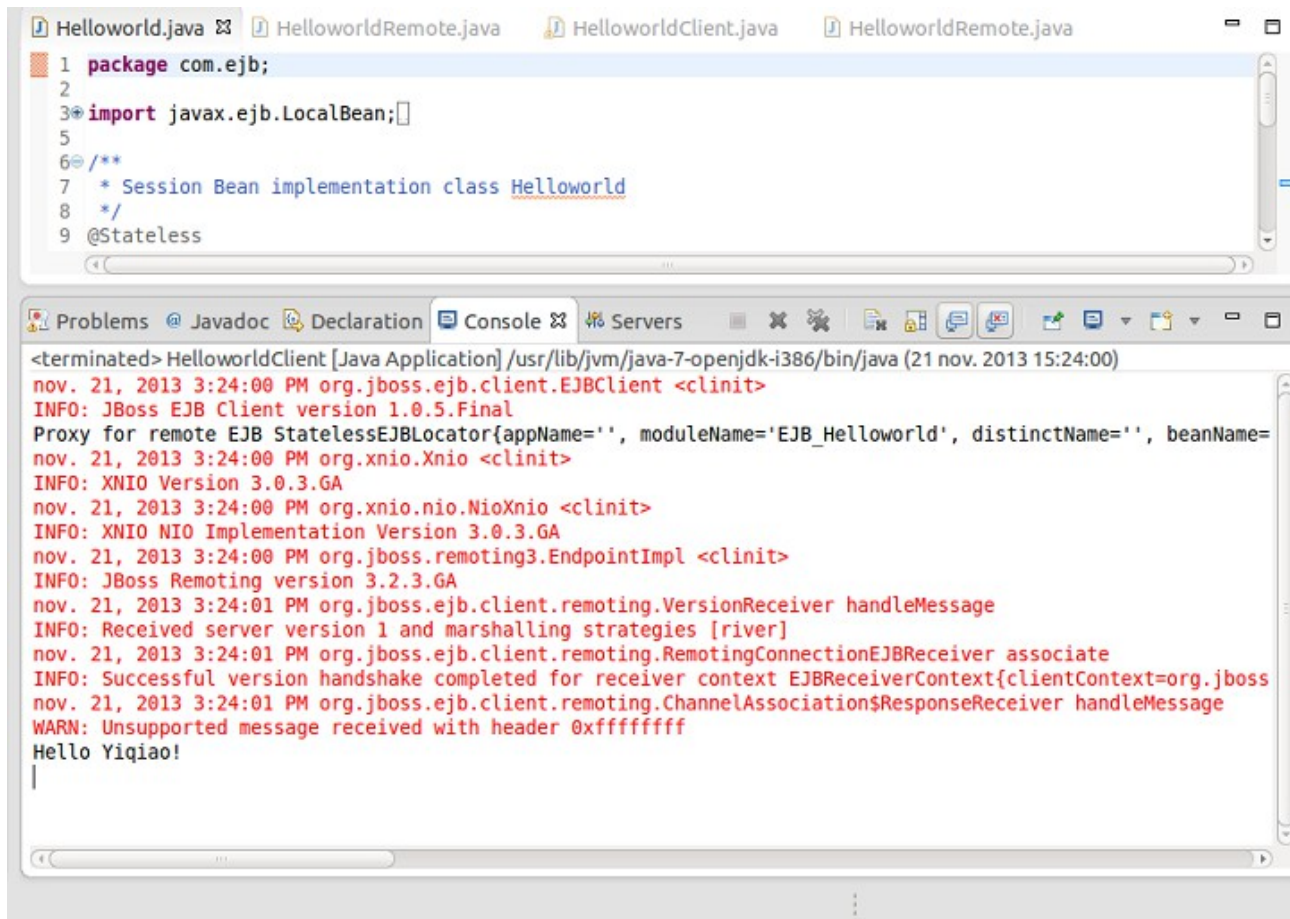
Cette classe sert à envoyer la requête du client.

Pour l'exécuter , on exécute d'abord la classe « Helloworld » de la partie serveur en tant que serveur Jboss. On peut ensuite utiliser le profiler VisualVM :





Puis, on peut exécuter la classe « HelloWorldClient » :



