JMX (Java Management Extension)

Sang Shin
Michèle Garoche
http://www.javapassion.com
“Learn with Passion!”

Topics

- What is and Why JMX?
- Architecture
- MBean
- JMX Services
- MBean server as JMX agent
- Step by step guide of instrumenting you Java application
- Accessing JMX agent
- Performance impact of using JMX
- MXBeans in Java SE 6
- Web services connector for JMX

What is and Why JMX?

Why Management & Monitoring?

- Infrastructure software is getting more complicated
  - Administration
  - Configuration
  - Monitoring
- Enterprise business application characteristics:
  - Distributed
  - Complex
  - Mission Critical
  - High-Volume
  - Dynamic

What is JMX?

- Defines the Architecture, Design Patterns, APIs and the Services for *exposing* and *managing* applications and network devices.
- Provides a means to:
  - Instrument Java code.
  - Implement distributed management middleware and managers.
  - Smoothly integrate these solutions into existing management systems.
- From Opaque Applications to Transparent Processes

What is JMX?

- Standard API for developing observable applications – JSR 3 and JSR 160
- Provides access to information such as
  - Number of classes loaded
  - Virtual machine uptime
  - Operating system information
- Applications can use JMX for
  - Management – changing configuration settings
  - Monitoring – getting statistics and notifications
- Mandatory from J2SE 5.0 and J2EE 1.4

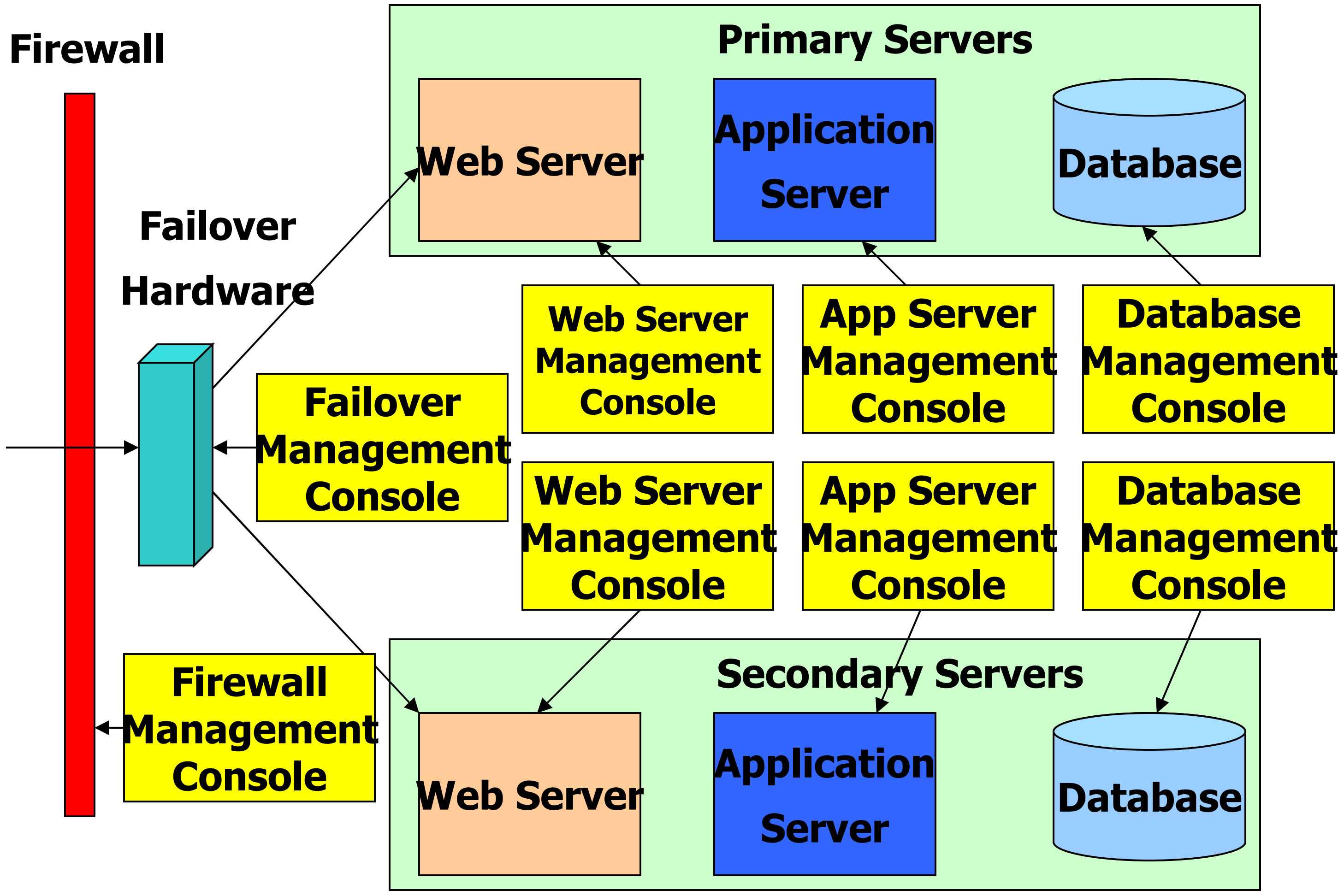
JMX Benefits

- Low Cost
- Scalable Management Architecture – Modularization of agent services
- Easy Integration – JMX smart agents manageable through various protocols
- Dynamic Management
- Integrates Existing Management Solutions
- Leverages Existing Standard Java Technologies
- Applicable to a Wide Range of Applications

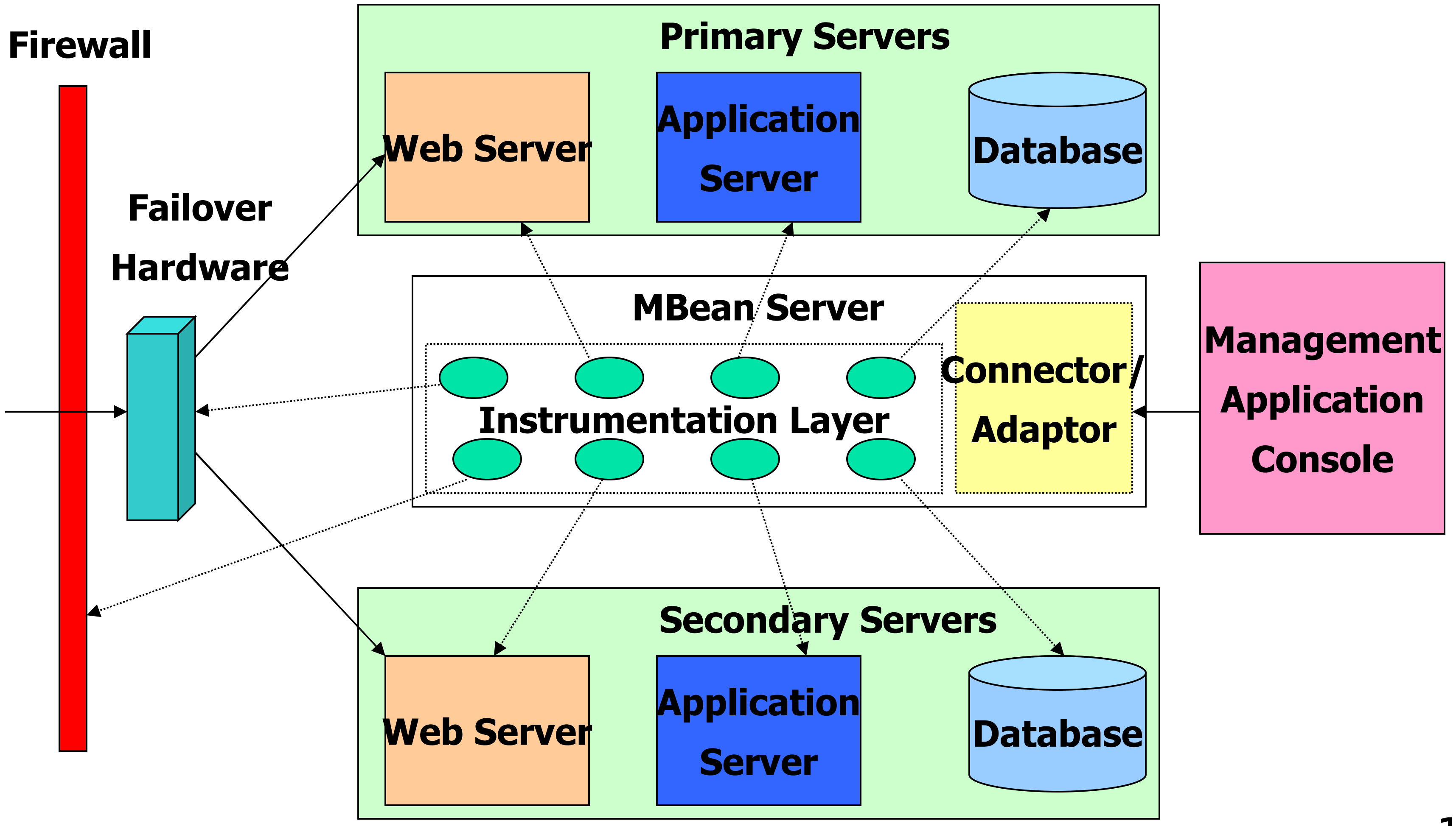
Typical JMX Usage

- Reading and Changing Application Configurations
- Infrastructure and Business Level Operational Statistics
  - Availability
  - Early Detection of Capacity Problems
  - Application Performance, Business Process Productivity
  - Resources usage
  - Problems
- Signaling events
  - Faults
  - State changes
  - Improving Services via Proactive Alerting

Management Before JMX


Firewall
Failover Hardware
Primary Servers
Web Server
Application Server
Database
Web Server Management Console
App Server Management Console
Database Management Console
Failover Management Console
Web Server Management Console
App Server Management Console
Database Management Console
Firewall Management Console
Secondary Servers
Web Server
Application Server
Database

