Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

JMX
TP04

TAPR2 - Rapport

Filière informatique

Classe I3

Auteurs :

Jocelyn Duc
jocelyn.duc@edu.hefr.ch

Andrea Marcacci
andrea.marcacci@edu.hefr.ch

Enseignant :

Dr Rudolf Scheurer
rudolf.scheurer@hefr.ch

Fribourg, 14 janvier 2013

Table des matières

P1 MBeans already registered

Which MBeans are already registered ? What are they used for ?

Initiallement, on trouve deux MBeans répertoriés sur le serveur :
- `HtmlAdaptorServer` : Permet la gestion de l'agent via un navigateur internet. Il fourni une interface qui nous donne accès aux différents MBeans enregistrés auprès du `MBeanServer`.
- `MBeanServerDelegate` Donne accès aux informations propres au `MBeanServer` telles que la version, le noms Contient des informations en read only à propos du serveur de MBeans tel que la version, le nom, l'identifiant du serveur

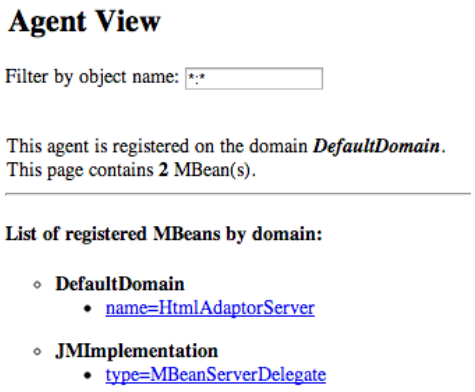


FIGURE 1 – MBean enregistrés de base

P2 MBeans in the DefaultDomain

Make a screenshot of the Agent View with the two MBeans displayed in the DefaultDomain

Après avoir initialisé les deux objets de type `SimpleMBean`, nous les avons enregistré auprès du `MBeanServeur` à l'aide de la commande `registerMBean()`.

Enregistrement des SimpleMBeans

```
1 Simple simple1 = new Simple();
2 Simple simple2 = new Simple();
3
4 ObjectName objectName1 = new ObjectName("DefaultDomain:type=Simple,
      name=Duc");
5 mBeanServer.registerMBean(simple1, objectName1);
6
7 ObjectName objectName2 = new ObjectName("DefaultDomain:type=Simple,
      name=Marcacci");
8 mBeanServer.registerMBean(simple2, objectName2);
```

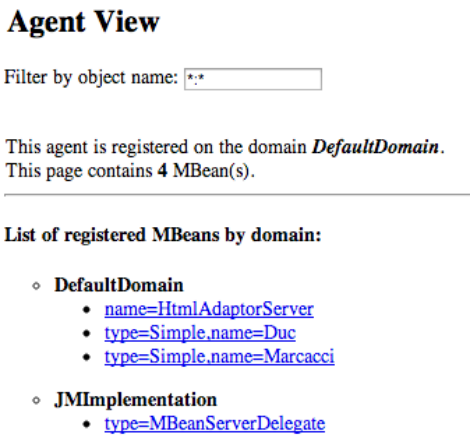


FIGURE 2 – MBean enregistrés dans le DefaultDomain

P3 Named Simple MBeans

Make a screenshot of the MBean View of one of the Simple MBeans with the master attribute set to your name

Sur la capture suivante, nous pouvons voir ce qu'il y a à l'intérieur d'un `MBean` enregistré.

Sur un `MBean`, on retrouve 3 attributs :
– `master` : Attribut de type `String`, modifiable depuis l'interface.
– `state` : Attribut de type `String`, modifiable depuis l'interface.
– `nbChanges` : Attribut de type `int`, non modifiable depuis l'interface.

À noter que les attributs `master` et `state` sont modifiables depuis l'interface parce qu'au niveau de l'implémentation, des `setters` sont définis, contrairement à l'attribut `nbChanges`, qui n'en a pas. Ce dernier est donc affiché en `read-only`.

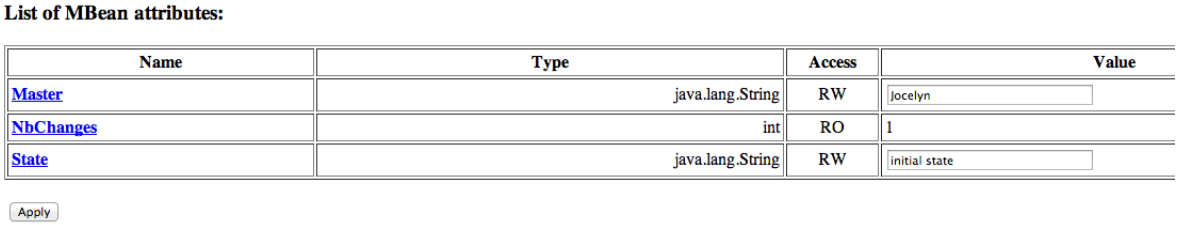


FIGURE 3 – Vue d'un Simple MBean

P4 nbChanges attribute

What do you observe concerning the attribute nbChanges when you change the attribute master ? Explain and fix the problem.

