JMX : Java Management eXtension

Gaël Thomas
gael.thomas@lip6.fr

Université Pierre et Marie Curie
Master Informatique
M2 – Spécialité SAR

Introduction

Administration d'un serveur d'application

- ✓ Administration humaine
  Création d'une interface Web, d'une console (GUI ou non)
- ✓ Administration automatique
  Création d'une API d'administration

Problème

- ✓ Uniformiser les API d'administration
- ✓ Uniformiser les consoles d'administration (Web, GUI, text)

Solution proposée par JMX

- ✓ Exposer des paramètres configurables d'objets Java de façon uniforme
- ⇒ Offrir des interfaces de configuration

Introduction

JMX : Java Management eXtension

- ✓ Offre des API d'accès aux objets à distance (pull)
- ✓ Offre une API de notification (push)

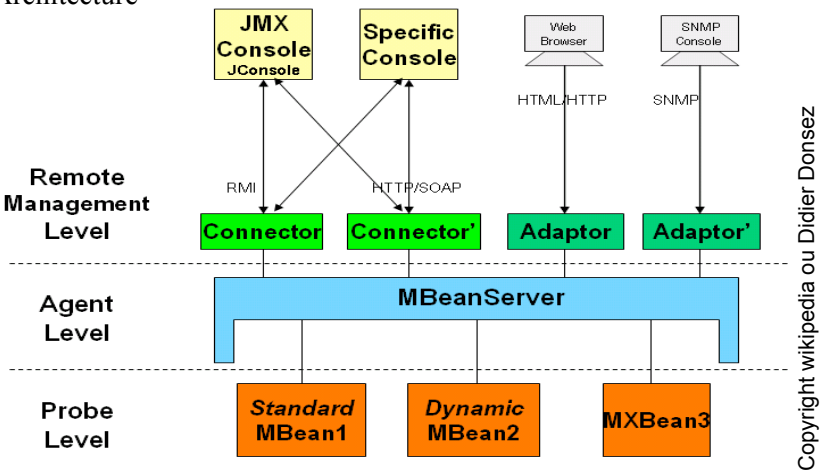
Largement adoptée pour configurer, observer des applications Java

JMX : un nouveau middleware à composants

- ✓ Managed Bean (MBean) : un objet du middleware
- ✓ MBeanServer : le serveur d'application
- ✓ Console : le client du MBean

Introduction

Architecture



Développement de MBean

MBean : expose une interface

- ✓ Ensemble de getter et setter
- ✓ Ensemble de méthode accessibles à distance
- ✓ Une auto-description

Principe de programmation :

- ✓ Développer une interface de Mbean
- ✓ Implanter l'interface
- ✓ Exposer le MBean

Packages des MBean :

`java.lang.management.* et javax.management.*`

Développement de MBean

Définir l'interface

```
interface HelloMBean {
  public void   sayHello();        // opération
  public int    add(int x, int y); // opération
  public void   setSolde(int x);   // variable en R/W
  public int    getSolde();
  public String getName();         // variable en R
}
```

Convention : l'interface a comme nom ImplMBean

Développement de MBean

Développement de l'objet Java

```
public class Hello implements HelloMBean {
  public void   sayHello() { System.out.println("Hello"); }
  public int    add(int x, int y) { return x+y; }
  public String getName() { return name; }
  public int    getSolde() { return solde; }
  public void   getSolde(int s) { return s; }

  private int solde;
  private String name;
}
```