Management With JMX

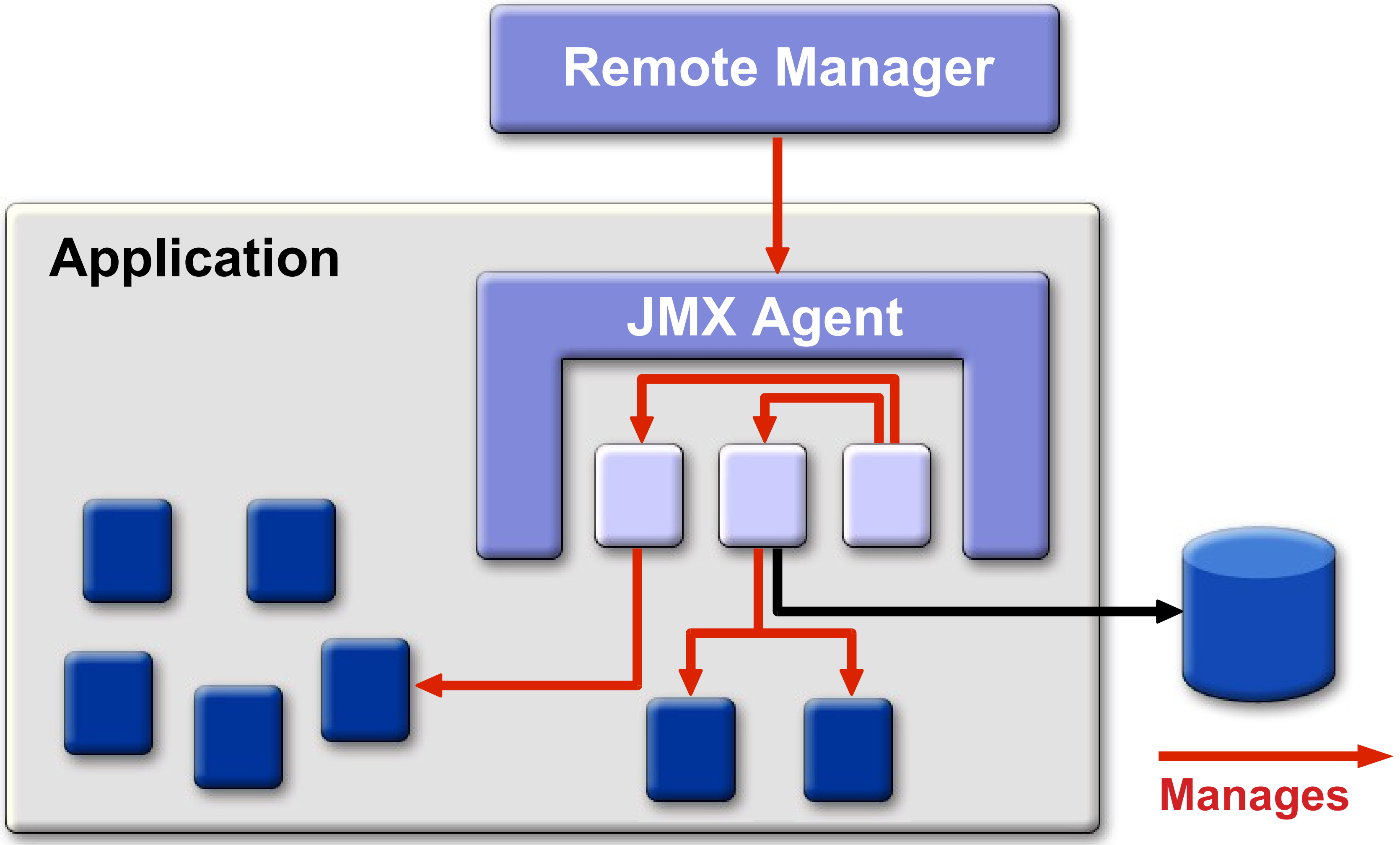


Architecture

JMX Architecture

- Instrumentation Level
  - *MBeans* instruments resources, exposing attributes and operations
- Agent Level
  - MBean Server
  - Predefined services
- Remote Management
  - Protocol Adaptors and Standard Connectors enables remote Manager Applications

JMX Architecture

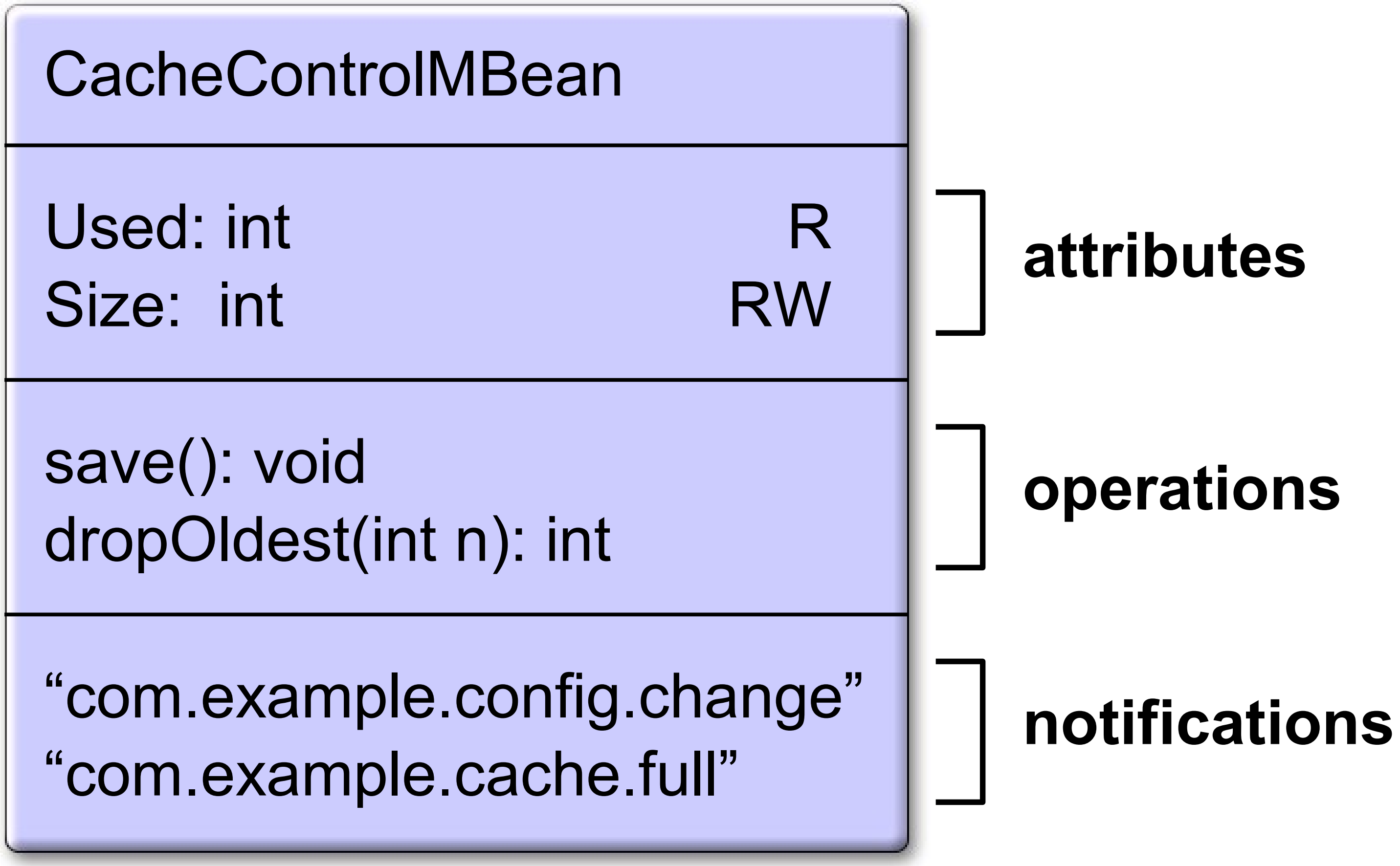


MBean

Managed Beans(MBeans)

- A *MBean* is a named *managed object* representing a *resource* - resource could be anything
  - An application configuration setting
  - EJB
  - Device
  - Any resource to be managed
- A *MBean* can have
  - Attributes that can be read and/or written
  - Operations that can be invoked
  - Notifications that the MBean can broadcast

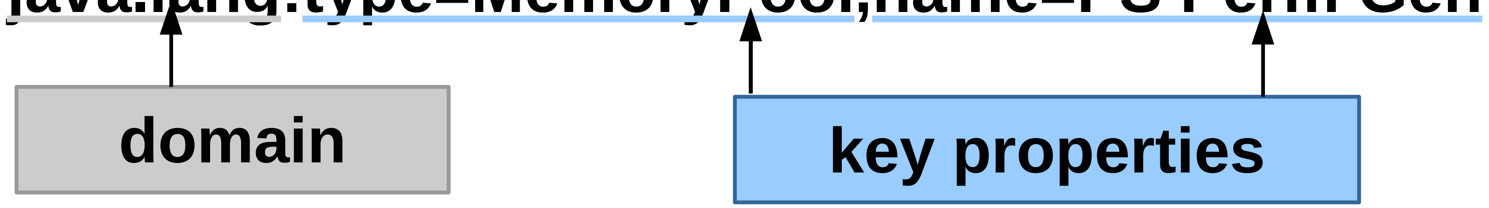
A MBean Example



Naming MBeans

- Every MBean has a name
- A name is an instance of the ObjectName class (*javax.management.ObjectName*)
- A name has a *domain* and one or more *key properties*

com.example:type=CacheControl
com.example:type=CacheControl,name=whatsitCache
java.lang:type=Threading
java.lang:type=MemoryPool,name=PS Perm Gen