Initiallement, dès la modification de `master` ou `state`, on constate que l'attribut `nbChanges` s'incrémente de 2. Lors de la mise à jour, cette dernière a lieu sur les deux attributs, en écrasant la valeur préalablement sauvée.

Sortie console

```
1 - attribute 'master' has been changed.
2 - attribute 'state' has been changed.
```

Pour pallier le problème, nous avons modifier l'implémentation des méthodes `setMaster()` et `setState()` :

Enregistrement des SimpleMBean

```
1  public void setMaster(String s) {
2    if(!this.getMaster().equals(s)){
3      System.out.println("Master changed");
4      attributeChanged("master");
5      master = s;
6    }
7  }
8
9  public void setState(String s) {
10   if(!this.getState().equals(s)){
11     System.out.println("State changed");
12     attributeChanged("state");
13     state = s;
14   }
15 }
```

P5 Setting the master attribute (using the HTML Adaptor)

Is it possible (using the HTML Adaptor) to set the master attribute to an empty string ? If not, what's the reason ?

Il n'est pas possible d'enregistrer un `String` vide comme valeur de l'attribut `master`. Un `String` vide est bel et bien passé en paramètre si l'on regarde l'URL appelée.

L'agent ne prend pas en compte les valeurs vides et n'écrase pas les valeurs préalablement enregistrées.

P6 JMXMP connector

Make a screenshot of the MBean View of the JMXMP connector server.

Après l'impémentation de `JMXMP_Agent.java`, nous pouvons relancer le serveur et découvrir de nouveaux objets enregistrés. En ouvrant le connecteur JMXMP, on y retrouve sa configuration.

Certains champs ne sont pas visibles (Type not supported), puisque leur affichage en format texte n'est pas supporté sur l'`HtmlAdaptorServer`.

JMXMP_Agent.java

```
1  public class JMXMP_Agent {
2
3    public static void main(String argv[]) {
4      try {
5        // Instanciate the MBean server
6        System.out.println("Create MBean server ...");
7        MBeanServer mBeanServer = ManagementFactory.
             getPlatformMBeanServer();
8        // Create and start an HTML protocol adaptor
9        System.out.println("Create, register and start an HTML
             Adaptor server ...");
10       HtmlAdaptorServer htmlAdaptor = new HtmlAdaptorServer();
11       mBeanServer.registerMBean(htmlAdaptor, null);
12       htmlAdaptor.start();
13       ObjectName objectName = new ObjectName("Connectors:type=JMXMP
             ,id=remote");
14       JMXServiceURL url = new JMXServiceURL("jmxmp", null, 1234);
15       JMXConnectorServer jmxConnectorServer =
             JMXConnectorServerFactory.newJMXConnectorServer(url, null,
              mBeanServer);
16       mBeanServer.registerMBean(jmxConnectorServer, objectName);
17       jmxConnectorServer.start();
18     }
19     catch (Exception e) {
20       e.printStackTrace();
21       return;
22     }
23   }
24 }
```

List of MBean attributes:

Name	Type	Access	Value
Active	boolean	RO	true
Address	javax.management.remote.JMXServiceURL	RO	*Type Not Supported*: [service:jmx:jmxmp://ironmac.sofr.hefr.lan:1234]
Attributes	java.util.Map	RO	*Type Not Supported*: [{}]
ConnectionIds	java.lang.String[]	RO	view the values of ConnectionIds
MBeanServerForwarder	javax.management.remote.MBeanServerForwarder	WO	*Type Not Supported*

FIGURE 4 – Attribut du connecteur JMXMP

P7 jconsole

While you are still connected : open jconsole and connect to the agent. Make a screenshot of jconsole displaying all the attributes of the connector's MBean (double-click on the data types written in bold to see values).

Il suffit de démarrer la console Java après avoir lancer l'agent et rechercher les attributs demandés.

