JMX
Java Management eXtension API

Cnam Paris
jean-michel Douin, douin au cnam point fr
version 19 Février 2008

Architecture à base de composants

Sommaire

- **Objectifs**
  - Supervision de JVM
- **Une Première approche**
  - Un exemple
    - Un Manageable Bean (MBean)
    - Un Agent
    - Une supervision
    - Une démonstration
- **De bien plus près : 3 niveaux**
  - Instrumentation
    - Standard, dynamic, open, model Beans et MXBeans.
  - Agent / serveur
    - Installation et accès aux « MBeans »
  - Distribué
    - Connecteurs et adaptateurs

Bibliographie utilisée

- **La présentation de Christophe Ebro**
  - **http://rangiroa.essi.fr/cours/internet/02-JMX-partie2.pdf**
- **L'indispensable tutoriel de Sun**
  - **http://java.sun.com/docs/books/tutorial/jmx/index.html**
  - **http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/**
- **Hello world**
  - http://java.sun.com/j2se/1.5.0/docs/guide/jmx/examples.html
- **Le blog de Daniel Fuchs**
  - http://blogs.sun.com/jmxetc/
- JMX et Design Patterns chez hp
  - http://devresource.hp.com/drc/resources/jmxbestp_dp_pres/index.jsp
- Côté développeur ibm
  - http://www-128.ibm.com/developerworks/java/library/j-jmx2/

Orienté MX4j, date un peu http://admc.com/blaine/howtos/jmx/

http://www.xmojo.org/products/xmojo/index.html

Spécialisé MXBean / accès à la JVM

- **http://www-128.ibm.com/developerworks/java/library/j-mxbeans/**

Divers

- **http://www-adele.imag.fr/users/Didier.Donsez/ujf/sujet/jmx.html**

Pré-requis

- **Notions de**
  - **Client/serveur,**
    - **Protocole JRMP(rmi) et HTTP**
  - **Introspection**
  - **Patrons Fabrique, Publish-Subscribe, Proxy**

JMX : Objectifs

- **Gestion/administration de Ressources**
  - **Matérielles comme logicielles**
- **Configuration/déploiement**
  - **Statique et dynamique**
- **Contrôle**
  - **Du Cycle de vie : start/stop/suspend/resume**
  - **De la Charge en vue d'une meilleure répartition**
- **Supervision**
  - **Performance**
  - **Des erreurs/ exceptions**
  - **De l'état (cf. cycle de vie)**

JMX API

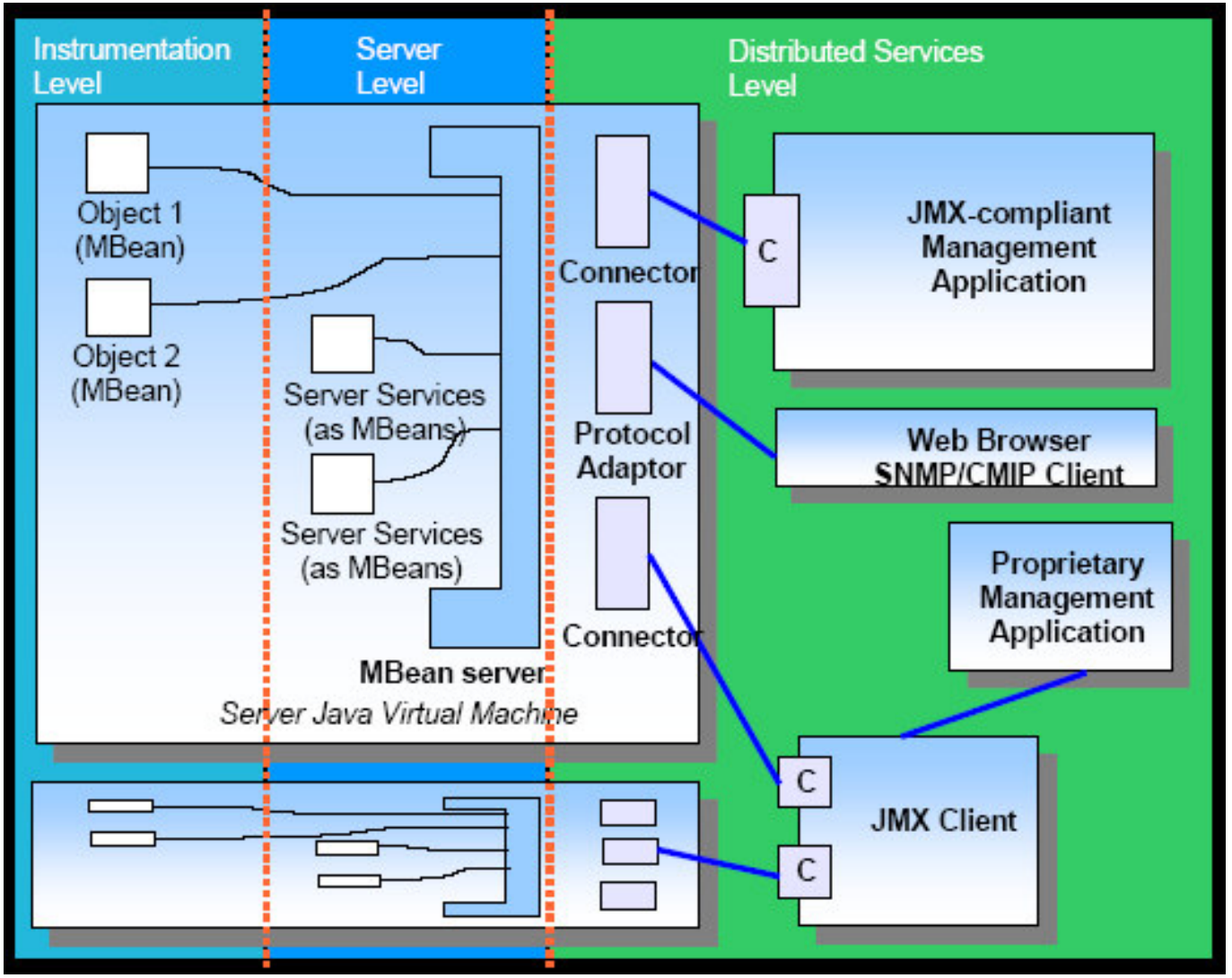
- **Hypothèse : tout est JAVA-JVM**
  - **Ressources matérielles**
  - **Ressources logicielles**
- **Adoptée par de nombreuses entreprises**
  - http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/jmxadoption.jsp
- **Outils prédéfinis**
  - **jconsole, rmi/http/snmp/jini adapter**

Architecture

- **3 niveaux**
  - **Instrumentation**
    - **Gestion de la ressource par un composant (MBean, Managed Bean)**
  - **Agents**
    - **Initialisation, installation et contrôle des MBean**
  - **Distribué/intégration**
    - **Adaptateurs de Protocoles**
    - **RMI/HTTP/SNMP-CMIP**
    - **Sécurité**

    - **Common Management Information Protocol (CMIP), protocole ISO**
    - **Simple Network Management Protocol (SNMP)**

Schéma de Christophe Ebro Sun



- **3 niveaux**

  Instrumentation, serveur, distribué

Objectifs rappel contexte JVM

- **Accès à une JVM « de l'extérieur »**
  - **Une JVM en cours d'exécution …**

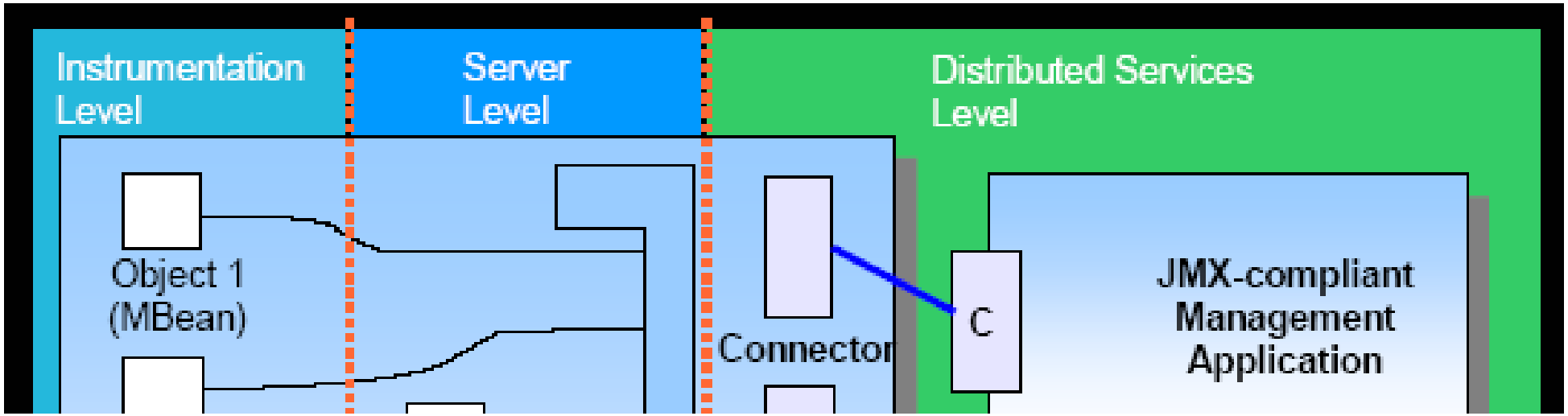
- **Gestion d'objets Java** *déjà en place…* *MBean*
  - **Accès aux attributs**
    - **Lecture/écriture**
  - **Appels de méthodes**
    - **Passage de paramètres / retour de résultat**
  - **Notifications, installations d'observateurs**
    - **évènements**

- **Ajout dynamique de nouveaux objets** *MBean spécialisé*
  - **Code des classes en place**
  - **Code « hors-place » à télécharger**

Par l'exemple : un capteur…

- **SensorMBean** — **Instrumentation**
  - **Standard, dynamic, …**
- **SensorAgent** — **Serveur**
- **Outil jconsole** — **Distribué**

Un exemple comme présentation



- **SensorMBean**
- **Sensor**

Instrumentation

- **SensorAgent**
- **SensorClient**

Serveur

Outil de Sun jconsole

Distribué

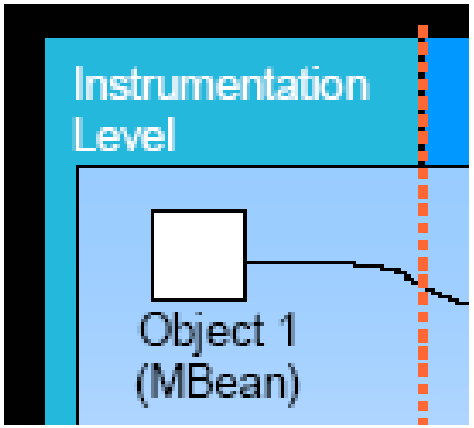
Instrumentation, Standard MBean

```
// un capteur comme ressource
public interface SensorMBean {

   // getter/setter
   public int getValue();
   public void setValue(int val);

   // operations
   public void reset();

}
```

MBean suffixe imposé …

Sensor implements SensorMBean

```
public class Sensor implements SensorMBean{

  private final int PERIOD;
  private int value;
  private Acquisition local; // un thread interne

  public Sensor(int period){
    this.PERIOD = period;
    local = this.new Acquisition();
  }

  public synchronized int getValue(){
    return value;
  }

  public synchronized void setValue(int value){
    this.value = value;
  }

  // operations
  public void reset(){…
```



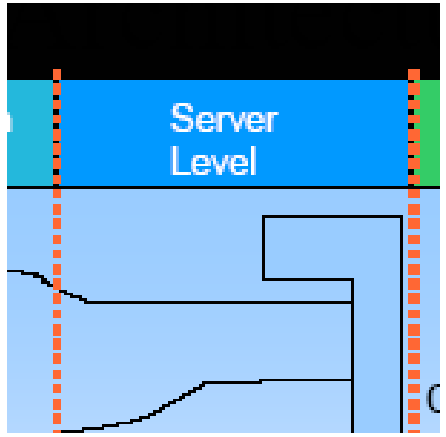
Sensor préfixe imposé …

Agent

- **L'agent se charge de**
  - **L'Installation (instanciation) du MBean**
  - **L'Enregistrement auprès du serveur de MBeans**
    - `MBeanServer`
- **Un nom unique lui est attribué**
  - **en général par l'utilisateur,**
  - **selon une convention de nommage**
    - **Apparenté Properties exemple :** `SensorAgent:name=Sensor1`

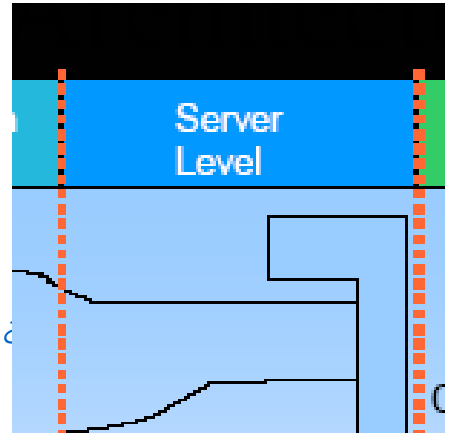
      » name la clé, Sensor1 la valeur
- **Est installé sur la même JVM**
  - ***D'autres peuvent le faire : un autre MBean, le serveur, de l'extérieur …***