Standard MBean

- Standard MBean is the simplest model to use
  - Quickest and Easiest way to instrument static manageable resources
- Steps to create a standard MBean
  - Create an Java interface call **`FredMBean`**
  - Follows JavaBeans naming convention
  - Implement the interface in a class call **`Fred`**
- An instance of **`Fred`** is the MBean

Dynamic MBean

- Expose attributes and operations at Runtime
- Provides more flexible instrumentations
- Step to create Dynamic MBeans
  - Implements DynamicMBeans interface
  - Method returns all Attributes & Operations
- The same capability as Standard MBeans from Agent’s perspective

DynamicMBean Interface

<<Interface>>
DynamicMBean

```
getMBeanInfo():MBeanInfo
getAttribute(attribute:String):Object
getAttributes(attributes:String[]):AttributeList
setAttribute(attribute:Attribute):void
setAttributes(attributes:AttributeList):AttributeList
invoke(actionName:String,
            params:Object[],
            signature:String[]):Object
```

JMX Notification

- JMX notifications consists of the following
  - **`NotificationEmitter`** – event generator, typically your MBean
  - **`NotificationListener`** – event listener
  - **`Notification`** – the event
  - **`NotificationBroadcasterSupport`** – helper class
- *NotificationListener* object registers with MBean server to receive events

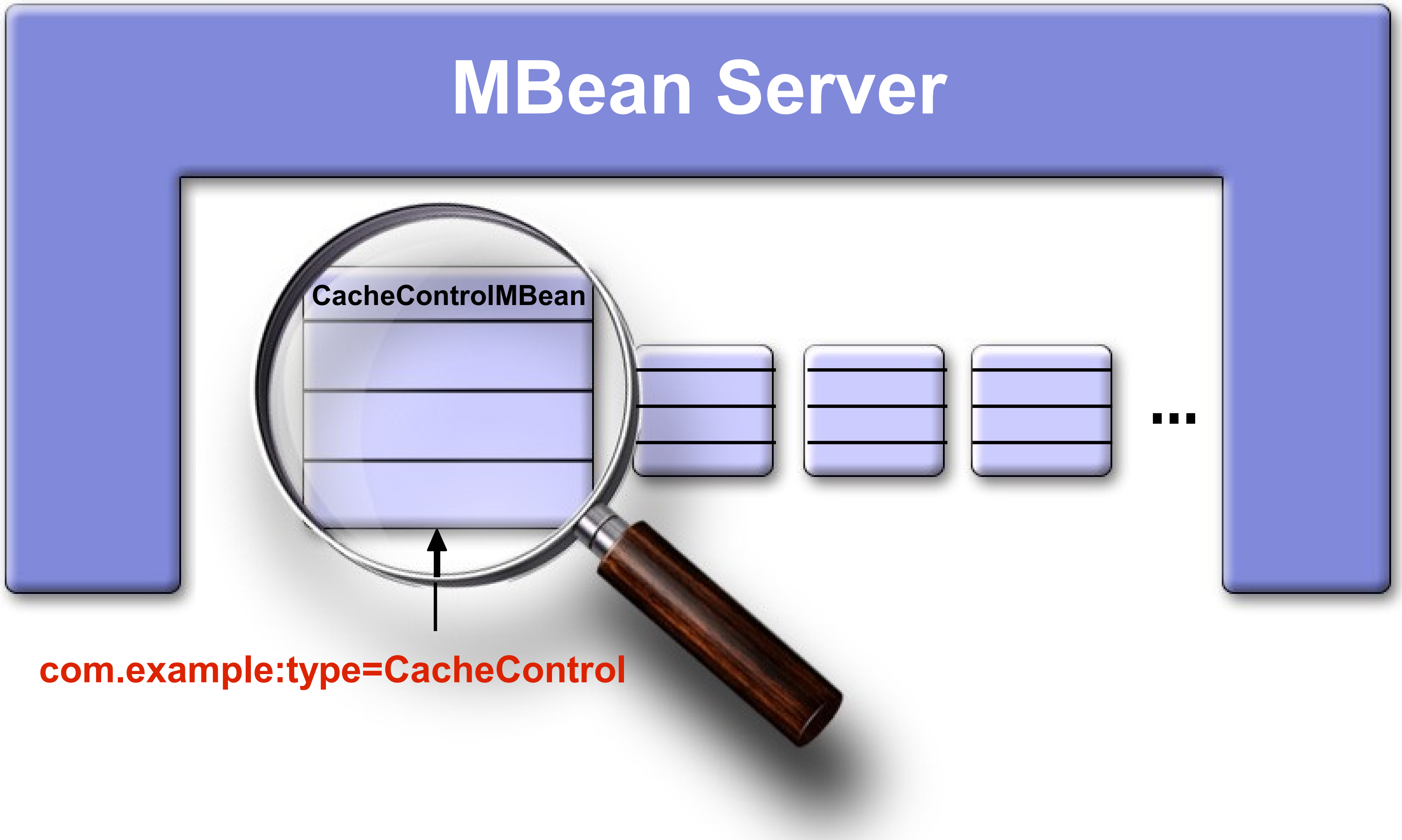
JMX Services

JMX Services

- JMX API includes a number of pre-defined services
  - Services are themselves MBeans
- Monitoring service (thresholding)
  - javax.management.monitor
- Relation service (relations between MBeans)
  - javax.management.relation
- Timer service
  - javax.management.timer
- M-let service
  - javax.management.loading

MBean Server - JMX Agent

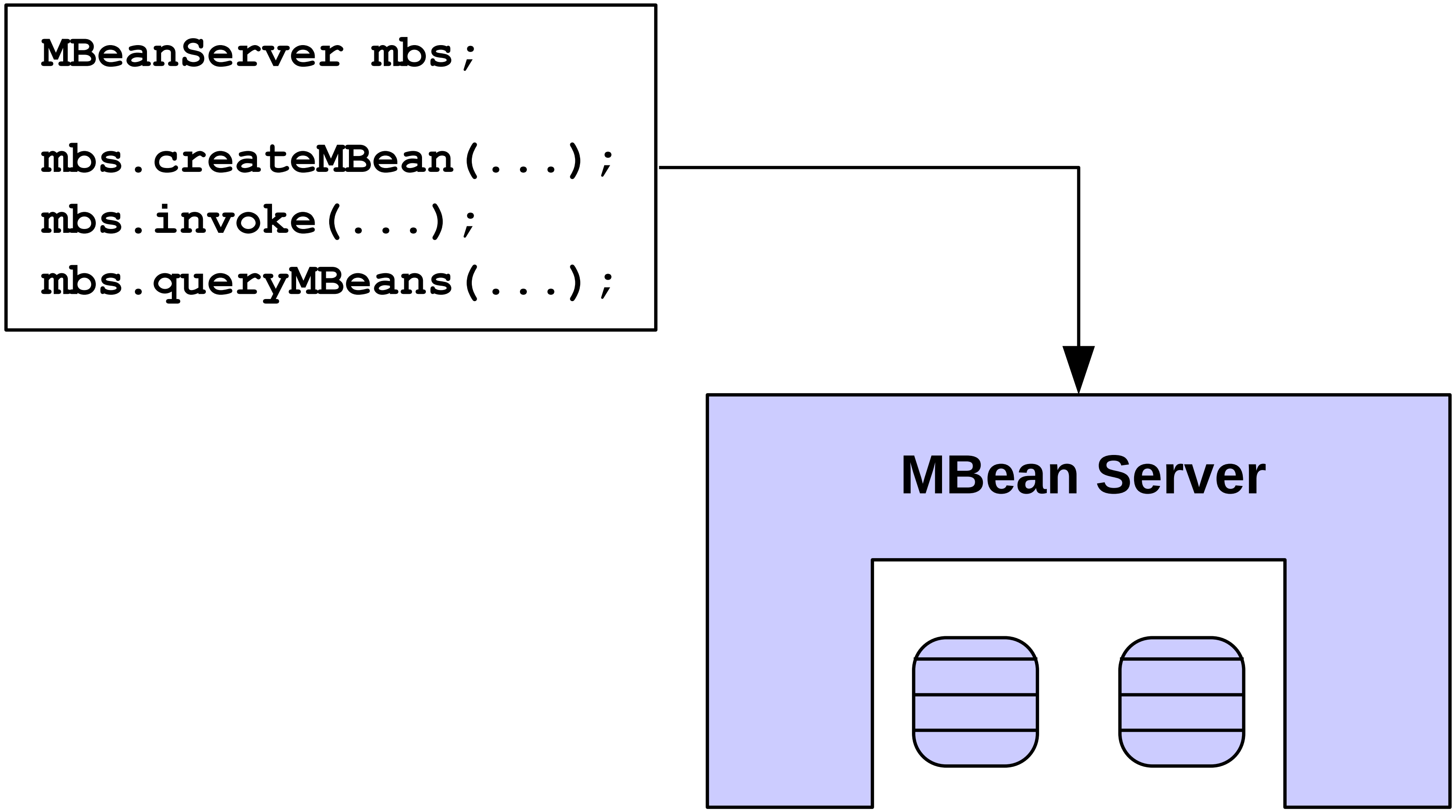
MBean Server as JMX Agent



MBean Server

- To be useful, an MBean must be registered in an *MBean Server*
- Each MBean is registered with its ObjectName
- Usually, the only access to MBeans is through the MBean Server
- You can have more than one MBean Server per Java™ Virtual Machine (JVM™ machine)
- But usually, as of Java 5.0, everyone uses the *Platform MBean Server*
  - java.lang.management.ManagementFactory.getPlatformMBeanServer()