Développement de MBean

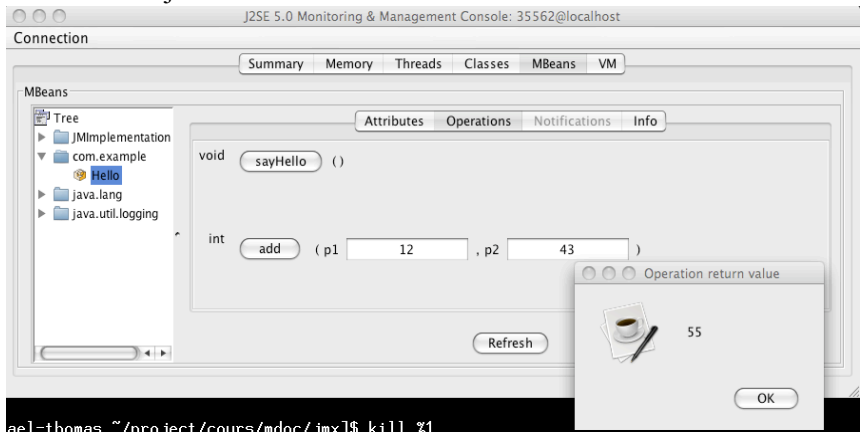
Exposer le MBean

```
public static void main(String[] args) throws Exception {
  MBeanServer mbs =
        ManagementFactory.getPlatformMBeanServer();
  // ou MBeanServerFactory.createMBeanServer();
  ObjectName name = new ObjectName("com.example:type=Hello");
  Hello mbean = new Hello();
  mbs.registerMBean(mbean, name);
  …
```

Lancer l'application : java -Dcom.sun.management.jmxremote …

Développement de MBean

La console : jconsole



Développement de MBean

Nom de Bean : le ObjectName

- ✓ Un domaine : com.example
- ✓ Une liste de propriété : type=Hello

Quelques conventions :

- ✓ Pas de domaine : domaine par défaut (racine)
- ✓ Ordre des propriétés non significatif : yop:a=2,b=3 ⇔ yop:b=3,a=2
- ✓ Propriété *type* requise
- ✓ Propriété *name* souvent utilisée
- ✓ Arborescence de domaine

  :type=A

  mdoc:type=B

  mdoc.cinema:type=C

Développement de MBean

Interception de l'enregistrement

```
interface MBeanRegistration{
  void postDeregister();
  void postRegister(Boolean registrationDone);
  void preDeregister();
  ObjectName preRegister(MBeanServer server, ObjectName name);
}
```


Développement de MBean

Notification d'événements

MBean offre une interface d'enregistrement à la console

- ✓ La console expose un écouteur
- ✓ La console peut posséder un filtre

```
interface NotificationBroadcaster{
  void addNotificationListener(NotificationListenerlistener,
            NotificationFilterfilter, Object handback);
  void
    removeNotificationListener(NotificationListenerlistener);

   MBeanNotificationInfo[] getNotificationInfo();
}
```

Limitation des MBeans

Données transmises entre console et MBean simples

- ✓ Type primitifs
- ✓ Chaînes de caractères

Problème : comment gérer des données complexes

Solution proposée les MXBean (Java 6)

- ✓ Définition de données complexe
- ✓ Auto-description des paramètres
- ✓ Annotation sur les paramètres pour l'auto-description

Les MXBean

Définition d'un MXBean

- ✓ Comme un MBean, mais remplace MBean par MXBean
- ✓ Paramètres construits autorisés

```
interface MonBeanMXBean {
  public TypeComplex getComp(int x, int y);
}

public class MonBean implements MonBeanMXBean {
  public TypeComplex getComp(int x, int y) {
    return new Comp(1, 2);
  }
}
```


Les MXBean

Définir le type complexe

Une unique annotation **`javax.bean.ConstructorProperties`**

```
class TypeComplex {
  private int x;
  private int y;

  @ConstructorProperties({x, y})
  public Comp(int x, int y) {
  }

  public int getX() { return x; }
  public int getY() { return y; }
}
```


Les MXBean

Principe :

- ✓ Type encapsulé dans un type **générique** lors du transport : **`javax.management.openmbean.CompositeData`**
- ✓ Accesseurs pour les champs lors de la reconstruction
  - Object getKey(String name)
- ✓ Annotation génère une méthode :
  `static LeTypeDeDonnee from(CompositeData data)`

Côté console :

- ✓ Auto-description : affichage du type de donnée dans la console
- ✓ Emission et réception de CompositeData

Côté serveur de Bean :

- ✓ Transformation des CompositeData en structure lors de réception/émission

Les MXBean

Mais pourquoi ne pas utiliser l'API réflexive Java?