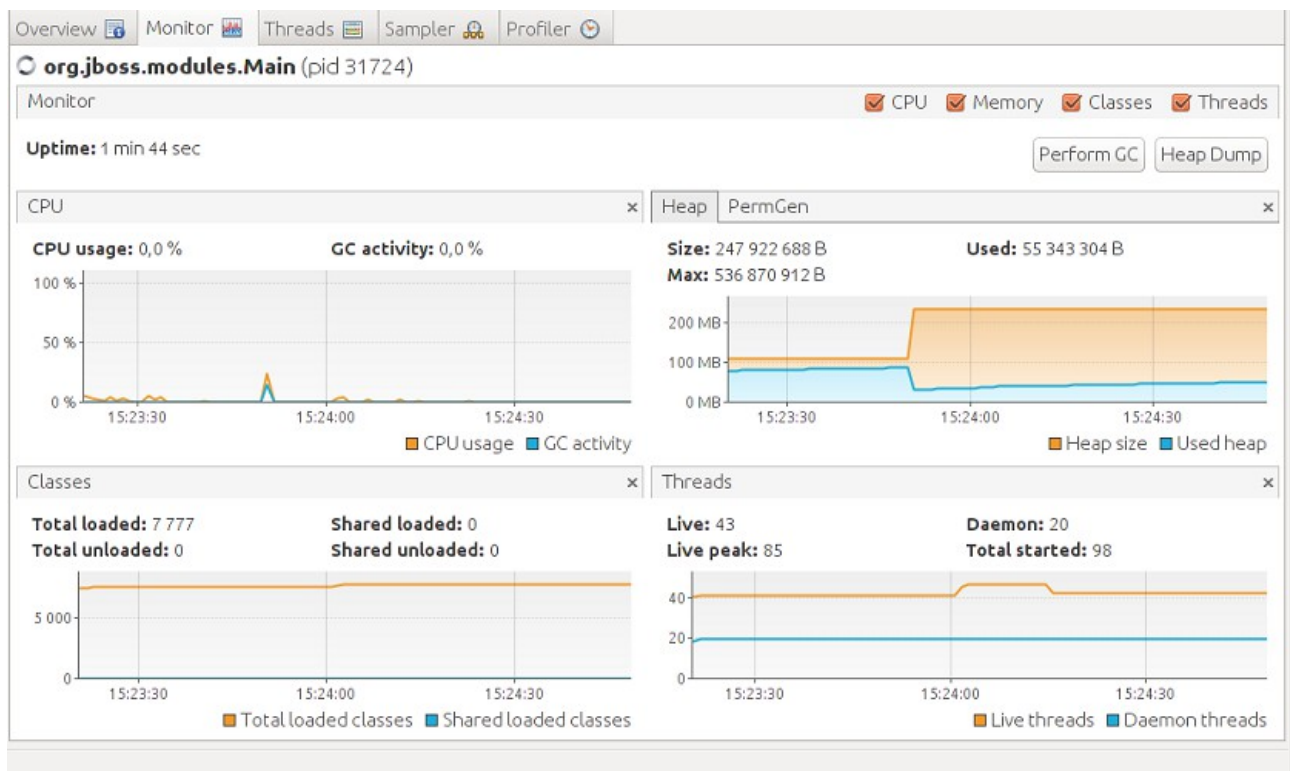
The screenshot shows an IDE with the following code in `HelloWorldClient.java`:

```
1 package com.ejb;  
2  
3 import javax.ejb.LocalBean;  
4  
5  
6 /**  
7  * Session Bean implementation class HelloWorld  
8  */  
9 @Stateless
```

The console output shows the execution of the client, including JBoss EJB Client initialization, proxy creation, and a successful message exchange with the server:

```
<terminated> HelloWorldClient [Java Application] /usr/lib/jvm/java-7-openjdk-i386/bin/java (21 nov. 2013 15:24:00)  
nov. 21, 2013 3:24:00 PM org.jboss.ejb.client.EJBClient <clinit>  
INFO: JBoss EJB Client version 1.0.5.Final  
Proxy for remote EJB StatelessEJBLocator{appName='', moduleName='EJB_Helloworld', distinctName='', beanName=  
nov. 21, 2013 3:24:00 PM org.xnio.Xnio <clinit>  
INFO: XNIO Version 3.0.3.GA  
nov. 21, 2013 3:24:00 PM org.xnio.nio.NioXnio <clinit>  
INFO: XNIO NIO Implementation Version 3.0.3.GA  
nov. 21, 2013 3:24:00 PM org.jboss.remoting3.EndpointImpl <clinit>  
INFO: JBoss Remoting version 3.2.3.GA  
nov. 21, 2013 3:24:01 PM org.jboss.ejb.client.remoting.VersionReceiver handleMessage  
INFO: Received server version 1 and marshallng strategies [river]  
nov. 21, 2013 3:24:01 PM org.jboss.ejb.client.remoting.RemotingConnectionEJBReceiver associate  
INFO: Successful version handshake completed for receiver context EJBReceiverContext{clientContext=org.jboss  
nov. 21, 2013 3:24:01 PM org.jboss.ejb.client.remoting.ChannelAssociation$ResponseReceiver handleMessage  
WARN: Unsupported message received with header 0xffffffff  
Hello Yiqiao!
```

Et enfin utiliser le profiler :



On peut alors lire la consommation énergétique sur les graphes ci-haut.

On voit que la consommation augmente quand on exécute le coté client, plutôt la consommation des mémoires.