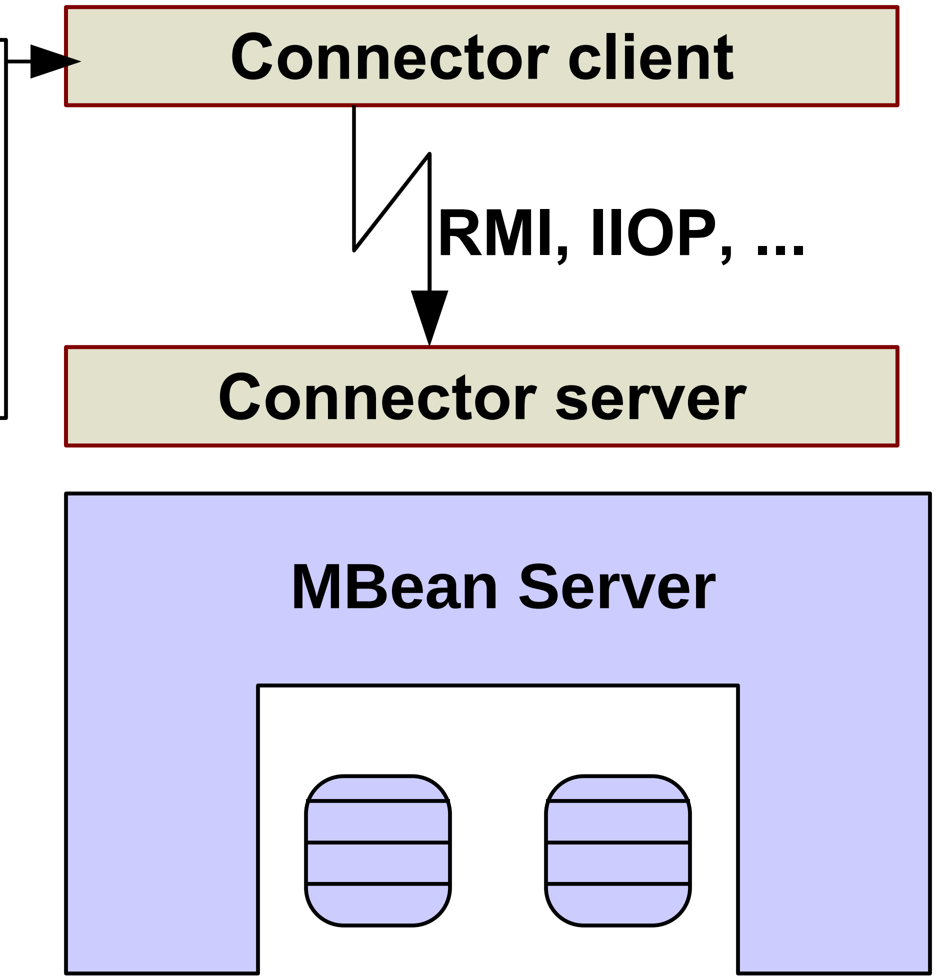
MBean Server: Local Clients



```
MBeanServer mbs;

mbs.createMBean(...);
mbs.invoke(...);
mbs.queryMBeans(...);
```

MBean Server: Connector Clients

```
MBeanServerConnection
     mbs;
mbs.createMBean(...);
mbs.invoke(...);
mbs.queryMBeans(...);
```



MBean Server: Connectors

- Connectors defined by the JMX Remote API (JSR 160)
- Java platform includes RMI and RMI/IIOP connectors
- JSR 160 also defines a purpose-built protocol, JMXMP
  - > fits into existing security infrastructures.
- Current work: a SOAP-based connector for the Web Services world
- Some non standard Connector implementation
  - > eg: JMS (project lingo), HTTP (glassfish)

Step By Step Process of Instrumenting Your Java Application

Original JMX Free Application

- Change the display value by buttons clicks; Count click number at back end
- ClickFrame
  - Main JFrame View
- ClickModel
  - Contains displayNumber and countNumber
- ClickCounterRunner
  - Wrapper Runner

Instrument ClickCounter
(Standard MBean)

1. Create MBean interface
   - *ClickCounterStdMBean*
2. Implement MBean interface
   - Create *ClickCounterStd* implementing *ClickCounterStdMBean*, and extending *ClickFrame* as well
3. Get System MBean Server
4. Register MBean
   - *ClickCounterStd*

1. Create MBean Interface

```
public interface ClickCounterStdMBean {
   public void reset();
   public int getDisplayNumber();
   public void setDisplayNumber(int inNumber);
   public int getCountNumber();
}
```

2. Implement MBean Interface

```
public class ClickCounterStd
    extends ClickFrame
    implements ClickCounterStdMBean {

  public void reset() {
      getModel().reset();
      updateLabel();
  }

  public int getDisplayNumber() {
      return getModel().getDisplayNumber();
  }
  . . . . . .
}
```

3. Get Platform MBean Server
4. Register MBean

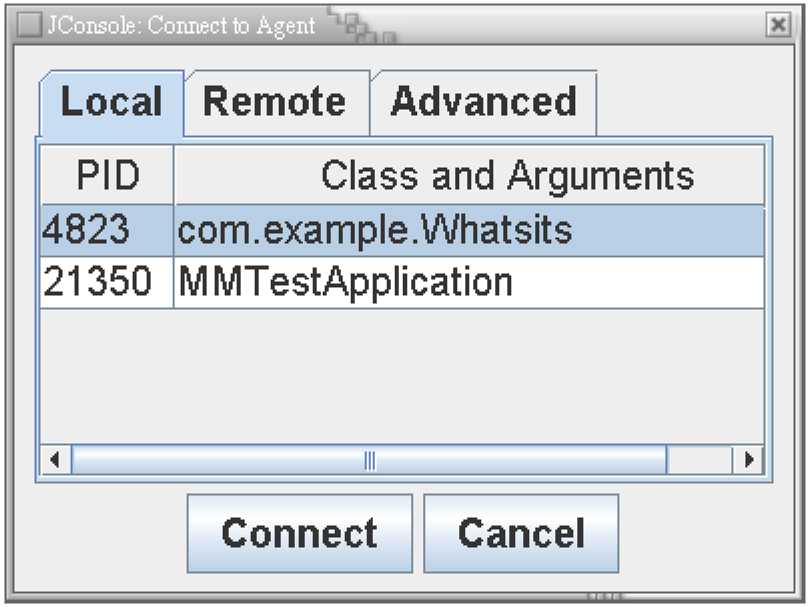
```
 MBeanServer mbs = ManagementFactory
        .getPlatformMBeanServer();

ObjectName name = new ObjectName(
   “shen.joey.demo.ClickCount:type=ClickCounterStd”);


ClickFrame counter = new ClickCounterStd();

mbs.registerMBean(counter, name);
```

Accessing JMX Agent

Accessing the JMX Agent

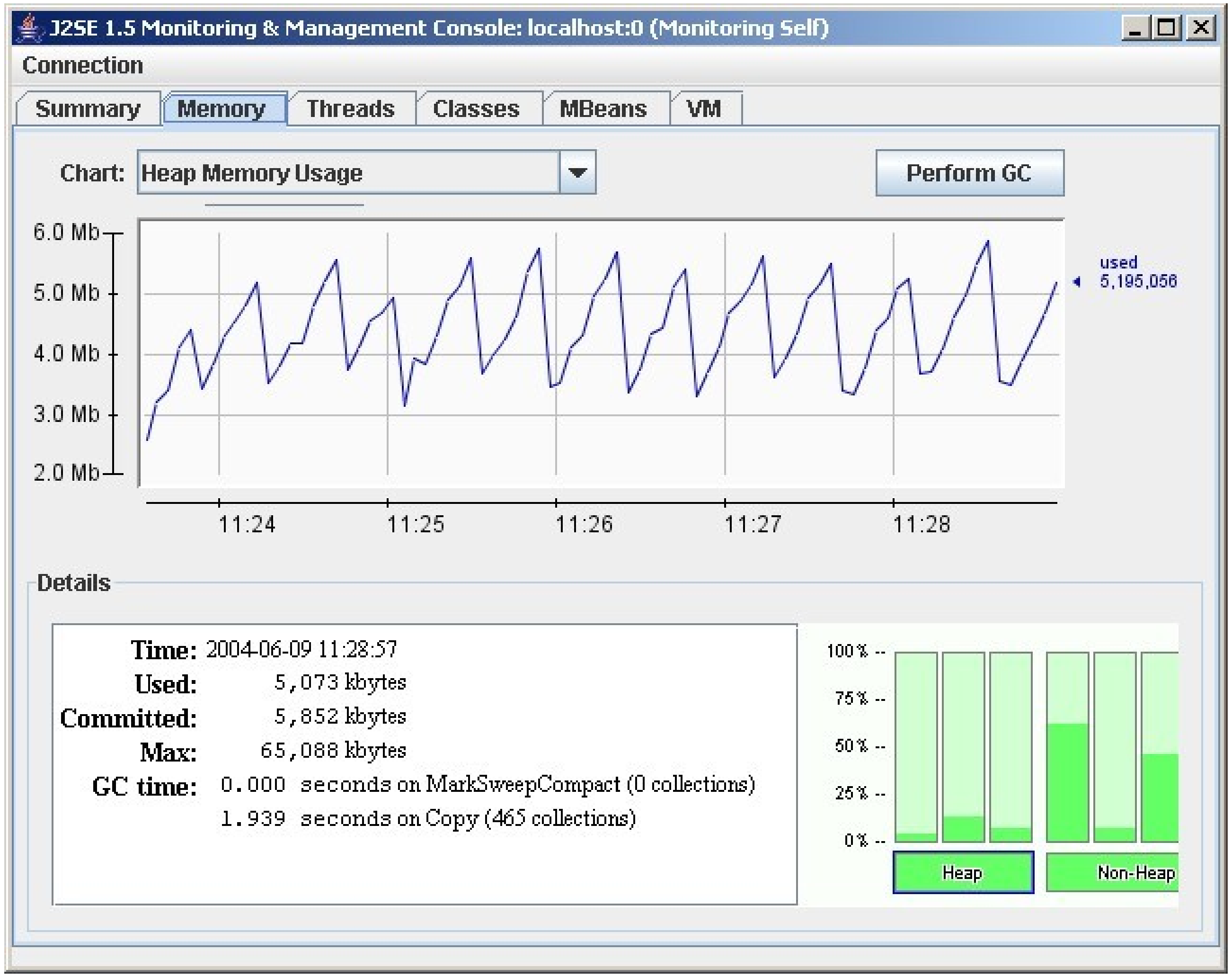
- J2SE 5.0 and later releases
  - `java -Dcom.sun.management.jmxremote Main`
  - See **http://java.sun.com/j2se/1.5.0/docs/guide/management/agent** for more options
- Start jconsole
- In JavaSE 6.0, you can attach jconsole without settng `com.sun.management.jmxremote` property
  - Dynamic attachment
- One click monitoring with NetBeans IDE