Agent : SensorAgent




```
public class SensorAgent{
  private MBeanServer mbs;

  public SensorAgent(){
    try{

      mbs = ManagementFactory.getPlatformMBeanServer();//*
      ObjectName name = new ObjectName("SensorAgent:name=Sensor1");
      Sensor mbean = new Sensor(2000);    // création du mbean
      mbs.registerMBean(mbean, name);     // enregistrement
    }catch(Exception e){
      e.printStackTrace();
    }
  }

  public static void main(String[] args)throws Exception{
     … new SensorAgent(); … Thread.sleep(Long.MAX_VALUE);

  * usage ici du patron fabrique …
```

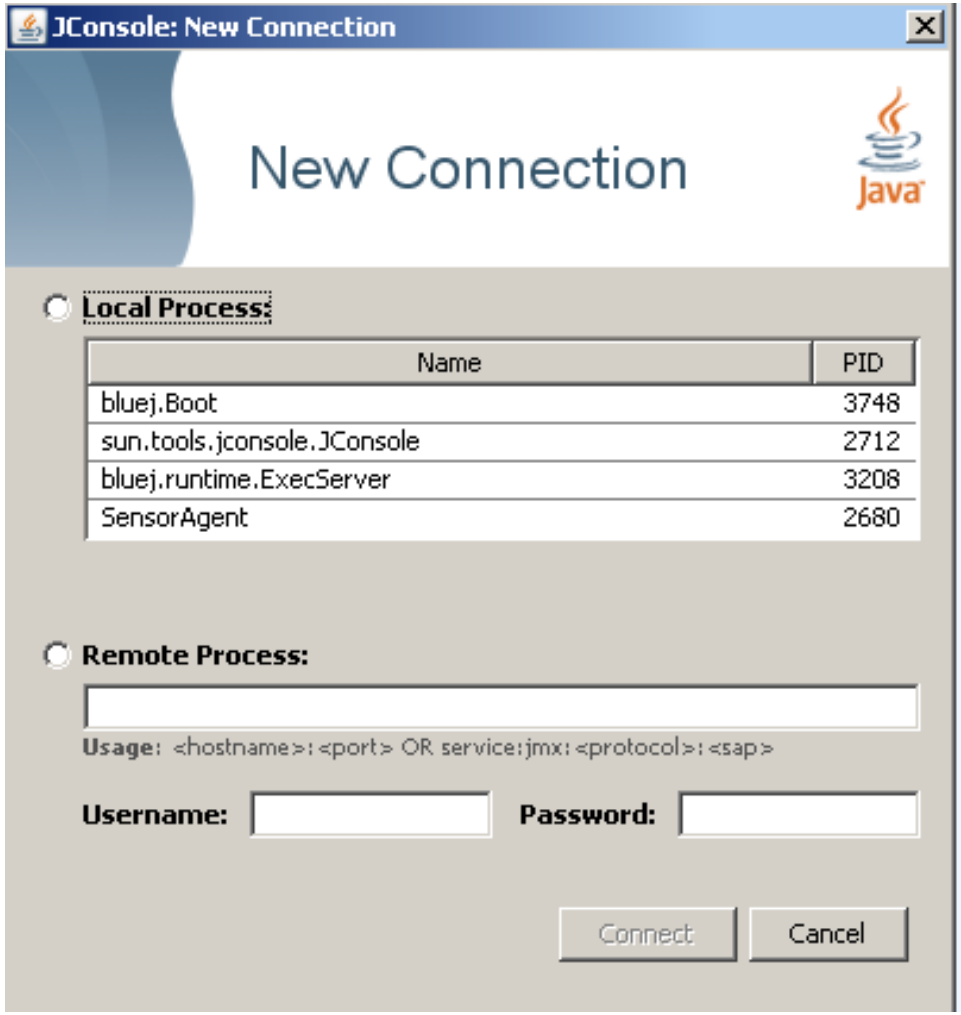
Client : SensorClient




```
public class SensorClient{

  // même JVM
  // SensorAgent a = new SensorAgent(); préala

 MBeanServer mbs =  ……
 // recherche du MBean
 ObjectName name = new ObjectName("SensorAgent:name=Sensor1");

 // accès à l'attribut, getValue()
 System.out.println(mbs.getAttribute(name, "Value"));

…
```

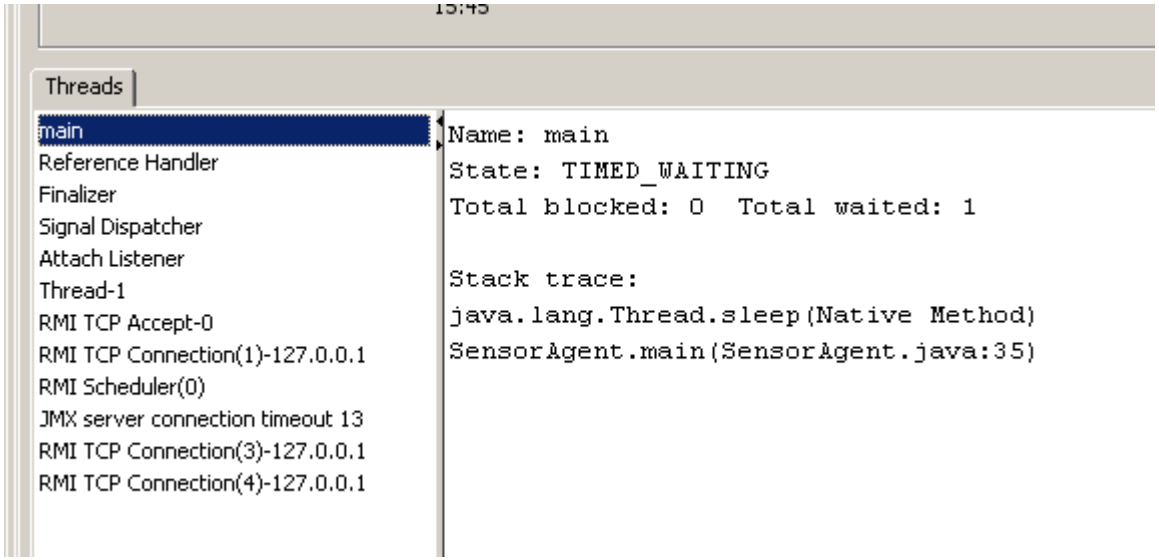
Distribué : jconsole

- **Accès à l'une des JVM,**
  - **munie par défaut de son MBeanServer**



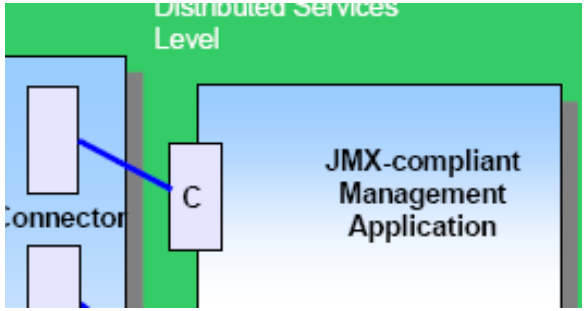
Distribué : jconsole

- **Onglet Threads**
  - **SensorAgent est bien endormi…**



Distribué : jconsole

- **name=Sensor1**
  - **Accès aux attributs**



Java Monitoring & Management Console

Connection Window Help

pid: 3188 SensorAgent

Overview | Memory | Threads | Classes | VM Summary | MBeans

- JMImplementation
- SensorAgent
  - Sensor1
    - Attributes
      - Value
    - Operations
      - reset
- com.sun.management
- java.lang
- java.util.logging

Attribute value

Name	
Value	12

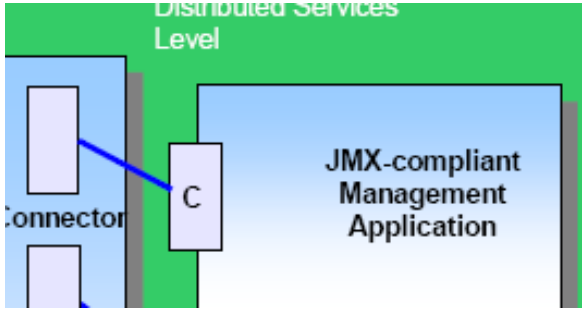
Refresh

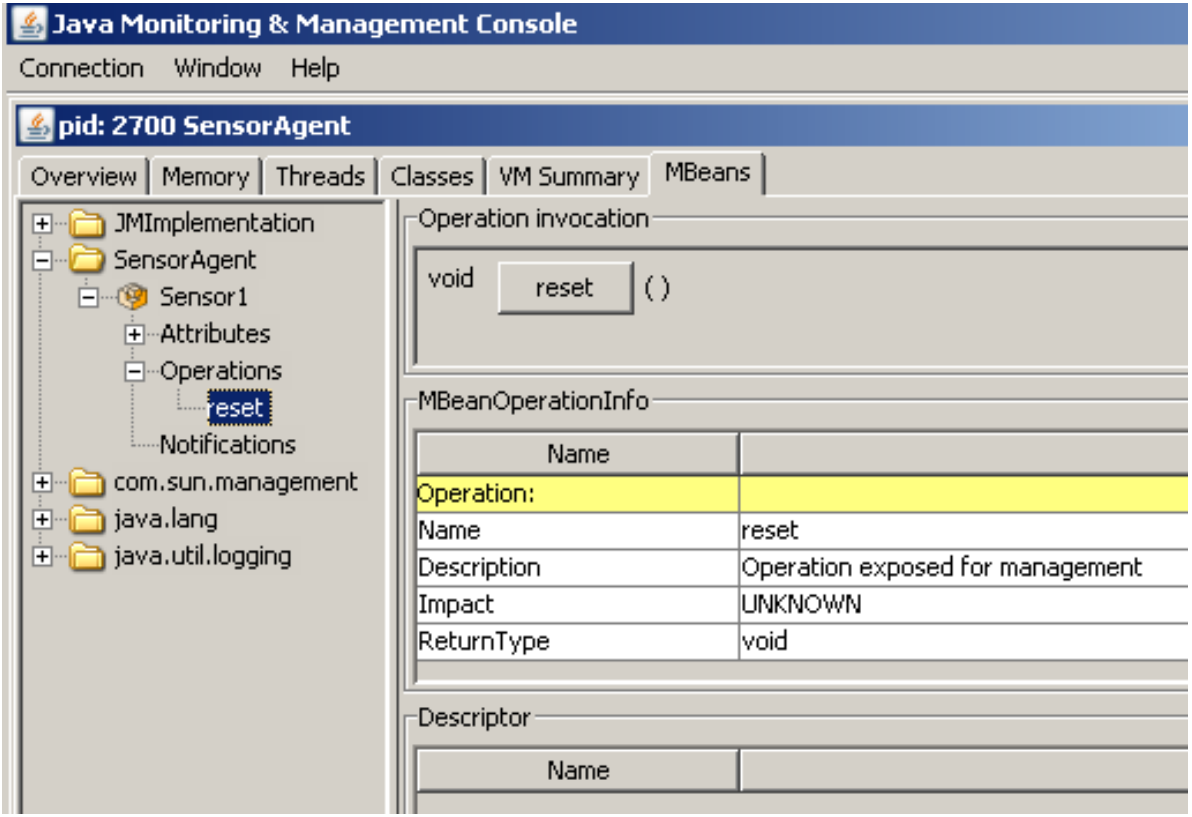
MBeanAttributeInfo

Name	Value
Attribute:	
Name	Value
Description	Attribute exposed for management
Readable	true
Writable	true
Is	false
Type	int