- ✓ La console n'est pas forcément écrite en Java
- ✓ Et même dans ce cas, il faudrait que la classe complexe soit dans le ClassPath de la console

Les connecteurs

But de JMX : rendre les MBean accessibles à distance

Connecteurs : expose les MBean à distance

- ✓ De nombreux connecteurs disponibles
  RMI, IIOP, HTTP/SOAP, HTTP/HTML, SNMP, JMXMP (JMX Message Protocol)
- ✓ Nécessite la librairie jmxtools.jar (JMX 1.2.1 Reference Implementation)

Les connecteurs

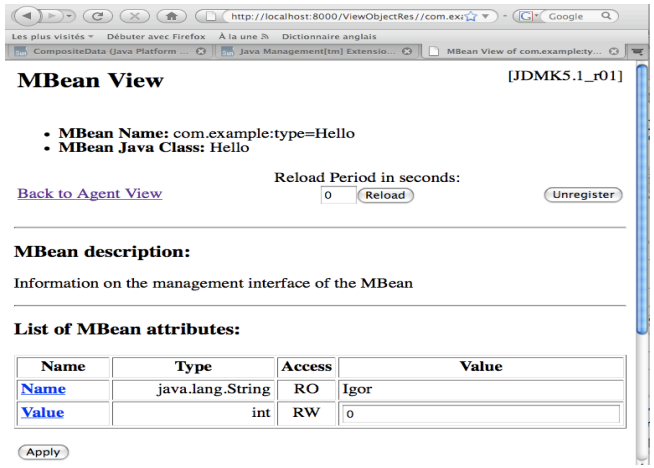
Exemple : le connecteur HTTP

```
import com.sun.jdmk.comm.HtmlAdaptorServer;

MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
ObjectName name = new ObjectName("com.example:type=Hello");
HtmlAdaptorServer adapter = new HtmlAdaptorServer();
adapter.setPort(8000);
ObjectName adapterName= new
        ObjectName("SimpleAgent:name=htmladapter,port=8000");
mbs.registerMBean(adapter, adapterName);
Hello mbean = new Hello();
mbs.registerMBean(mbean, name);
adapter.start();
```


Les connecteurs

Exemple avec le connecteur HTTP



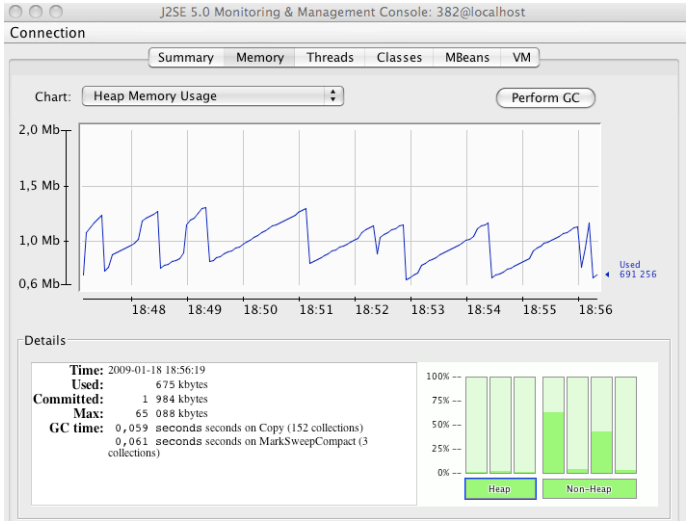
Introspection de la JVM

De nombreux MXBean directement offerts par la JVM

- ✓ `ClassLoadingMXBean` : info sur les classes chargées
- ✓ `CompilationMXBean` : info sur le JIT
- ✓ `GarbageCollectorMXBean` : info sur les GC (copy, mark/sweep)
- ✓ `MemoryManagerMXBean` : info sur la mémoire
- ✓ `MemoryMXBean` : info sur la mémoire
- ✓ `MemoryPoolMXBean` : info sur la mémoire
- ✓ `OperatingSystemMXBean` : info sur archi/os
- ✓ `RuntimeMXBean` : info sur la JVM
- ✓ `ThreadMXBean` : info sur les threads

Introspection de la JVM

Exemple Gestion Mémoire



Conclusion

JMX : API d'administration Java

- ✓ Simple à mettre en œuvre
- ✓ Uniquement orienté administration
- ✓ Gestion de types complexes de façon générique

- ✓ API très utile et encore méconnue

MBean : composants orientés administration