JConsole connection dialog (JDK 5)



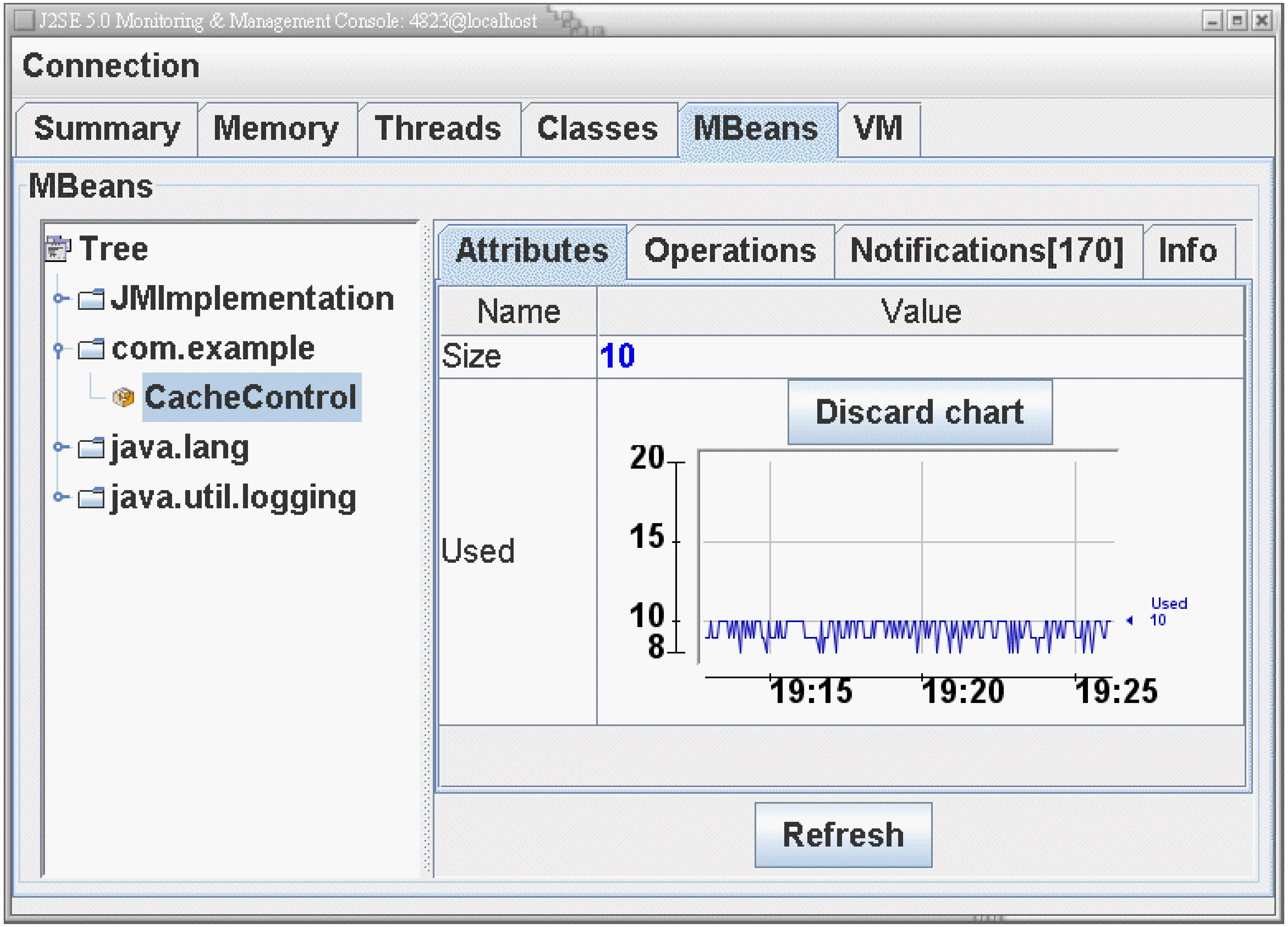
PID	Class and Arguments
4823	com.example.Whatsits
21350	MMTestApplication

Memory Tab



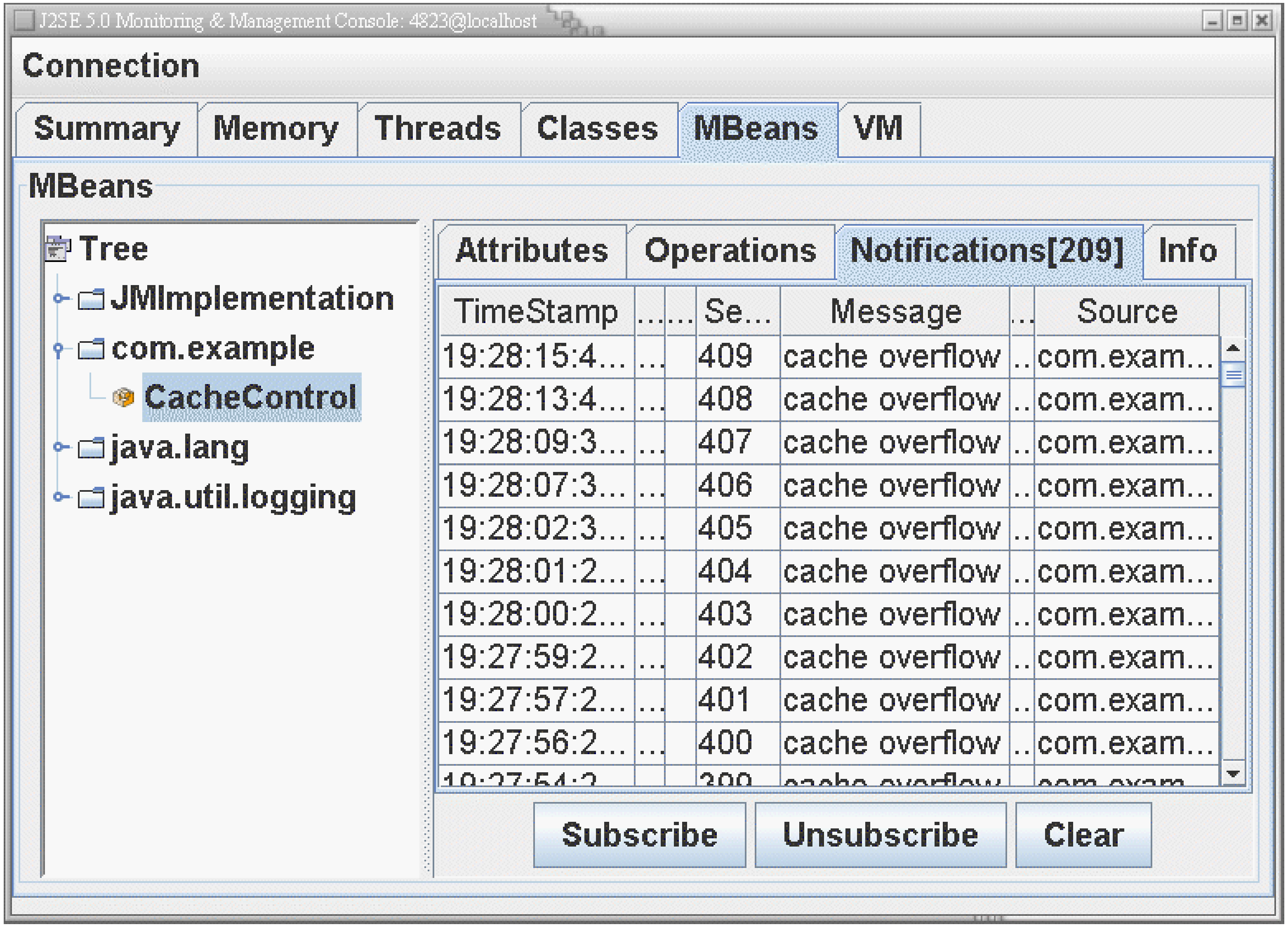
Time:	2004-06-09 11:28:57
Used:	5,073 kbytes
Committed:	5,852 kbytes
Max:	65,088 kbytes
GC time:	0.000 seconds on MarkSweepCompact (0 collections)
	1.939 seconds on Copy (465 collections)

MBean Tab



Name	Value
Size	10
Used	

MBean Notifications



J2SE 5.0 Monitoring & Management Console: 4823@localhost

Connection

Summary | Memory | Threads | Classes | MBeans | VM

MBeans

Tree
- JMImplementation
- com.example
  - CacheControl
- java.lang
- java.util.logging

Attributes | Operations | Notifications[209] | Info

TimeStamp	...	...	Se...	Message	...	Source
19:28:15:4...	...		409	cache overflow	..	com.exam...
19:28:13:4...	...		408	cache overflow	..	com.exam...
19:28:09:3...	...		407	cache overflow	..	com.exam...
19:28:07:3...	...		406	cache overflow	..	com.exam...
19:28:02:3...	...		405	cache overflow	..	com.exam...
19:28:01:2...	...		404	cache overflow	..	com.exam...
19:28:00:2...	...		403	cache overflow	..	com.exam...
19:27:59:2...	...		402	cache overflow	..	com.exam...
19:27:57:2...	...		401	cache overflow	..	com.exam...
19:27:56:2...	...		400	cache overflow	..	com.exam...
19:27:54:2			399	cache overflow		com.exam

Visual VM, a new Visual Tool

- Experimental for now
- Combines monitoring, troubleshooting and profiling capabilities.
- A java.net project
- Preview 2 available for download

Demo:

Monitoring JMX-Enabled Java Application with JConsole

http://www.netbeans.org/kb/60/java/jmx-tutorial.html

Demo:

Monitoring GlassFish App. Server with JMX/JConsole

Performance Impact of Using JMX

(Quoted from Eamonn McManus's Blog http://weblogs.java.net/blog/emcmanus/archive/2006/07/how_much_does_i.html)

Based on Unscientific Testing

- Quesiton:
  - If I run with *-Dcom.sun.management.jmxremote* and connect *jconsole*, how much will that affect the performance of my app?
- Answer
  - Running with *-Dcom.sun.management.jmxremote* has no appreciable impact.
    - So long as you don't connect jconsole, enabling monitoring should have no effect on your app's performance.
  - Connecting jconsole has an impact of 3--4%.