Distribué : jconsole

- **name=Sensor1**
  - **Opération reset()**



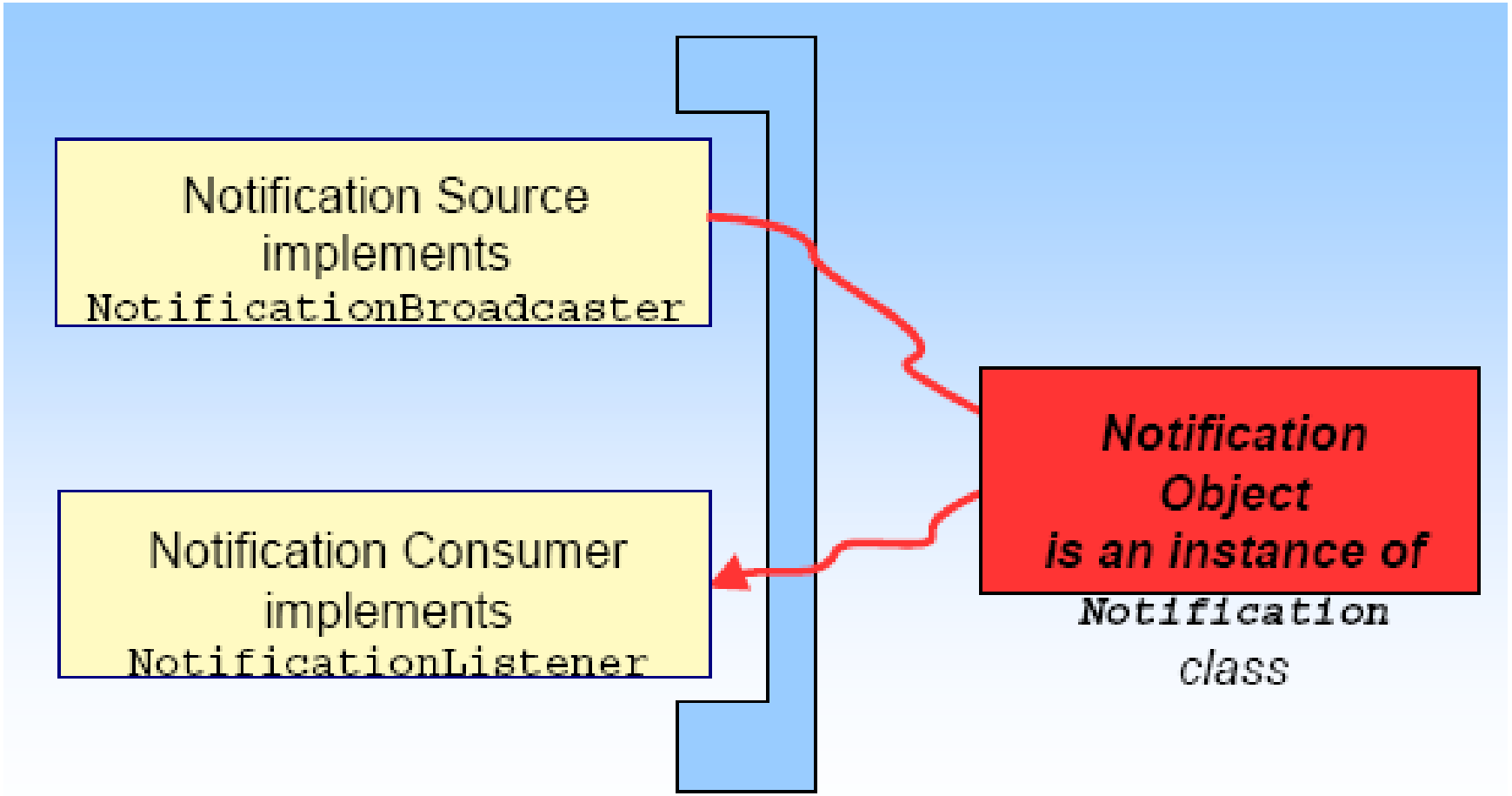


Résumé

- Instrumentation MBean
- Serveur Un Agent
- Distribué jconsole

- Réveil, alarmes, pannes …
  - Ou bien Notifications, évènements asynchrones ?

Réveil & notifications



- **À la suite d'un changement d'état**
  - **Patron publish/subscribe, mode pull**

Instrumentation & Notifications

```
public class Sensor
     extends NotificationBroadcasterSupport
     implements SensorMBean{


  public synchronized void setValue(int value){
    this.value = value;
    this.sequenceNumber++;
    sendNotification(
       new Notification(
             "setValue",                    // un nom
             this,
             sequenceNumber,                // un numéro
             System.currentTimeMillis(),    // une estampille
             Integer.toString(value)));     // un message
  }
```

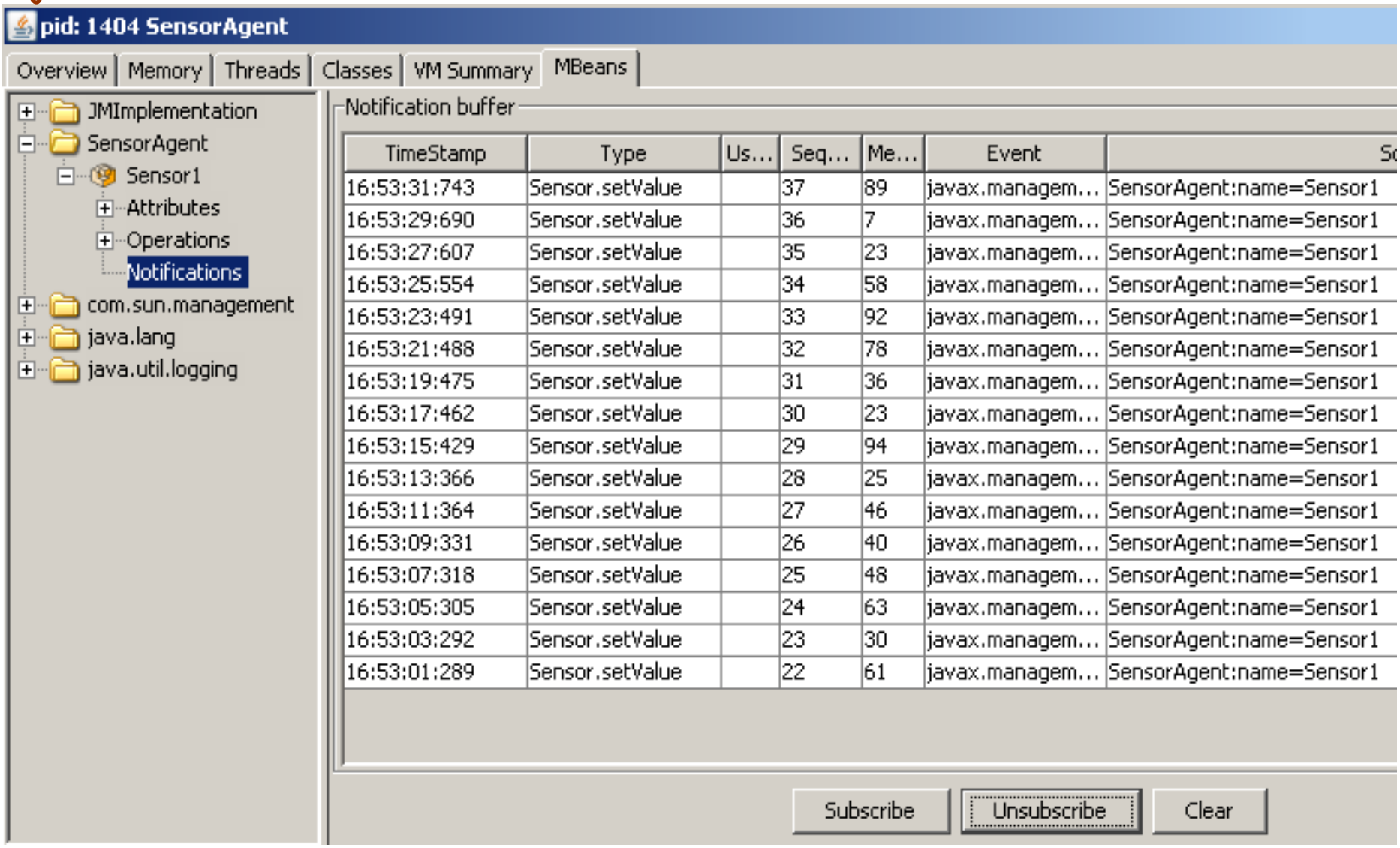
Agent & notifications

- L'agent est un ici observateur de « son » MBean

```
public class SensorAgent implements NotificationListener{
  private MBeanServer mbs;

  public SensorAgent(){
    try{
      …
      mbean.addNotificationListener(this,null,null);
    }catch(Exception e){
      e.printStackTrace();
    }
  }

  public void handleNotification(
      Notification notification, Object handback){
    System.out.print(notification.getMessage());
    System.out.println(" number : " + notification.getSequenceNumber());
}
```

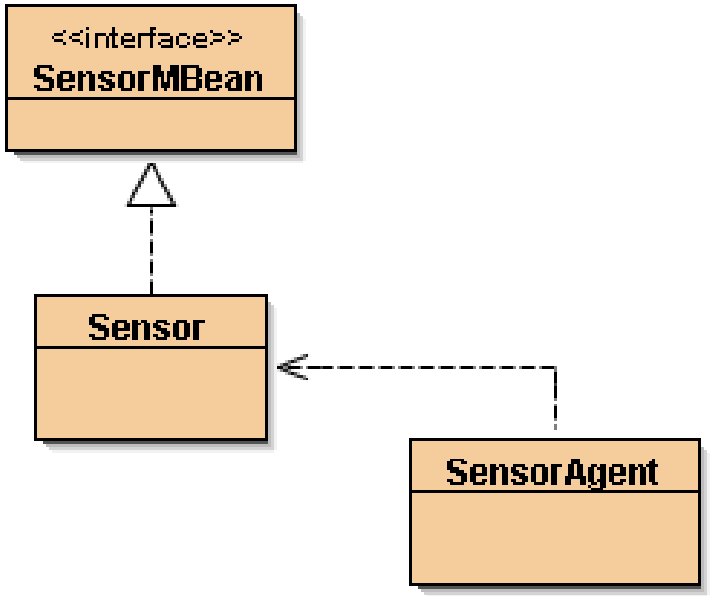
Client : jconsole & notifications

jconsole a souscrit



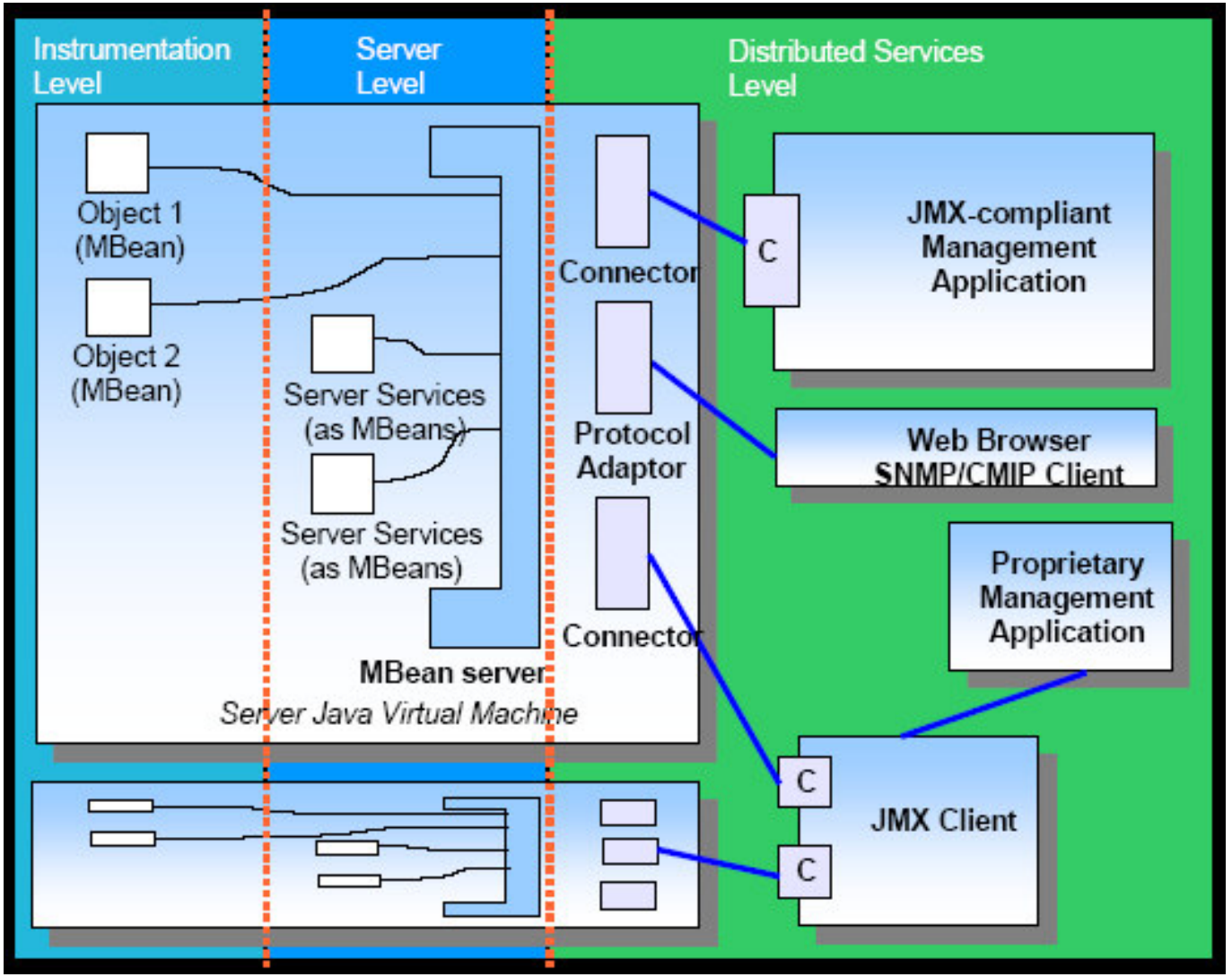
Petite conclusion

- **À la mode des Bean-Java**
  - Getter/setter, opérations
  - Standard MBean comme suffixe … *Instrumentation*
  - Enregistrement de ce MBean … *Agent*
  - Supervision *jconsole*