Lancement de jconsole

```
1 jconsole
```

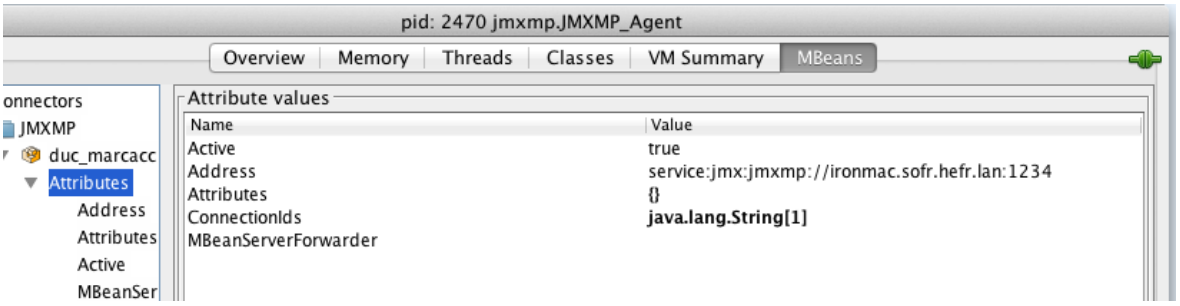


FIGURE 5 – Attribut du connecteur MBeanServer dans jconsole

P8 MBeanServerConnection and MBeanServer

What are the interfaces MBeanServerConnection and MBeanServer good for ? What are the differences (with respect to e.g. location / available methods) between those two interfaces ?

- `MBeanServerConnection` : Il est utilisé pour la gestion des `MBean` par le client. Grâce à lui, on peut créer, modifier et supprimer des `MBean`. Il donne un accès aux getters et setters.
- `MBeanServer` : Il permet les mêmes fonctionnalités que le `MBeanServerConnection`, mais du côté de l'agent. De plus, on y trouve d'autres méthodes à disposition : `getClassLoader()`, `deserialize()`, `instatiate()`, `registerMBean()`.

P9 MBean proxy dynamically created

Explain how it is possible for the JVM to create the MBean proxy dynamically. What information is basically needed for this ?

Grâce au mécanisme de réflexion de Java, ce dernier permet la création de `MBean` de manière dynamique pendant l'éxécution. Java offre la possibilité de charger dynamiquement des classes sans connaître leur nom à l'avance.

De ce fait, un `MBean` proxy peut être instancié de cette façon :

Instantiation d'un MBean proxy

```
1 MBeanServerInvocationHandler.newProxyInstance(mbs, name, Intf.class
      , false);
```

Ici, un bref descriptif des attibuts utilisés pour l'instantiation :
- `mbs` : Pour l'établissement d'une connexion via le `MBeanServer`.
- `name` : Nom de l'objet
- `Intf.class` : Méthodes mises à disposition au client par le proxy
- `false` : Indique si le `MBean` étend ou non la classe `NotificationBroadcaster`

P10 Notifications subscription

Subscribe to the notifications of the MBean and make a screenshot of the notifications received when changing attributes.

Premièrement, il faut modifier l'implémentation de la classe `Simple`, pour permettre la diffusion de notifications, comme suit. Ainsi, dès qu'un changement a lieu, une notification est envoyée.

Simple.java

```
1  public void setMaster(String s) {
2    if (!master.equals(s)) {
3      attributeChanged("master");
4      AttributeChangeNotification notif = new
           AttributeChangeNotification(this, nbChanges, m.
           currentTimeMillis(), "Attribute 'master' has changed!", "
           master", "String", master, s);
5      sendNotification(notif);
6    }
7    master = s;
8  }
9
10 public void setState(String s) {
11   if (!state.equals(s)) {
12     attributeChanged("state");
13     AttributeChangeNotification notif = new
           AttributeChangeNotification(this, nbChanges, m.
           currentTimeMillis(), "Attribute 'state' has changed!", "
           state", "String", state, s);
```

```
14      sendNotification(notif);
15    }
16    state = s;
17  }
```

En plus de cela, on redéfinit encore la méthode suivante, sans quoi, nous n'aurons pas de résultat dans jconsole.

Simple.java

```
1  public MBeanNotificationInfo[] getNotificationInfo() {
2    String[] types = new String[] {AttributeChangeNotification.
       ATTRIBUTE_CHANGE};
3    String name = AttributeChangeNotification.class.getName();
4    String description = "An attribute of this MBean has changed";
5    MBeanNotificationInfo info = new MBeanNotificationInfo(types,
       name, description);
6    return new MBeanNotificationInfo[] {info};
7  }
```

Il faudra encore cliquer sur le bouton Subscribe.