MXBeans in Java SE 6

MXBeans: Problem statement (1)

- An MBean interface can include arbitrary Java™ programming language types

```
public interface ThreadMBean {
   public ThreadInfo getThreadInfo();
}
public class ThreadInfo {
   public String getName();
   public long getBlockedCount();
   public boolean isSuspended();
   ...
}
```

MXBeans: Problem statement (2)

- Needed when values must be grouped atomically
- But:
  - Client must have these classes
  - What about generic clients like jconsole?
  - What about versioning?

Open MBeans

- JMX™ API defines Open MBeans
  - javax.management.openmbean
- Predefined set of basic types
  - Integer, String, Date, ObjectName, ...
- Complex types made using arrays and/or two predefined compositional types
  - CompositeData
  - TabularData

MXBeans (1)

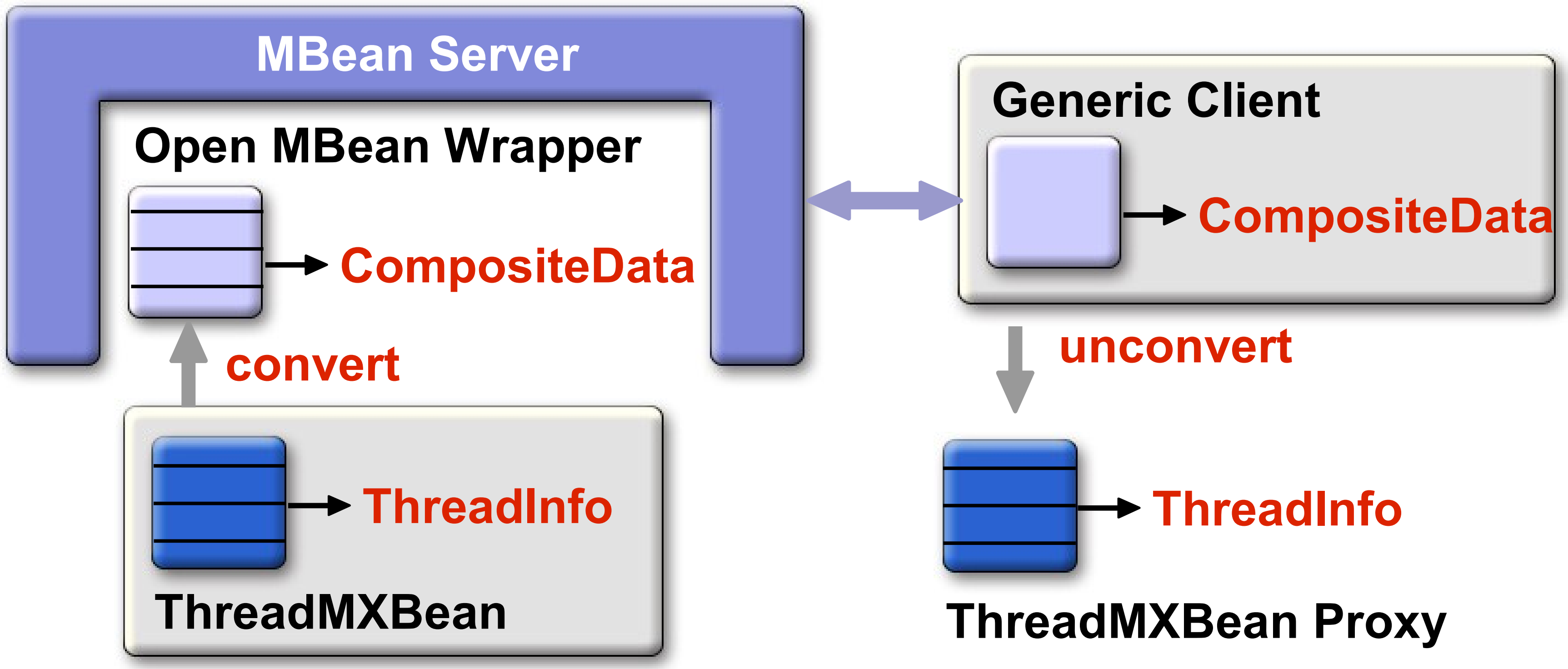
- MXBeans were designed for the instrumentation of the VM itself (JSR 174)
  - Already exist in java.lang.management
  - User-defined MXBeans are new in Java SE 6
- Management interface still a bean interface
- Can reference arbitrary Java Bean
- JMX™ API wraps an instance of this interface in an Open MBean

MXBeans (2)

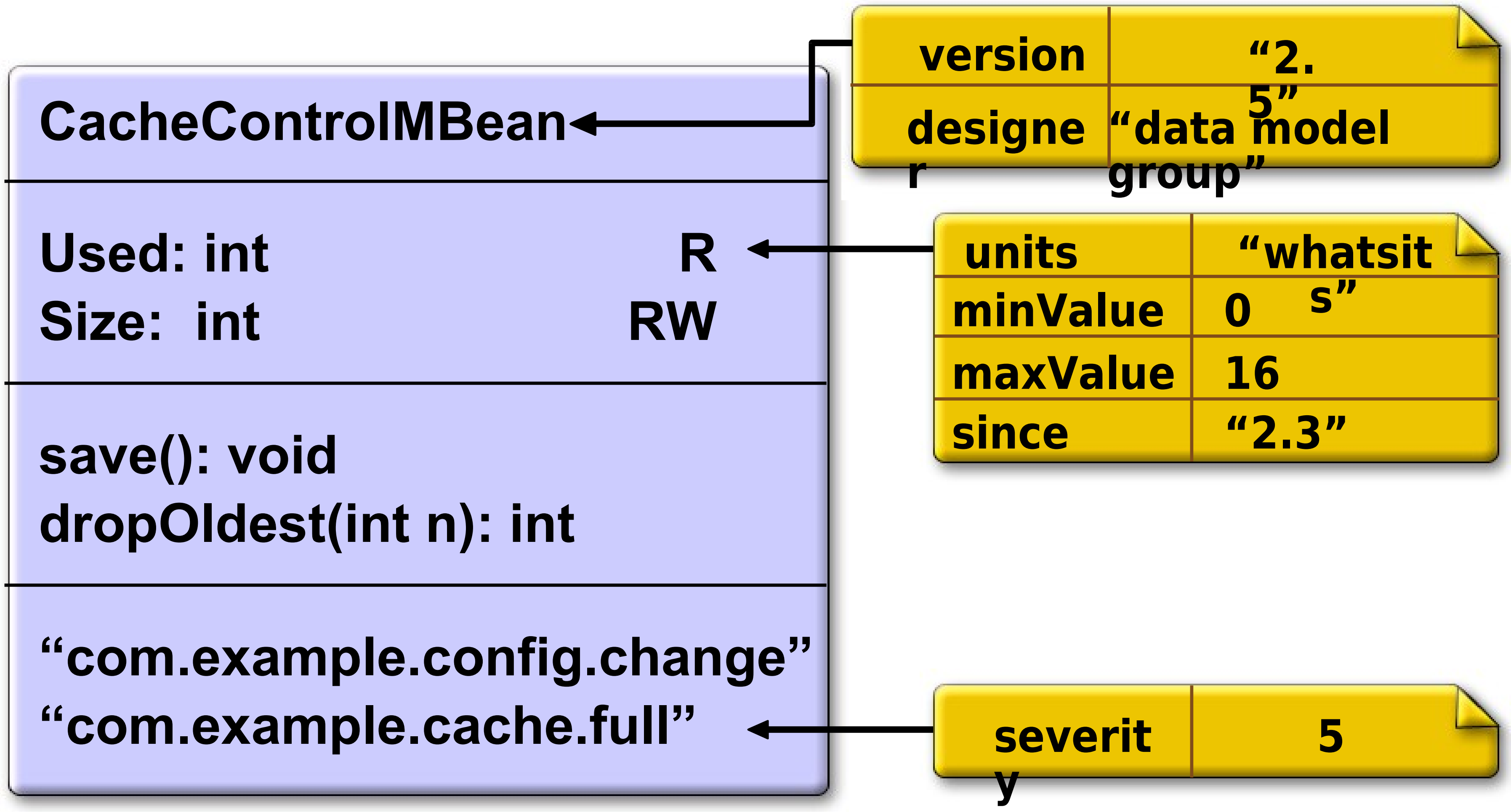
```
public interface ThreadMXBean {
    public ThreadInfo getThreadInfo();
}

public class ThreadMXBeanImpl implements ThreadMXBean {
   // Do not need Something/SomethingMXBean naming
   public ThreadInfo getThreadInfo() {
      return new ThreadInfo(...);
   }
}

ThreadMXBean mxbean = new ThreadMXBeanImpl();
ObjectName name =
   new ObjectName("java.lang:type=Threading");

mbs.registerMBean(mxbean, name);
```

MXBeans (3)

- Generic client can access as Open MBean
- Model-aware client can make ThreadMXBean proxy