- **Simple !**
- **À suivre…**

Sommaire



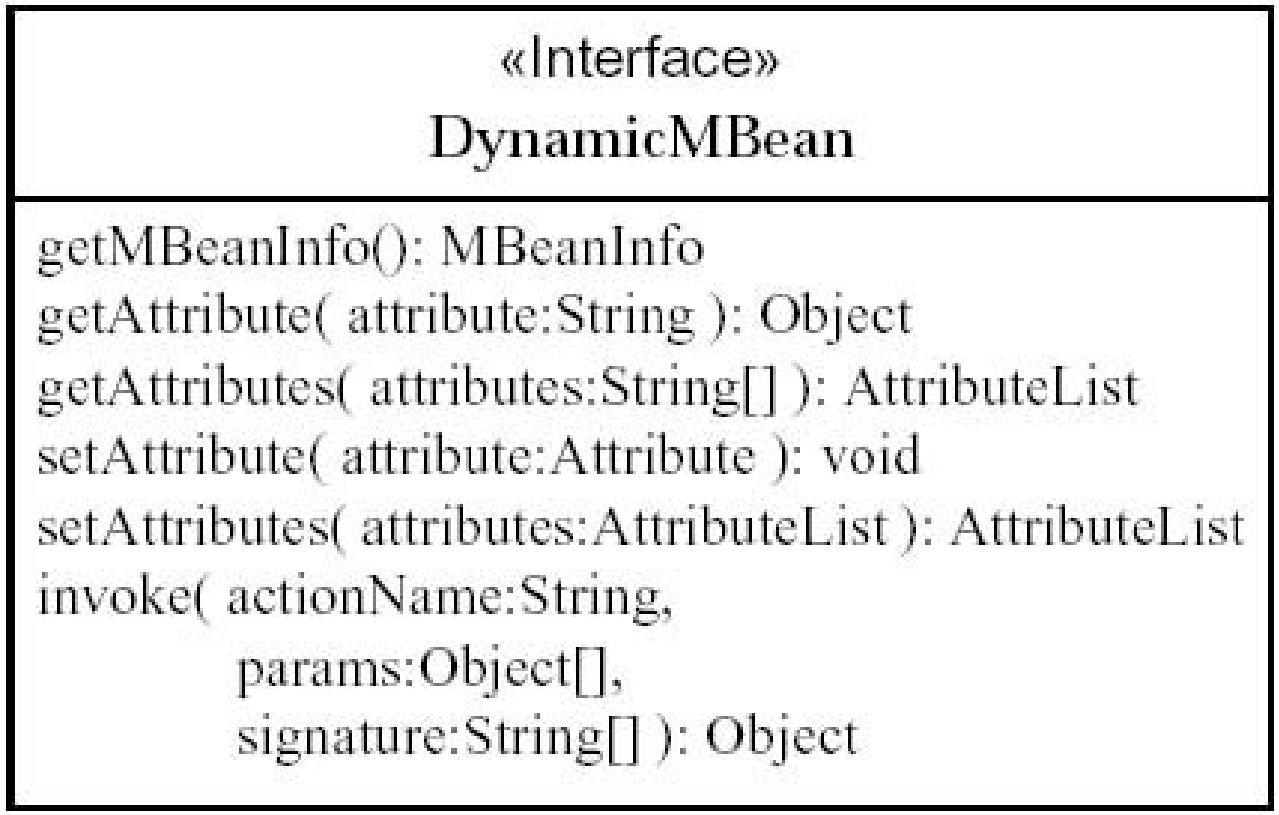
Maintenant d'un peu plus près

Instrumentation MBeans

5 types

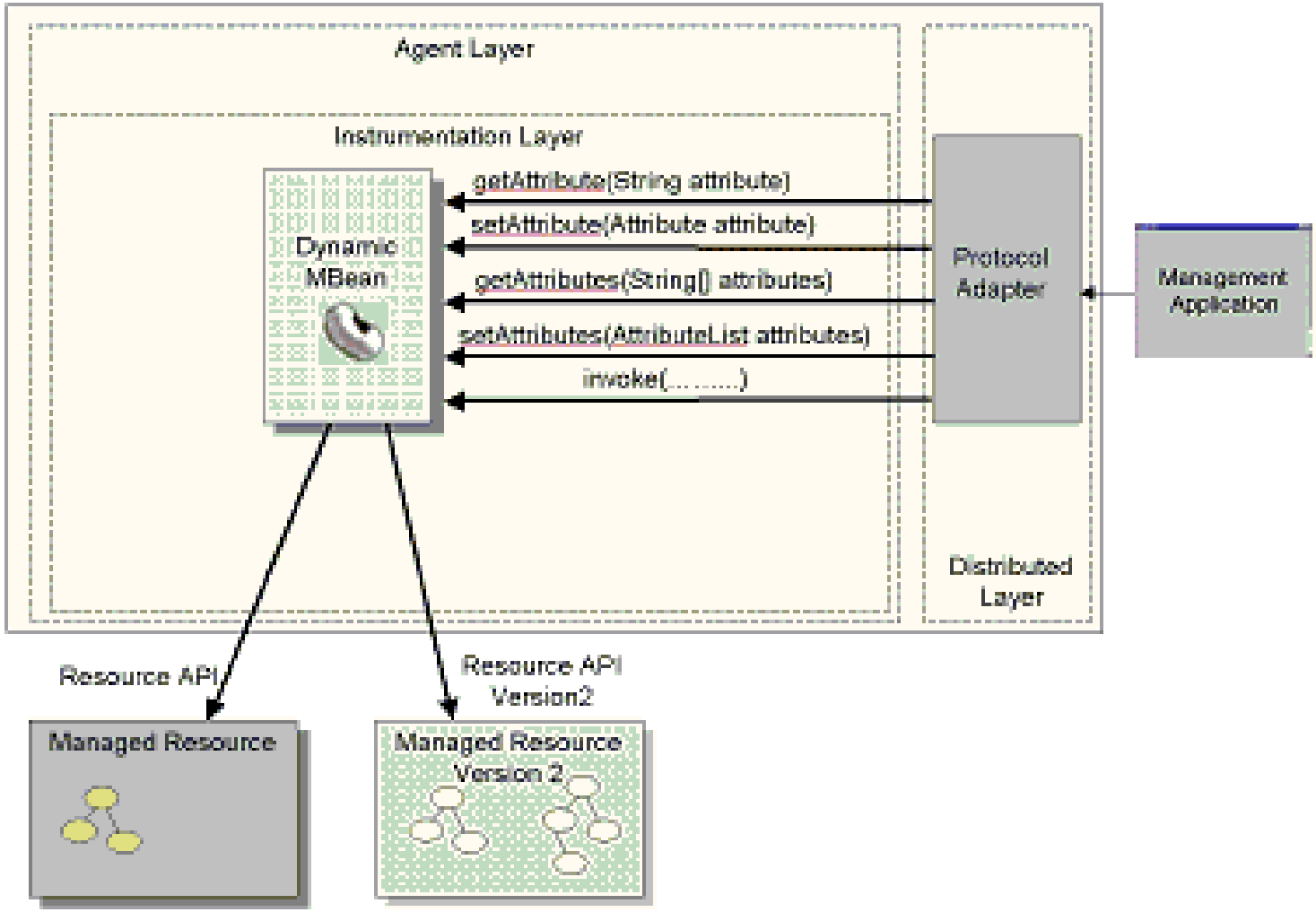
- **Standard : engendré depuis une interface xxxMBean**
  - **Voir l'exemple de présentation**
- **Dynamic : n'importe quel objet,**
  - **fonctionnalités découvertes à l'exécution**
- **Model : configurable, une template à instancier**
- **Open : limité à un ensemble de type Java**
  - *Inhibe les téléchargements de code*
- **MXBean 1.6 : accès aux ressources de la JVM**

DynamicMBean



- **Par introspection …**
  - **c'est un mandataire String/JVM**

Dynamic



- **Accès de l'extérieur …**
  - **possible avec des noms d'attributs ou d'opérations**
  - **Il sera possible de les télécharger …**

Exemple du capteur

```
public class SensorDynamic
    extends NotificationBroadcasterSupport
    implements DynamicMBean {

    public SensorDynamic() {
      buildDynamicMBeanInfo();
    }

    public Object getAttribute(String attributeName)
        throws AttributeNotFoundException,
               MBeanException,
               ReflectionException {
        if (attributeName.equals("Value")) {
            return getValue();
        }
…
```

Voir
http://java.sun.com/j2se/1.5.0/docs/guide/jmx/examples.html

DynamicMBean

- **Proposer une « signature » du composant**
  - **getMBeanInfo() retourne cette signature**
  - **Les méta-données MBeanInfo décrivent les attributs, opérations et notifications**
- **Utilisation ad'hoc**
  - **Un adaptateur d'objets Java existants afin de les rendre « compatibles MBean »**

Instrumentation MBeans

5 types

- **Standard : engendré depuis une interface xxxMBean**
- **Dynamic : n'importe quel objet,**
  - **fonctionalités découvertes à l'exécution**
- **Model : configurable, une template à instancier**
- **Open : limité à un ensemble de type Java**
  - *Inhibe les téléchargements de code*
- **MXBean 1.6 : accès aux ressources de la JVM**

Instrumentation MBeans

5 types

- **Standard : engendré depuis une interface xxxMBean**
- **Dynamic : n'importe quel objet,**
  - **fonctionalités découvertes à l'exécution**
- **Model : configurable, une template à instancier**
- **Open : limité à un ensemble de type Java**
  - *Inhibe les téléchargements de code*
- **MXBean 1.6 : accès aux ressources de la JVM**

Instrumentation MBeans

5 types

- **Standard : engendré depuis une interface xxxMBean**
- **Dynamic : n'importe quel objet,**
  - **fonctionalités découvertes à l'exécution**
- **Model : configurable, une template à instancier**
- **Open : limité à un ensemble de type Java**
- **MXBean : accès aux ressources de la JVM**

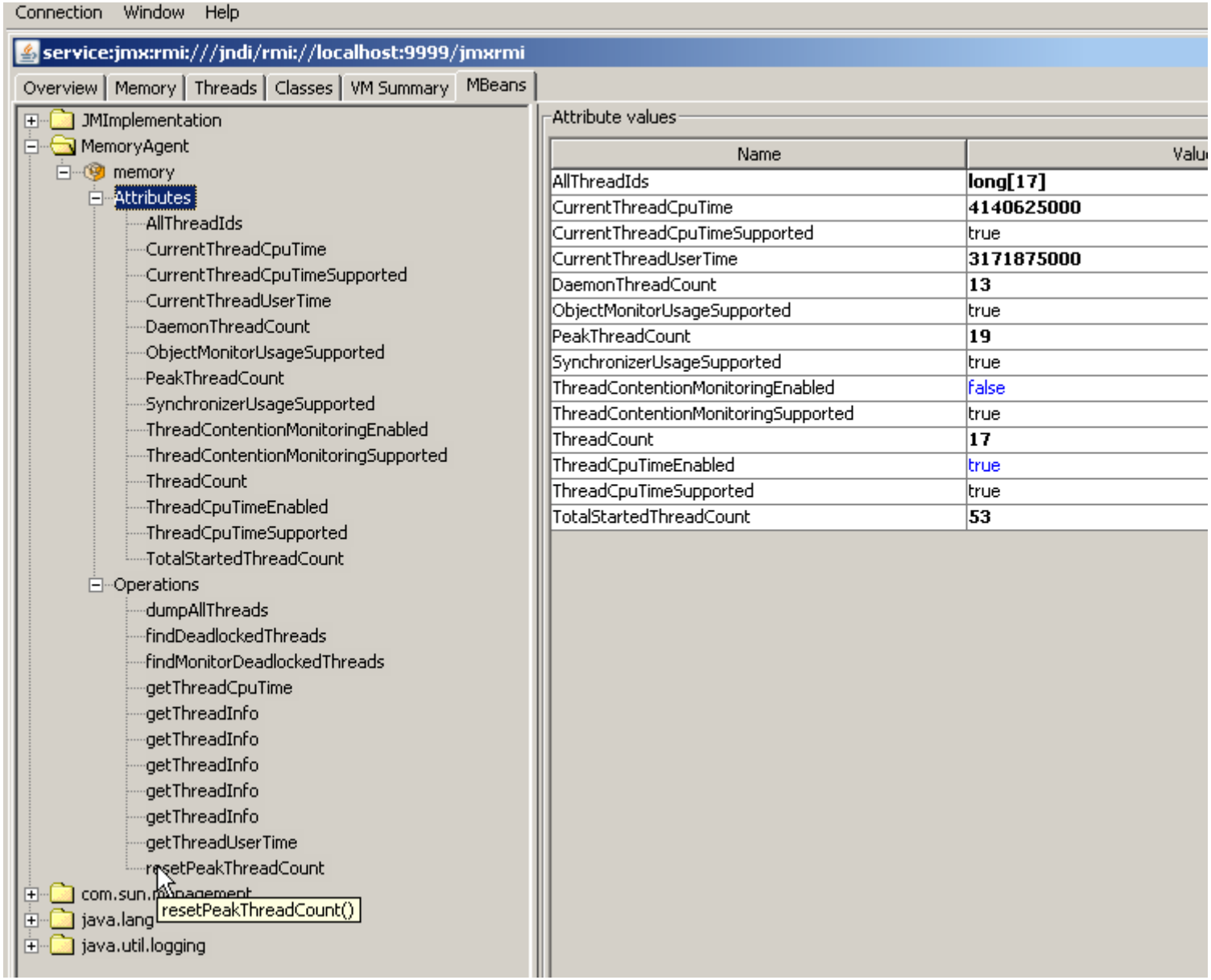
MXBean

• Compilation	CompilationMXBean
• Garbage collection system	GarbageCollectorMXBean
• Memory	MemoryMXBean
• Memory managers	MemoryManagerMXBean
• Threading	ThreadMXBean
• Operating system	OperatingSystemMXBean
• Runtime system	RuntimeMXBean
• Class loading system	ClassLoadingMXBean
• Memory resources	MemoryPoolMXBean