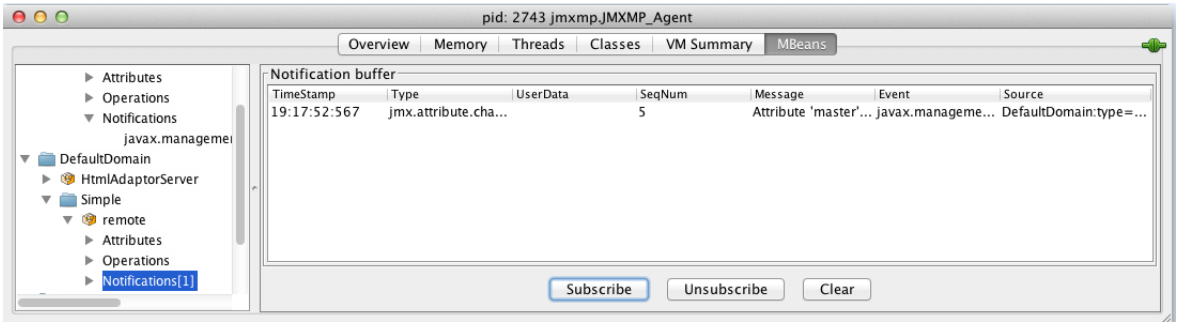
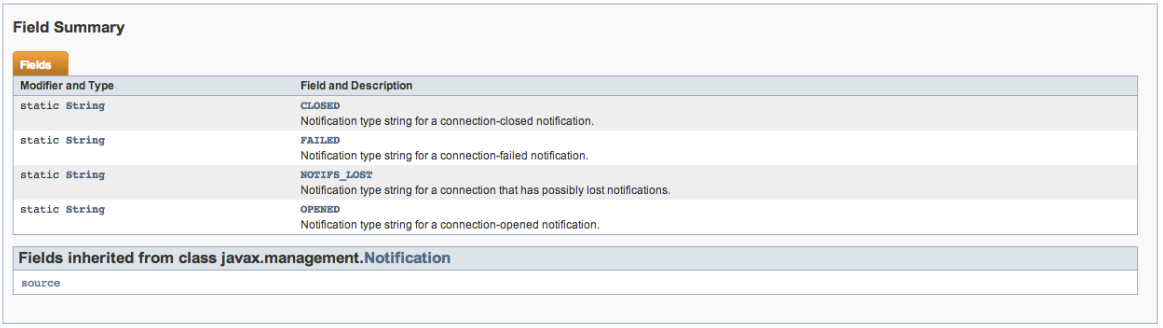


FIGURE 6 – Notifications dans jconsole

P11 JMXMP notifications

List up the notifications / notification types provided by the JMXMP connector and by the platform MBean server.

Selon la documentation officielle, voici une liste des types de notifications qui peuvent être propagées par le connecteur `JMXMP` et par le serveur `MBean` :



Field Summary

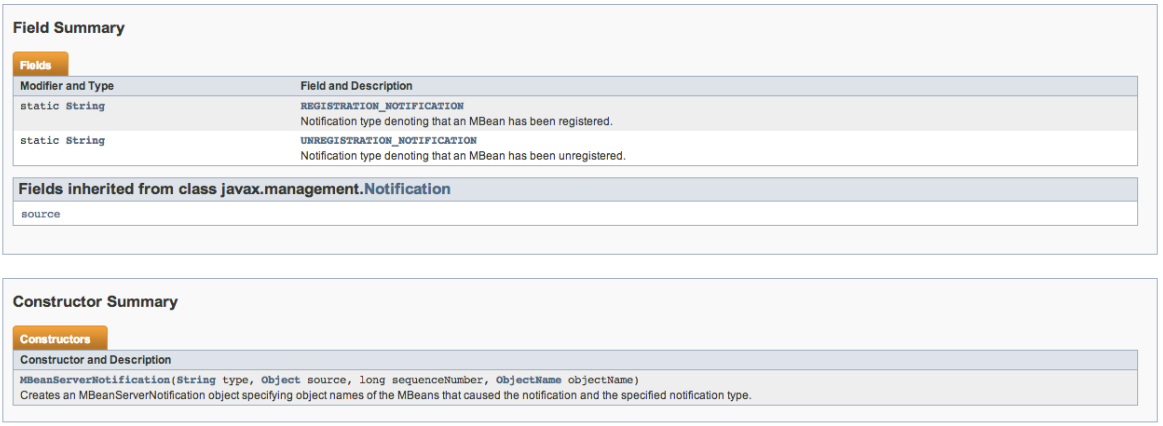
Fields

Modifier and Type	Field and Description
static String	CLOSED Notification type string for a connection-closed notification.
static String	FAILED Notification type string for a connection-failed notification.
static String	NOTIFS_LOST Notification type string for a connection that has possibly lost notifications.
static String	OPENED Notification type string for a connection-opened notification.

Fields inherited from class javax.management.Notification

source

FIGURE 7 – JMXConnectionNotification



Field Summary

Fields

Modifier and Type	Field and Description
static String	REGISTRATION_NOTIFICATION Notification type denoting that an MBean has been registered.
static String	UNREGISTRATION_NOTIFICATION Notification type denoting that an MBean has been unregistered.

Fields inherited from class javax.management.Notification

source

Constructor Summary

Constructors

Constructor and Description
MBeanServerNotification(String type, Object source, long sequenceNumber, ObjectName objectName) Creates an MBeanServerNotification object specifying object names of the MBeans that caused the notification and the specified notification type.

FIGURE 8 – MBeanServerNotification