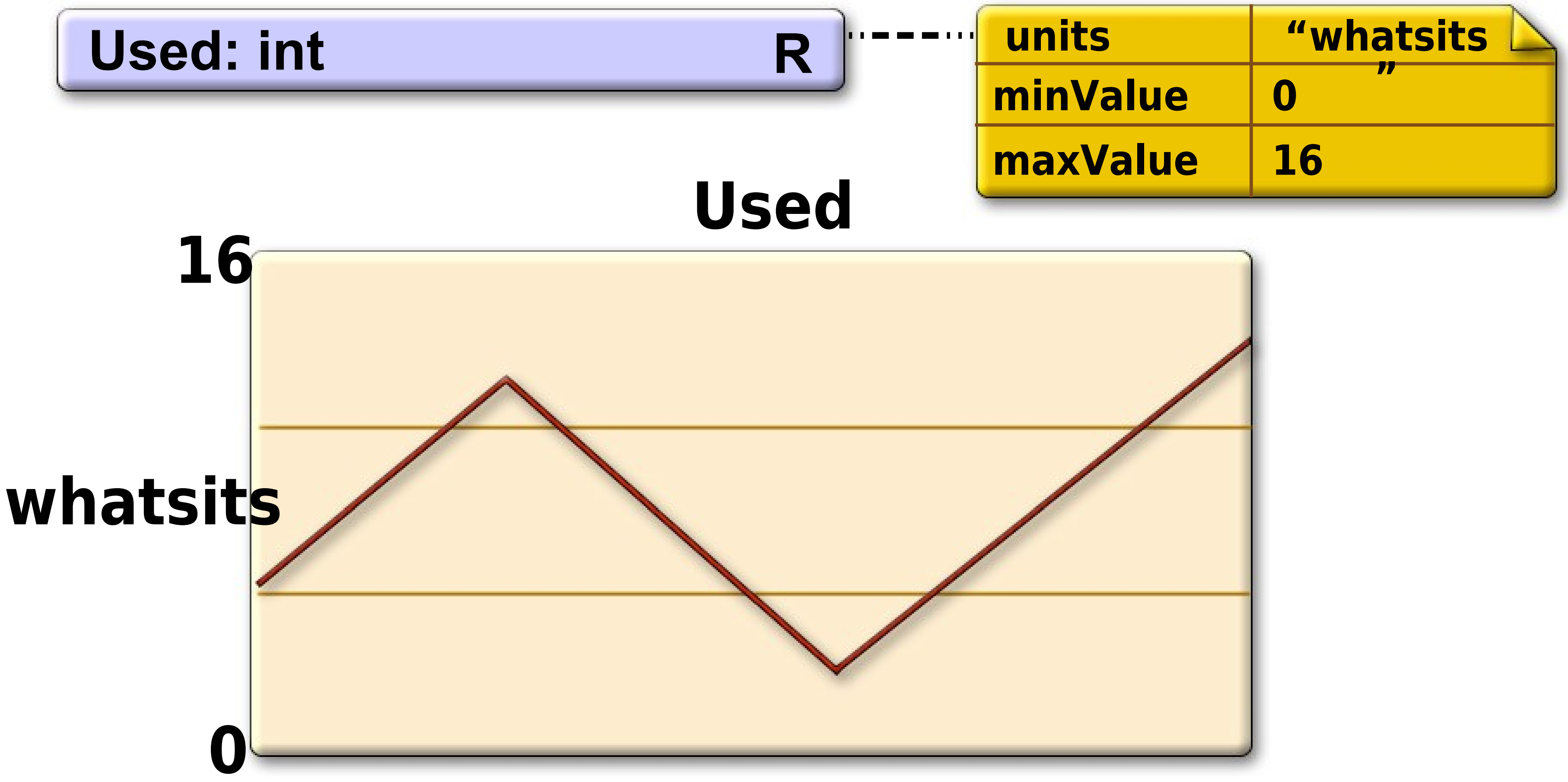


Descriptors



Descriptors and Generic Clients

(like jconsole)



units	“whatsits”
minValue	0
maxValue	16

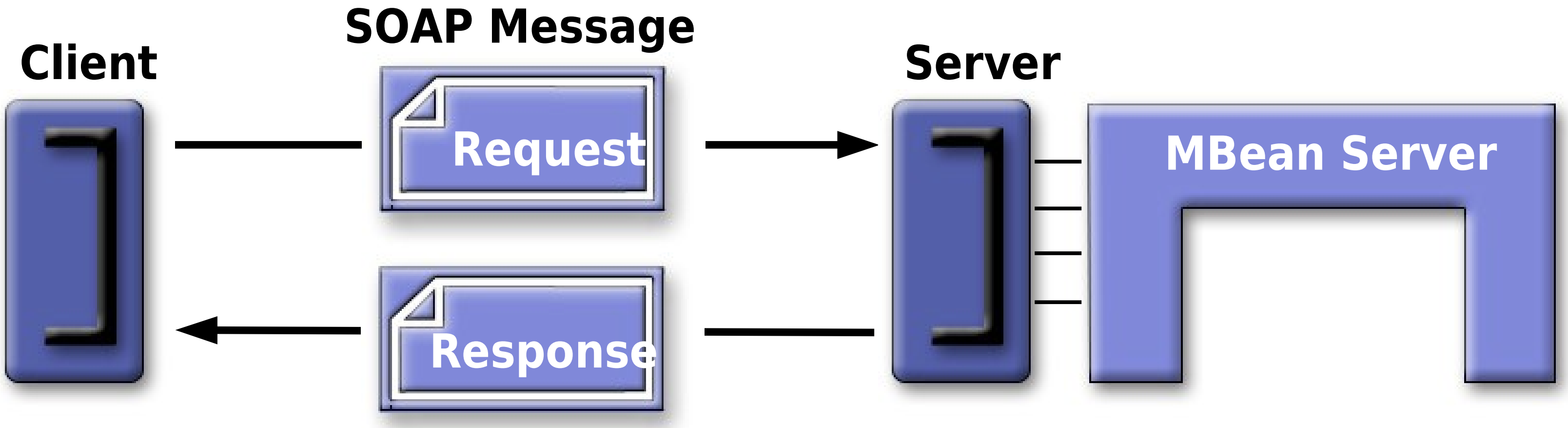
Descriptor details

- Classes MBeanInfo, MBeanAttributeInfo, etc., now have an optional Descriptor
- Every attribute, operation, notification can have its own Descriptor
- Descriptor is set of (key,value) pairs
- Some keys have conventional meanings
- Users can add their own keys
- Descriptors have always existed in Model MBeans

Web Services Connector for JMX

Web Services Connector for JMX

- Being defined by JSR 262
- Java platform clients can use JMX Remote API
- Allows clients of JMX agents on non-Java platforms
- Can exploit web infrastructure



Connector details

- JMX Remote API compliant connector
  - service:jmx:ws://<host>:<port>/<http context>
- WS-Management access for non JMX client
- Security
  - Authentification : HTTP Basic Auth
  - Encryption : HTTPS
  - Authorization : JMX Permission
  - WS-Security pluggability (XWSS)

Connector Reference Implementation

- Early Access 2
- Runs on J2SE 5.0 and Java SE 6
- JAX-WS 2.1 Endpoint
- Open Source WS-Man Java implementation
  - WiseMan java.net project
  - 1.0 release
- Proven interoperability with Microsoft WinRM