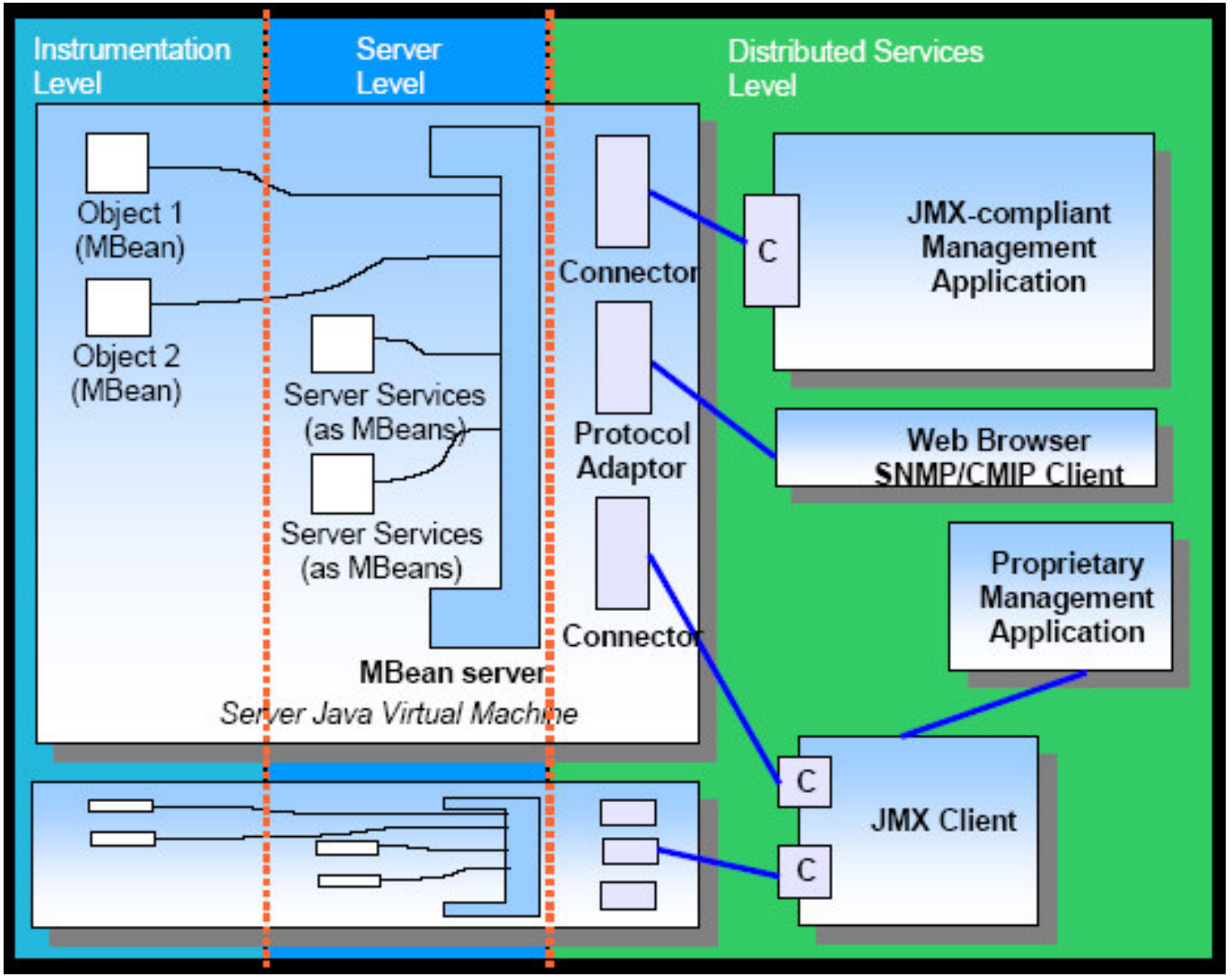
Un exemple : ThreadAgent, ThreadMXBean

```
public class ThreadAgent{
  private MBeanServer mbs;

   public ThreadAgent(){
    try{

      ThreadMXBean mbean = ManagementFactory.getThreadMXBean();
      mbs = ManagementFactory.getPlatformMBeanServer();
      ObjectName name = new ObjectName("ThreadAgent:name=thread1");
      mbs.registerMBean(mbean, name);
   }catch(Exception e){
      e.printStackTrace();
    }
  }
  public static void main(String[] args)throws Exception{
     … new ThreadAgent(); … Thread.sleep(Long.MAX_VALUE);
```

Distribué : jconsole



Sommaire



Agents ..

Agents

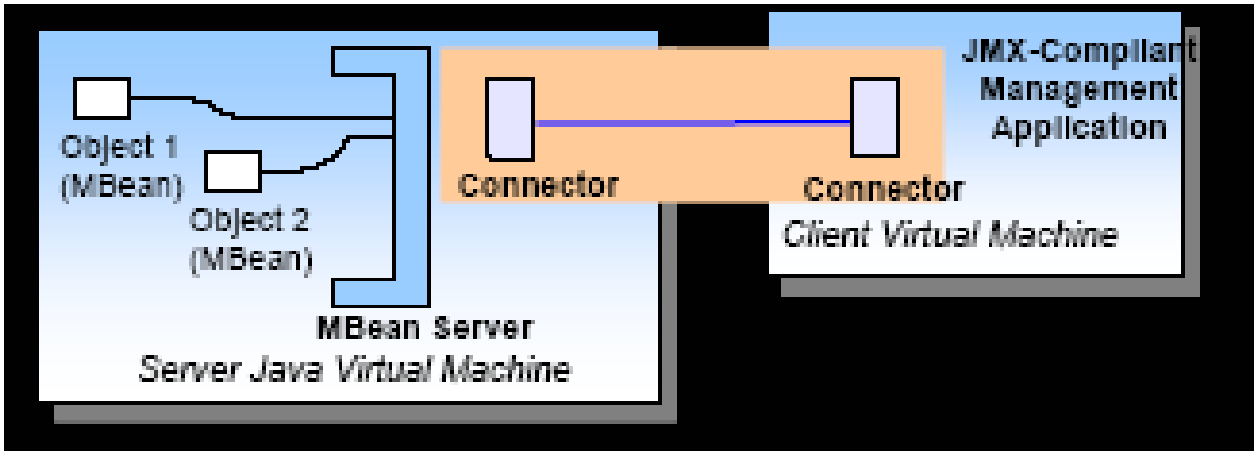
- Adpatateurs et connecteurs
- MBean server
- Interrogation et listage
- Chargement dynamique
- Agent services

Distribué, Connector, Adapter

- Un *connector*
  - Est un MBean,
  - Est enregistré auprès du “MBean server”,
  - Communique avec une machine (paire)
  - Exemple
    - Un rmi connecteur

- Un *adapter*
  - Est un MBean ,
  - Est enregistré auprès du “MBean server”,
  - Ecoute sur un port et respecte un certain protocole.
  - Exemple
    - Un *adapter* HTML accepte des requêtes au protocole HTTP
    - Un client type est un navigateur

Les connecteurs



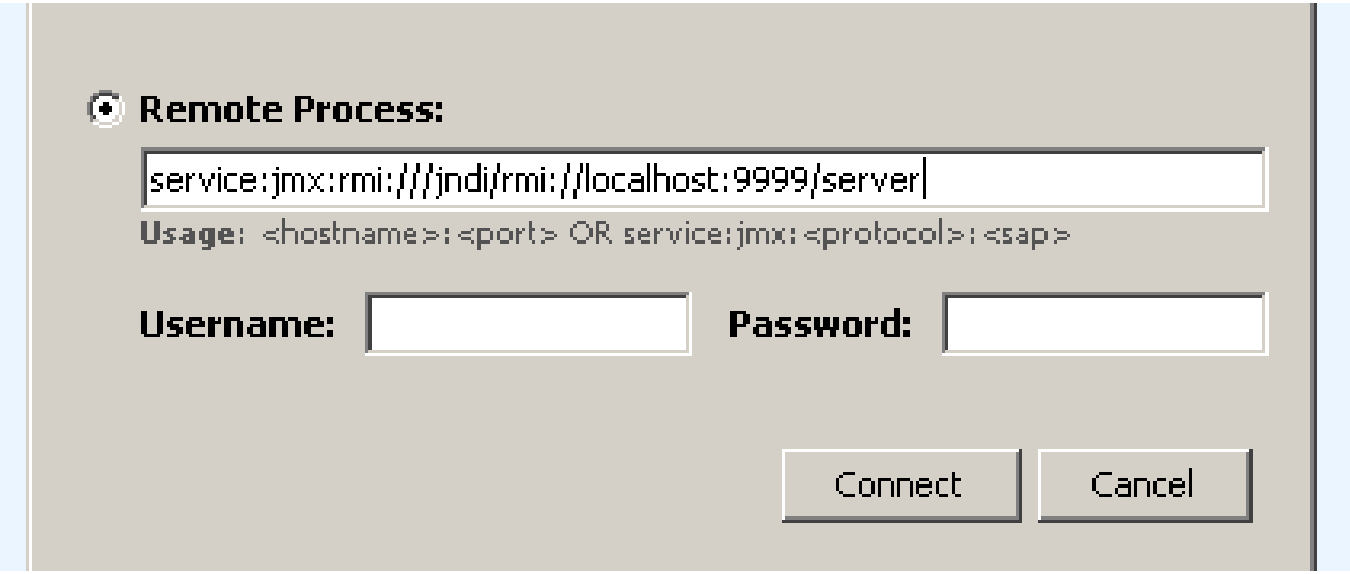
- **RMI**
- **TCP dédié**

Rmi connecteur, SensorAgent

```
try {
  MBeanServer mbs ….
  name = new ObjectName("SensorAgent:name=Sensor2");
  mbs.registerMBean(sensorBean, name);

  // Creation et démarrage du connecteur
JMXServiceURL url = new
  JMXServiceURL("service:jmx:rmi:///jndi/rmi://localhost:9999/server");

JMXConnectorServer cs =
  JMXConnectorServerFactory.newJMXConnectorServer(url, null, mbs);

  cs.start();

} catch(Exception e) {
}
```

Rmi + jconsole

- **start rmiregistry 9999**
  - **Côté MBeanServer**



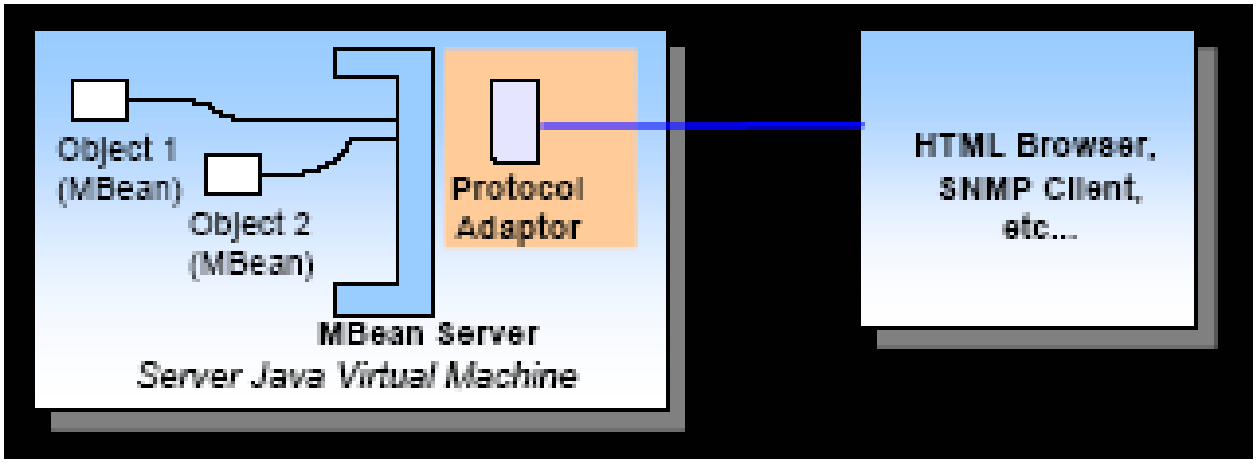
RmiClient

```
public class RMIClient{


   public static void main(String[] args) throws
   Exception{
// à la recherche du connecteur, via l'annuaire
      JMXServiceURL url = new
   JMXServiceURL("service:jmx:rmi:///jndi/rmi://localhost
   :9999/server");
      JMXConnector cs = JMXConnectorFactory.connect(url);
      MBeanServerConnection mbs =
   cs.getMBeanServerConnection();
     ObjectName name = new
   ObjectName("SensorAgent:name=Sensor2");


   System.out.println(" value : " + mbs.getAttribute(name,
   "Value"));
      }
}
```

Notification, déjà vue

```
public void addNotificationListener(ObjectName name,
  NotificationListener listener, NotificationFilter filter,
  Object handback)

public void addNotificationListener(ObjectName name, ObjectName
  listener, NotificationFilter filter, Object handback)



public void removeNotificationListener(ObjectName name,
  NotificationListener listener)

public void removeNotificationListener(ObjectName name,
  ObjectName listener)
```

Les adaptateurs



- **Navigateur → html adaptateur**

Html adaptateur, SensorAgent

```
try {
  MBeanServer mbs ….
  name = new ObjectName("SensorAgent:name=Sensor2");
  mbs.registerMBean(sensorBean, name);

  // Creation et démarrage de l'adaptateur
  HtmlAdaptorServer adapter = new HtmlAdaptorServer();//*
  adapter.setPort(8088);
  name = new
  ObjectName("HtmlAdaptorServer:name=html,port=8088");
  mbs.registerMBean(adapter, name);
  adapter.start();


* import com.sun.jdmk.comm.*; // 5.0.1
```

Le Client est un navigateur

Agent View

Filter by object name: *:*

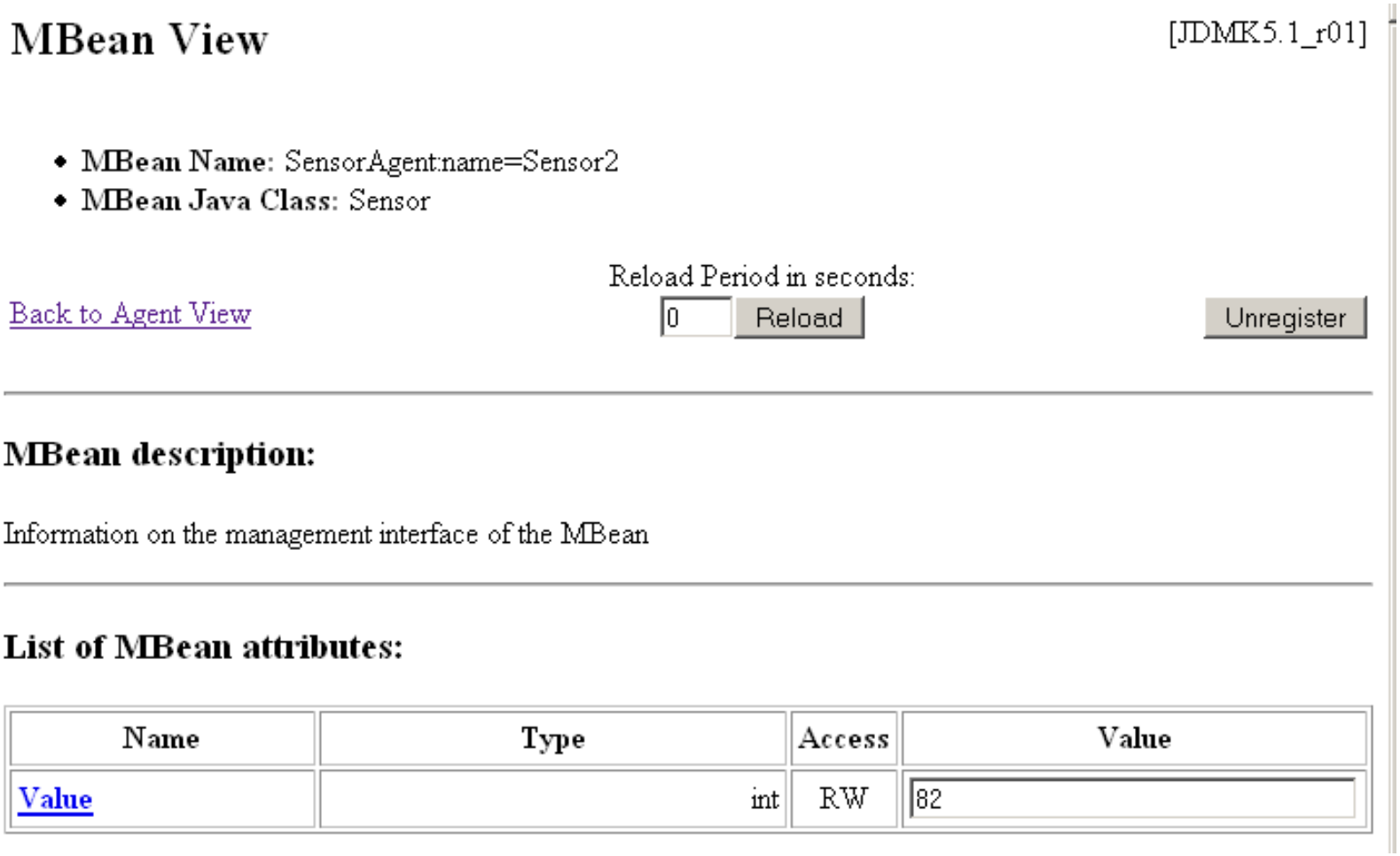
This agent is registered on the domain ***DefaultDomain***.
This page contains **19** MBean(s).

List of registered MBeans by domain:

- **HtmlAdaptorServer**
  - name=html,port=8088
- **JMImplementation**
  - type=MBeanServerDelegate
- **SensorAgent**
  - name=Sensor2
- **com.sun.management**
  - type=HotSpotDiagnostic
- **java.lang**
  - type=ClassLoading
  - type=Compilation
  - type=GarbageCollector,name=Copy
  - type=GarbageCollector,name=MarkSweepCompact

http://localhost:8088/ViewObjectRes//SensorAgent:name=Sensor2

Client, navigateur



MBean View

[JDMK5.1_r01]

- MBean Name: SensorAgent:name=Sensor2
- MBean Java Class: Sensor

Back to Agent View

Reload Period in seconds: 0 Reload

Unregister

MBean description:

Information on the management interface of the MBean

List of MBean attributes:

Name	Type	Access	Value
Value	int	RW	82

- Sans oublier, jdmkrt.jar dans le classpath
  - Bluej voir le répertoire lib/userlib/
  - java -cp .;../JMX/jdmkrt.jar ThreadAgent

A la recherche du MBean ?

- **MBean en connaisssant son nom : ok**
- **Tous les MBean d'un certain type ?**
- **Tous les MBean dont l'attribut « Counter » == 10**
- **Tous les MBean dont l'attribut « Counter » >= 10 et dont l'estampille < X**
- **« QL » pour les Mbean ? MQL ?**
  - `QueryExp`
  - **Expression booléenne**

QueryExp, comme un composite…

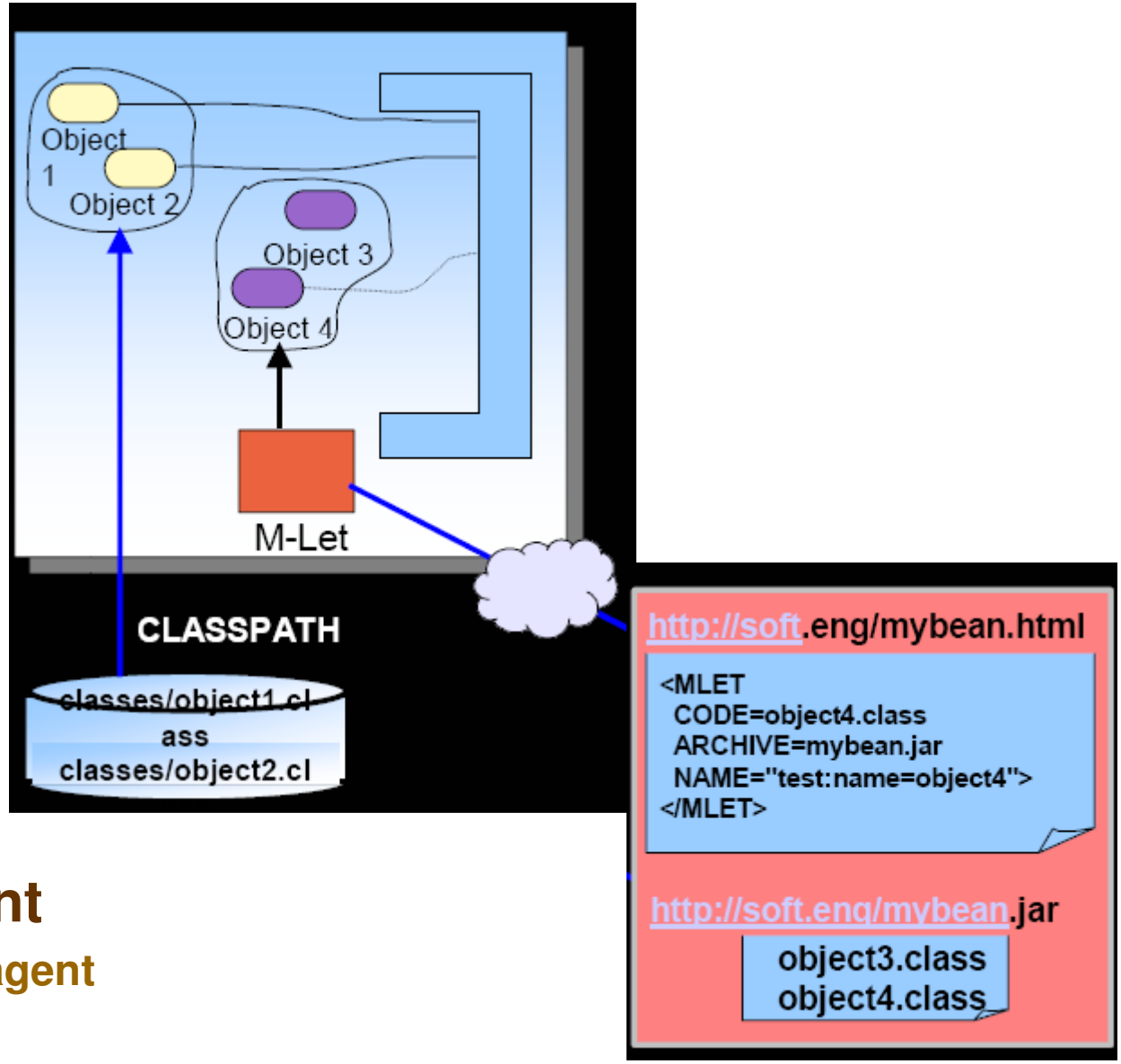
```
QueryExp exp = Query.gt(Query.attr("count"),
                        Query.value(10))


QueryExp prob1 = Query.eq(Query.attr("inkLevel"),
                          Query.value("LOW"));

QueryExp prob2 = Query.lt(Query.attr("paperCount"),
                          Query.value(50));

QueryExp exp = Query.or(prob1, prob2);
```

- *"(inkLevel = LOW) or (paperCount < 50)"*

Chargement dynamique de MBean

- **Agent M-Let**
- **Description XML**

M-Let



- **Comment**
  - **M-Let agent**

Chargement dynamique de MBean

- **M-Let**

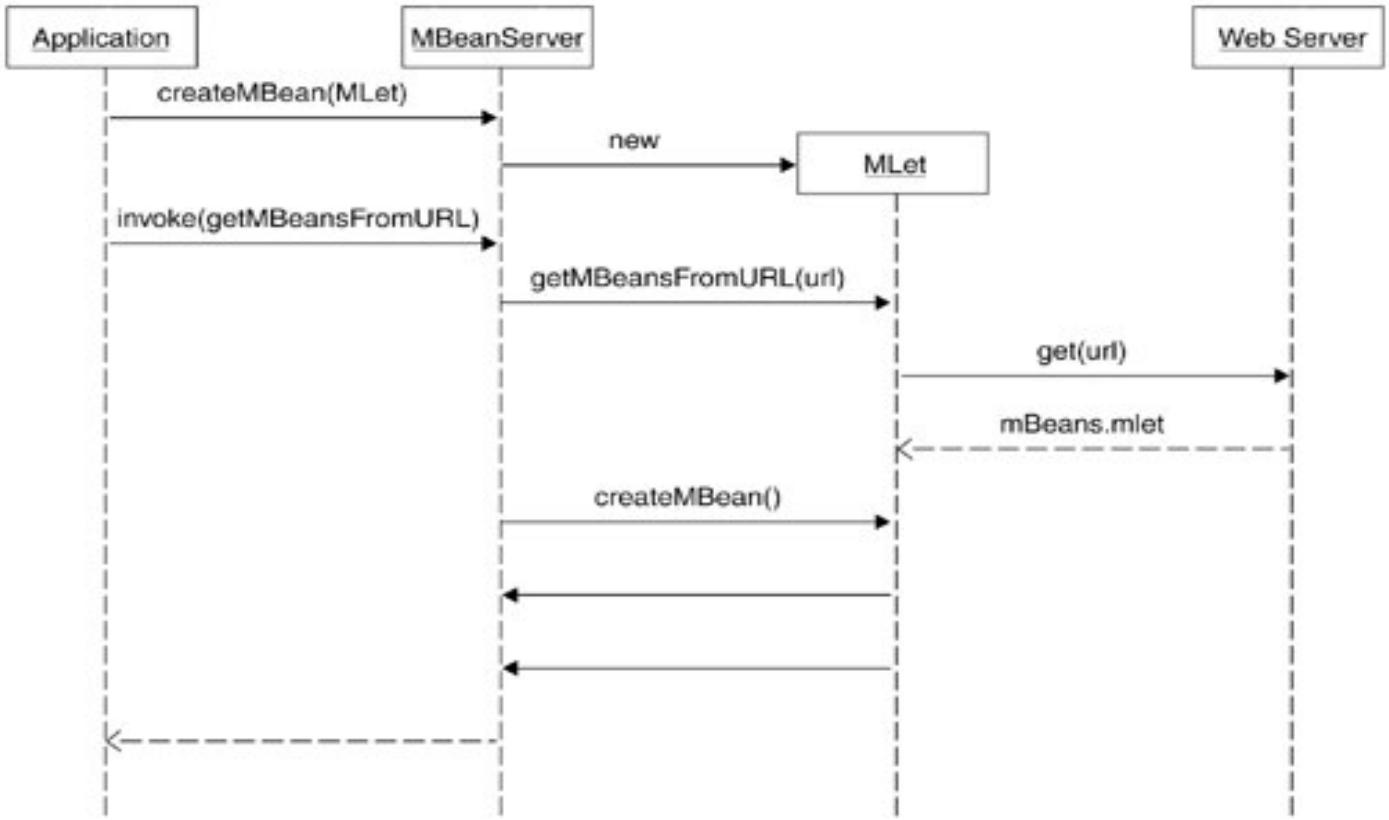
  – « URLClassLoader »

  Associé à un descripteur en XML

```
<MLET
      CODE=Sensor
      ARCHIVE=sensor.jar
      CODEBASE=http://jfod.cnam.fr/NSY102/jmx/
      NAME=Sensor:name=Sensor1>
</MLET>
```

  - Descripteur XML, nommé sensor.mlet, à cette URL http://jfod.cnam.fr/NSY102/jmx/sensor.mlet

Mlet en séquence

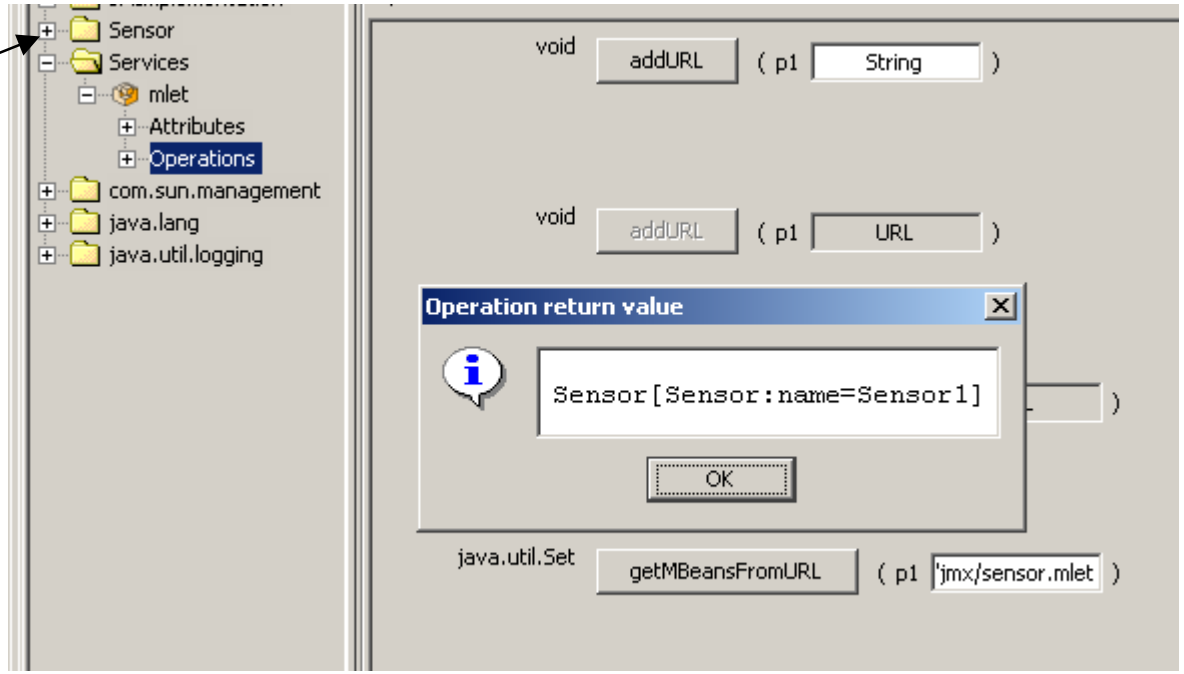


Agent M-let

```
public MLetAgent(){
    try{
      mbs = ManagementFactory.getPlatformMBeanServer();

      ObjectName name = new ObjectName("Services:name=mlet");
      mbs.createMBean("javax.management.loading.MLet", name);


JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://localhost:9999/server");
 JMXConnectorServer cs = JMXConnectorServerFactory.newJMXConnectorServer(url, null, mbs);
 cs.start();

    }catch(Exception e){
      e.printStackTrace();
    }
 }
```

Agent M-Let

Sensor
À la volée!



- **Téléchargement dynamique du MBean Sensor**
  - **Appel de getMBeansFromURL("http://jfod.cnam.fr/NSY102/jmx/sensor.mlet")**

MLetBean

```
public interface MLetMBean {

public Set getMBeansFromURL(String url) throws
  ServiceNotFoundException;

public Set getMBeansFromURL(URL url) throws
  ServiceNotFoundException;

public void addURL(URL url); public void addURL(String url)
  throws ServiceNotFoundException;

public URL[] getURLs();
public URL getResource(String name);
 public InputStream getResourceAsStream(String name);

public Enumeration getResources(String name) throws
  IOException; public String getLibraryDirectory();

public void setLibraryDirectory(String libdir); }
```

M-Let File

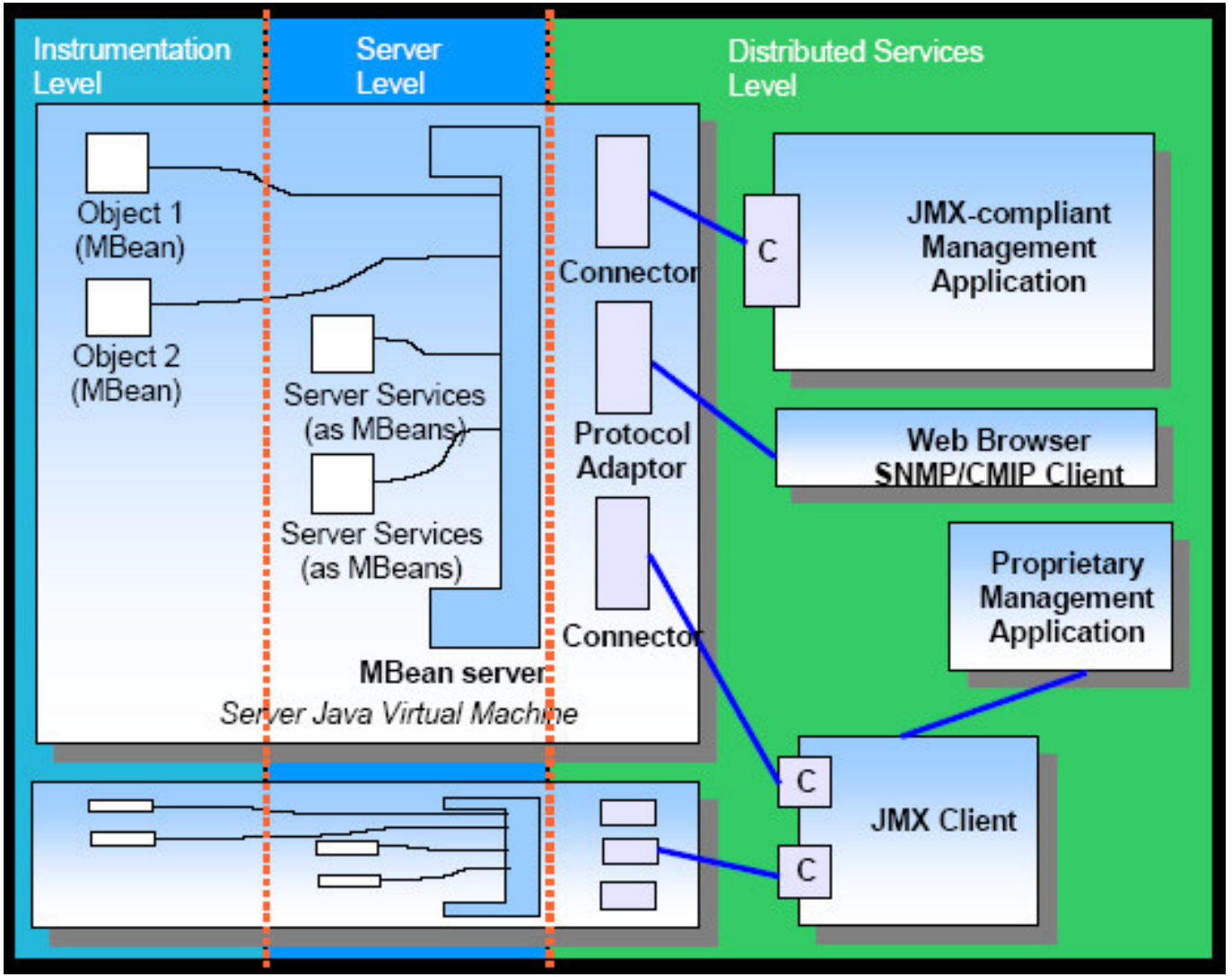
<MLET CODE="*className*" | OBJECT="*serializedObjectFileName*"
ARCHIVE="*classOrJarFileName*"
[CODEBASE="*relativePathToArchive*"]
[NAME="*mbeanObjectName*"]
[VERSION="*version*"] >
[<ARG TYPE="*type*" VALUE="*value*">]
</MLET>

- [<ARG TYPE="*type*" VALUE="*value*">]
  - Adaptés au constructeurs avec paramètres
  - Exemple **<ARG TYPE=int VALUE=3000>**

retrait dynamique de MBean

- `mbs = ManagementFactory.getPlatformMBeanServer();`
- **ObjectName name=…**
- **mbs.unregister(name)**

Sommaire


Instrumentation Level
Server Level
Distributed Services Level
Object 1 (MBean)
Object 2 (MBean)
Server Services (as MBeans)
Server Services (as MBeans)
MBean server
Server Java Virtual Machine
Connector
Protocol Adaptor
Connector
C
JMX-compliant Management Application
Web Browser SNMP/CMIP Client
Proprietary Management Application
C
C
JMX Client

Interrogations au sujet des MBeans

- **MBeanInfo meta-data:**

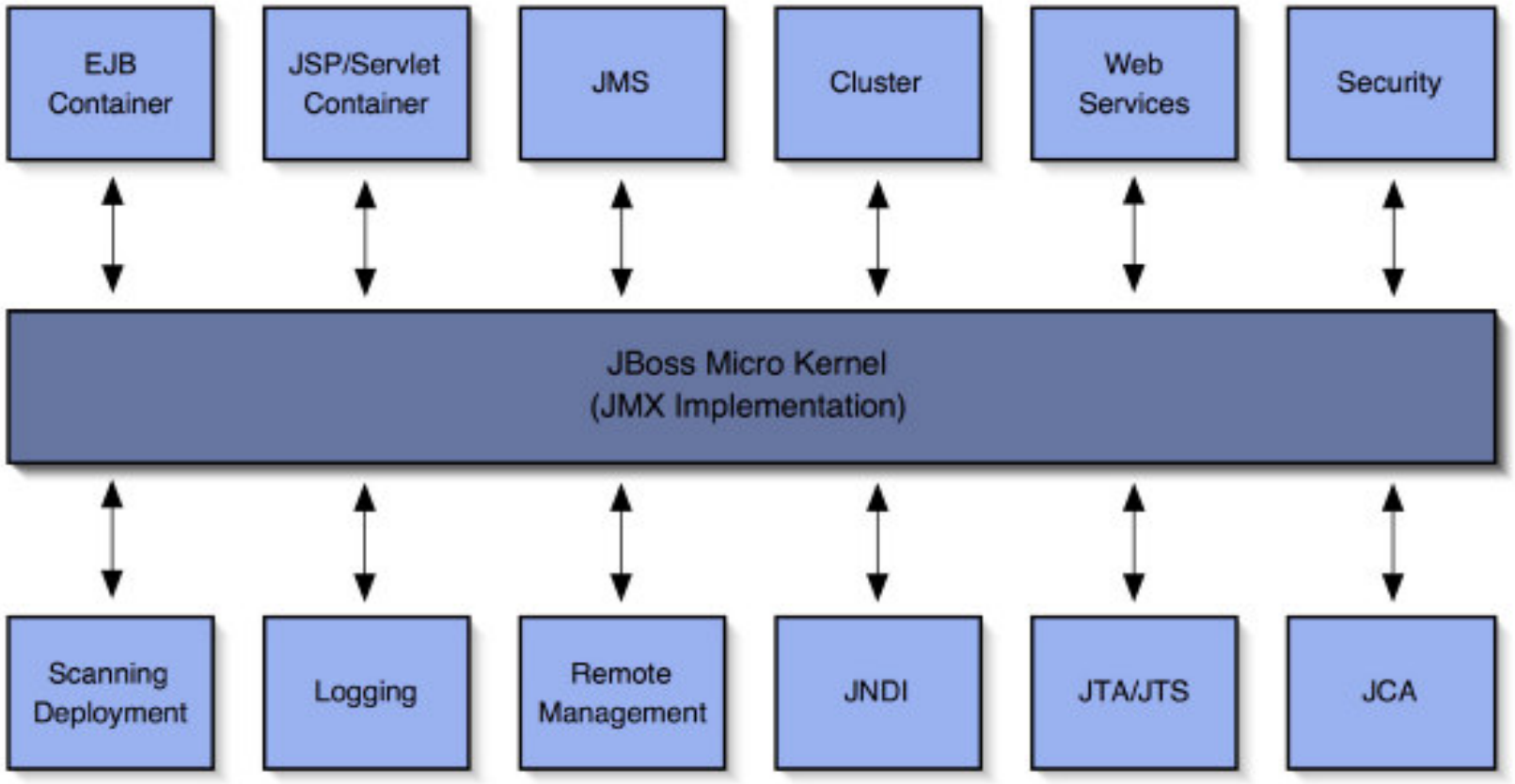
  `mbs.getMBeanInfo(objName);`

  – **Dynamic comme Standard MBeans**

- **And you can also run a type-check:**

  `mbs.isInstanceOf(objName, className)`

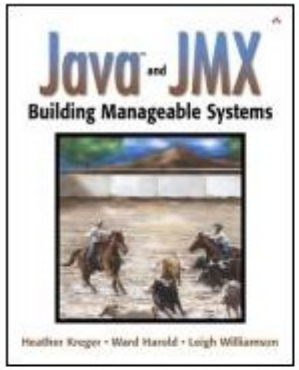
Exemplee : JMX-JBoss



- **http://www.openknowledge.de/pdf/jax2003/JMX_2003_05_10.pdf**

Conclusion

MBean Cycle de vie



- Table of Contents
- Index

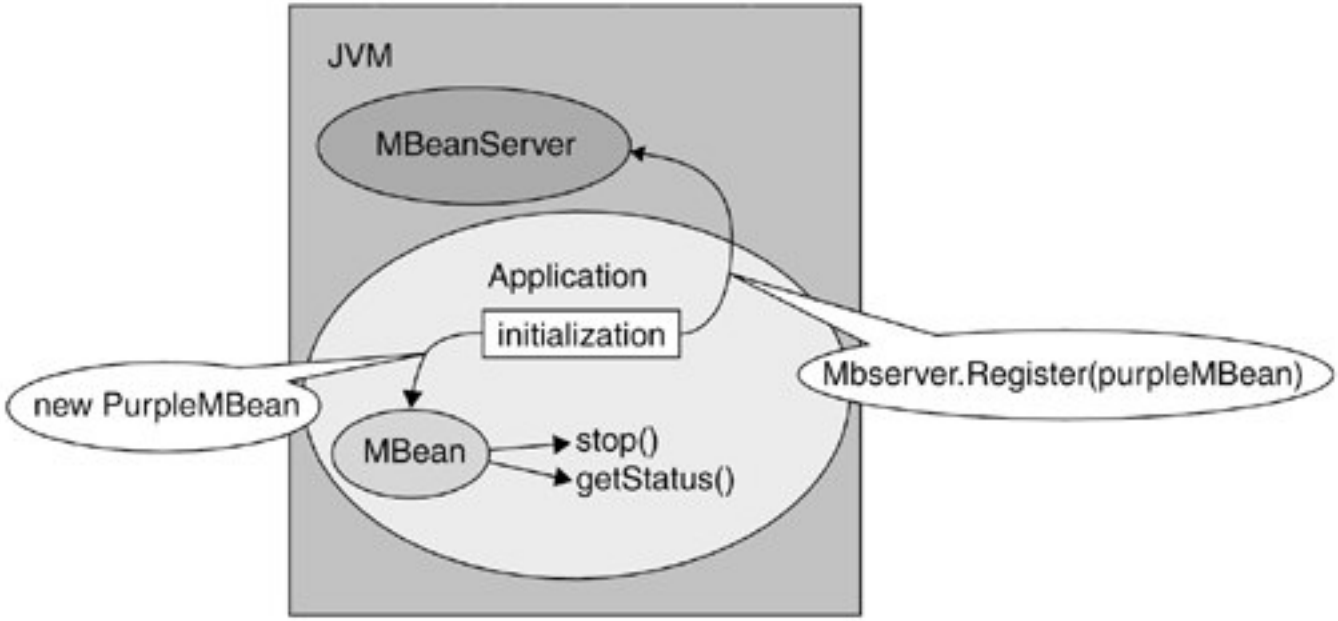
Java™ and JMX: Building Manageable Systems

By Heather Kreger, Ward Harold, Leigh Williamson

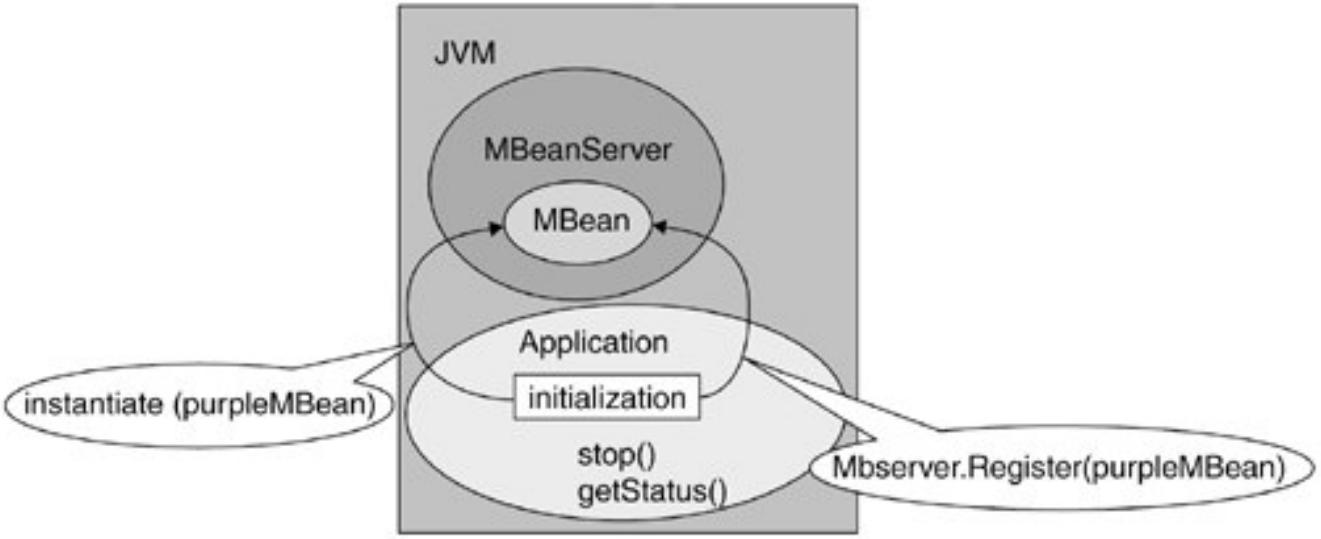
Publisher	: Addison Wesley
Pub Date	: December 30, 2002
ISBN	: 0-672-32408-3
Pages	: 592

- **Extrait de ce livre,**
  - **chapitre 9 : Designing with JMX**
    - **Paragraphe 3 MBean Registration and Lifecycle**

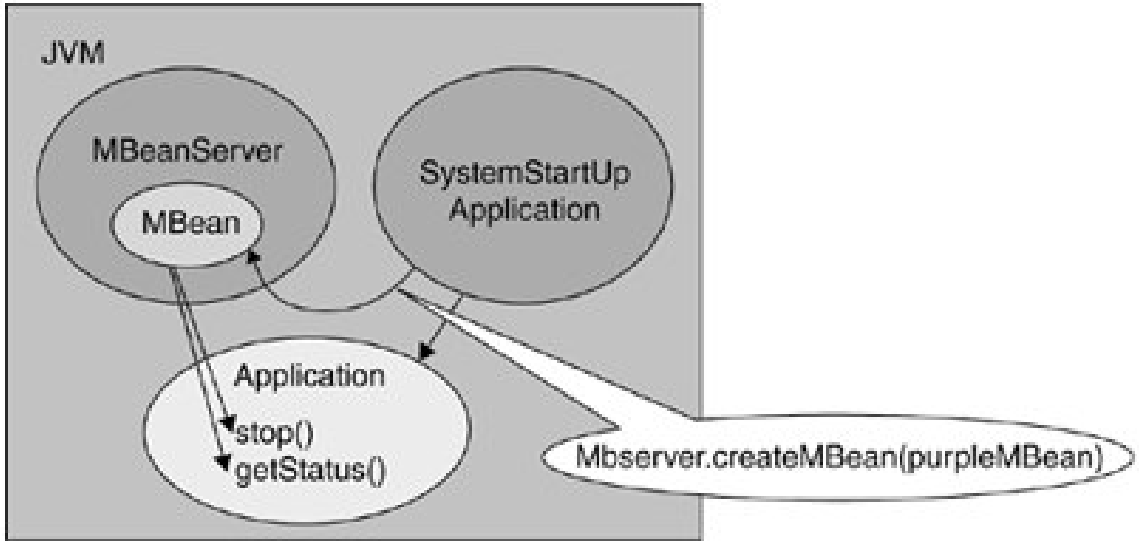
Création locale et enregistrement



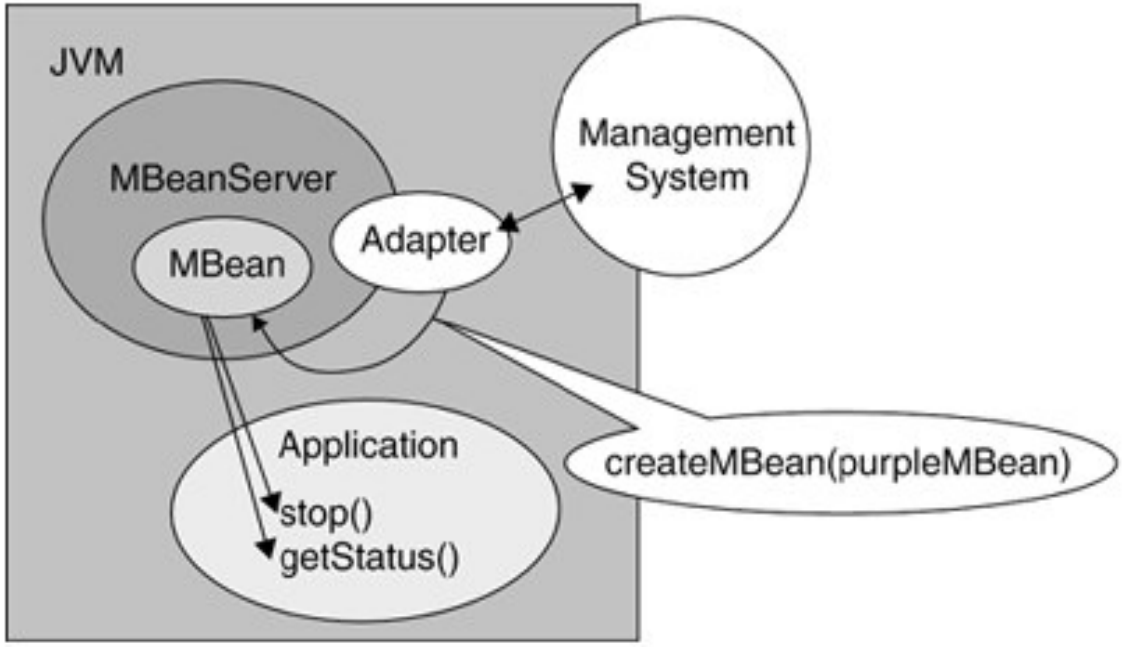
Création du MBean et enregistrement



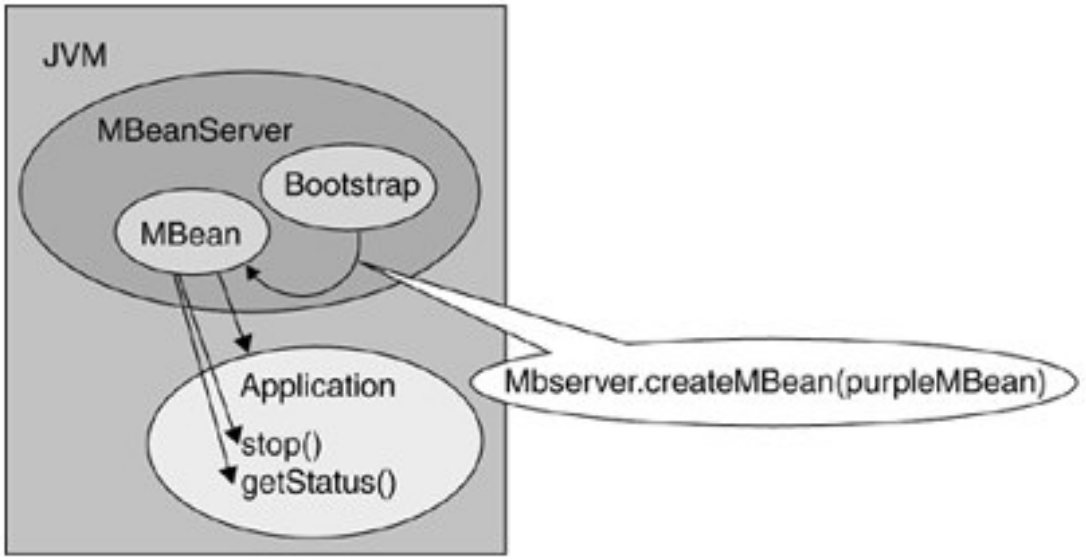
Création depuis une application externe



Création par un adaptateur



Création au démarrage



M-let et notifications

- **Être prévenu d'un téléchargement**
- **Voir l'adaptateur livre page 241**