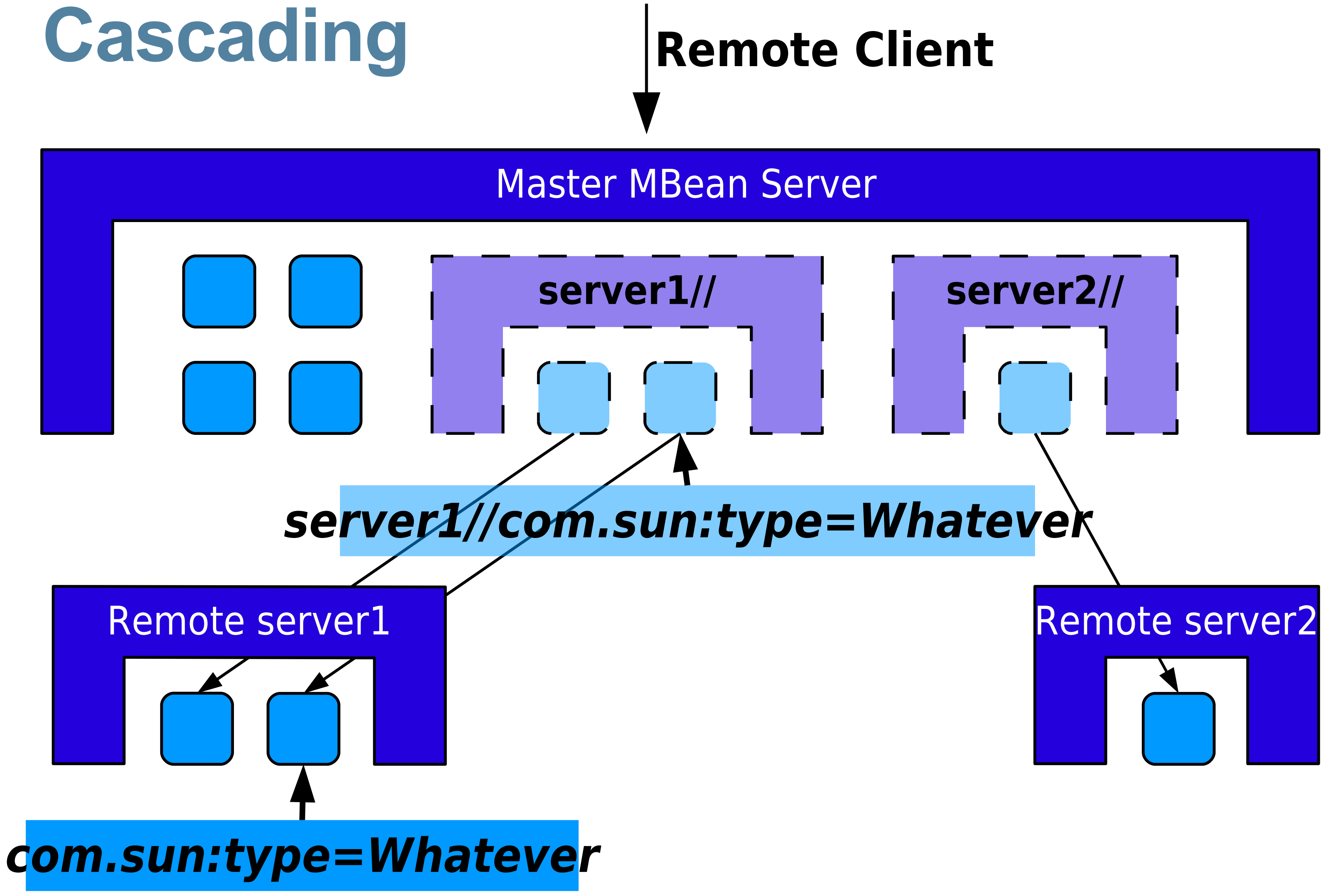
Future JMX Features in Java SE 7

JMX 2.0 defined by JSR 255

Namespaces and Cascading

- Hierarchical namespaces
  - Namespaces are like filesystem directories
  - ObjectNames beginning with foo// are in the foo// namespace
    - foo//com.sun:type=Whatever
- Cascading
  - Uses a special Namespace that forwards everything to a remote MBean Server
  - Clients can connect to the Master MBean Server and access MBeans in remote servers

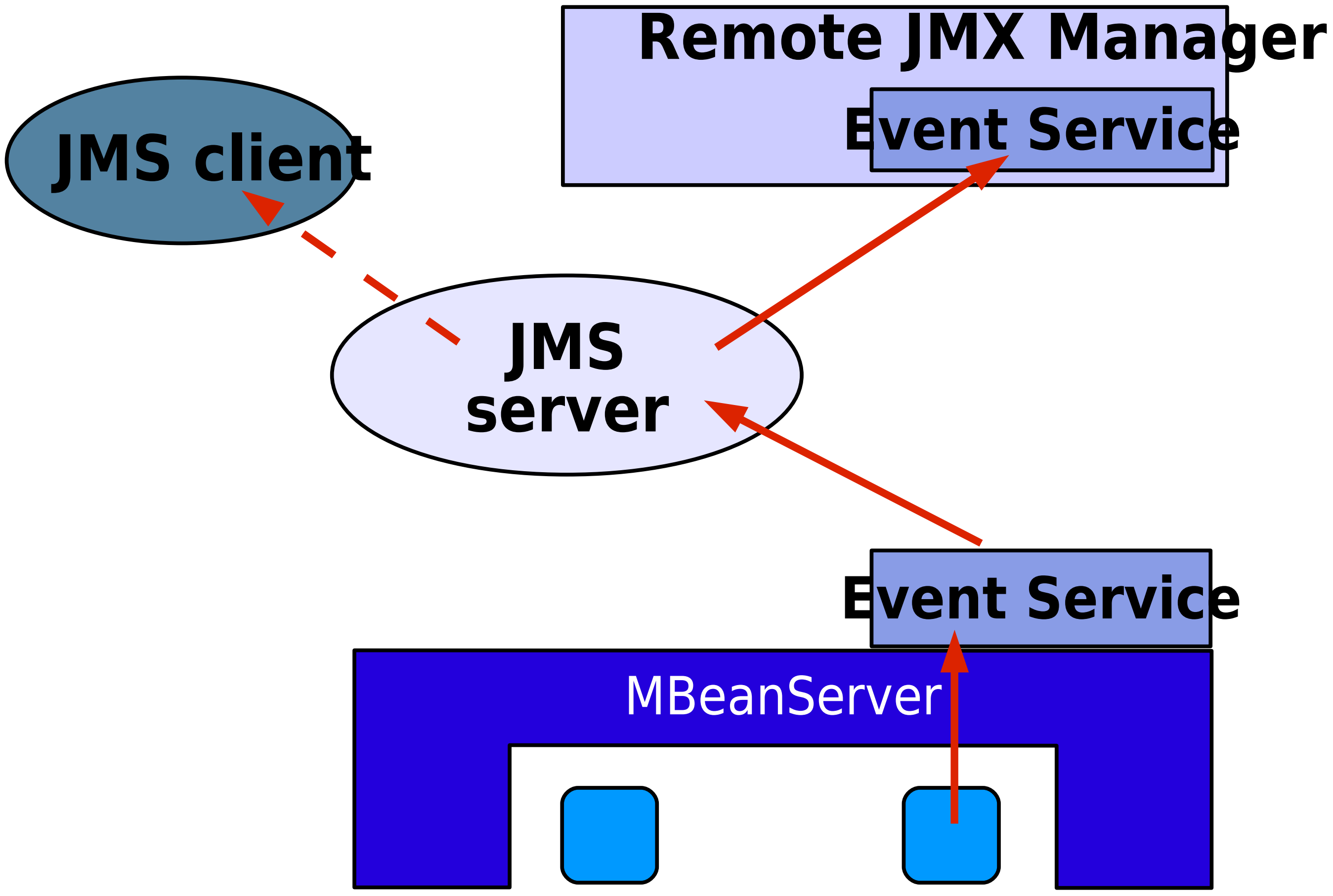
Cascading


Remote Client
Master MBean Server
server1//
server2//
server1//com.sun:type=Whatever
Remote server1
Remote server2
com.sun:type=Whatever

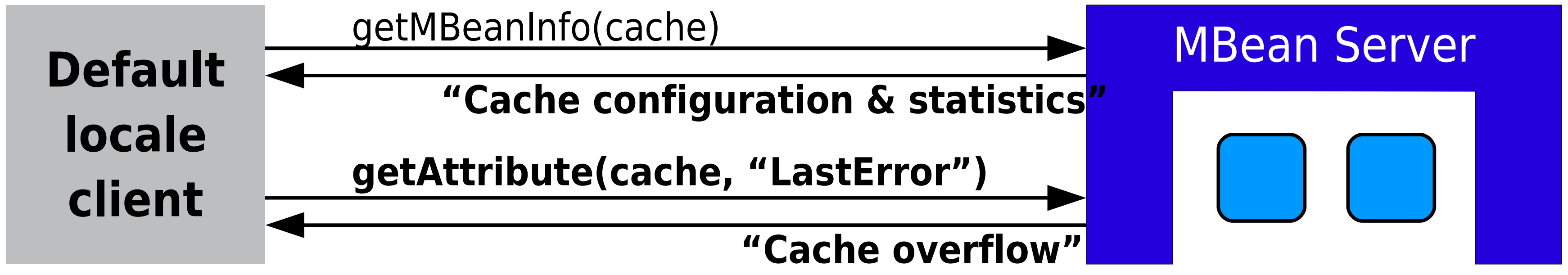
Event Service

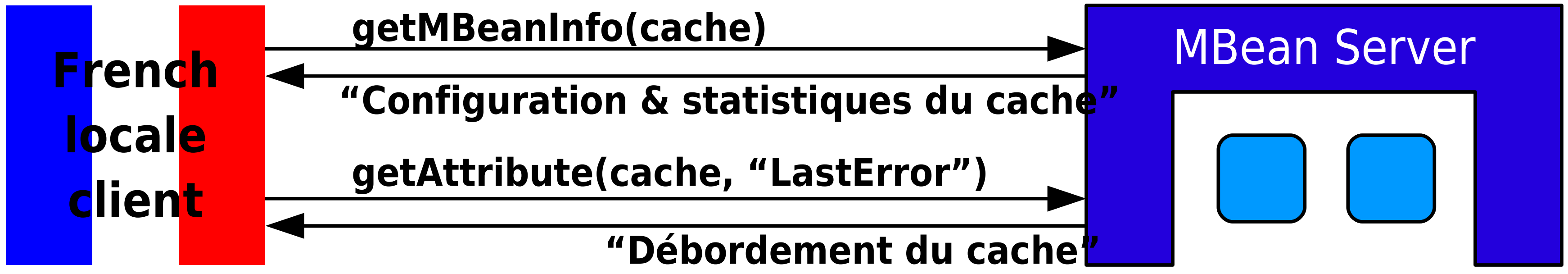
- New JMX service to handle Notification subscription and delivery
- An Open Architecture
  - Pluggable Notification Transport (eg: JMS, UDP)
  - Open to non-JMX world or non-Java world (eg: JMS, SMS, SNMP, ...)
  - Independent of JMX Connector
- Better scalability and extensibility
- Pull and Push delivery modes

EventService / JMS transport



Support for Locale





Annotations to define MBeans

- Number one feature request from our users
- Already exists in some environments, e.g. Spring

```
@ManagedResource
public class CacheControl ~~implements CacheControlMBean~~ {
  @ManagedAttribute
  public int getSize() {...}
  @ManagedAttribute
  public void setSize(int size) {...}
  @ManagedAttribute
  public int getUsed() {...}
  @ManagedOperation
  public int dropOldest(int n) {...}
```

Resources

For More Information

- http://java.sun.com/jmx
- jmx-forum@java.sun.com
- JMX team bloggers
  - > http://weblogs.java.net/blog/emcmanus
  - > http://blogs.sun.com/jmxetc/
  - > http://blogs.sun.com/lmalventosa/
  - > http://blogs.sun.com/jmxnetbeans/
  - > http://blogs.sun.com/joel/
  - > More bloggers on http://blogs.sun.com/

Resources

- Web Services Connector : http://ws-jmx-connector.dev.java.net/
- Visual VM : http://visualvm.dev.java.net/
- Lingo JMS Connector: http://lingo.codehaus.org/JMX+over+JMS
- Glassfish HTTP Connector: https://glassfish.dev.java.net/javaee5/admin-infra/http-jmx-connec

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps