Java™ Management Extensions (JMX™) Technology Today and Tomorrow

Éamonn McManus

JMX Specification Lead
Sun Microsystems
http://java.sun.com/jmx

Jean-François Denise

JMX Technology Team
Sun Microsystems
http://java.sun.com/jmx

TS-3523

Goals

Learn new ways to link Java™ Management Extensions (JMX™) MBeans to the rest of your application

Learn what is coming in future versions of the JMX API

Agenda

Demo

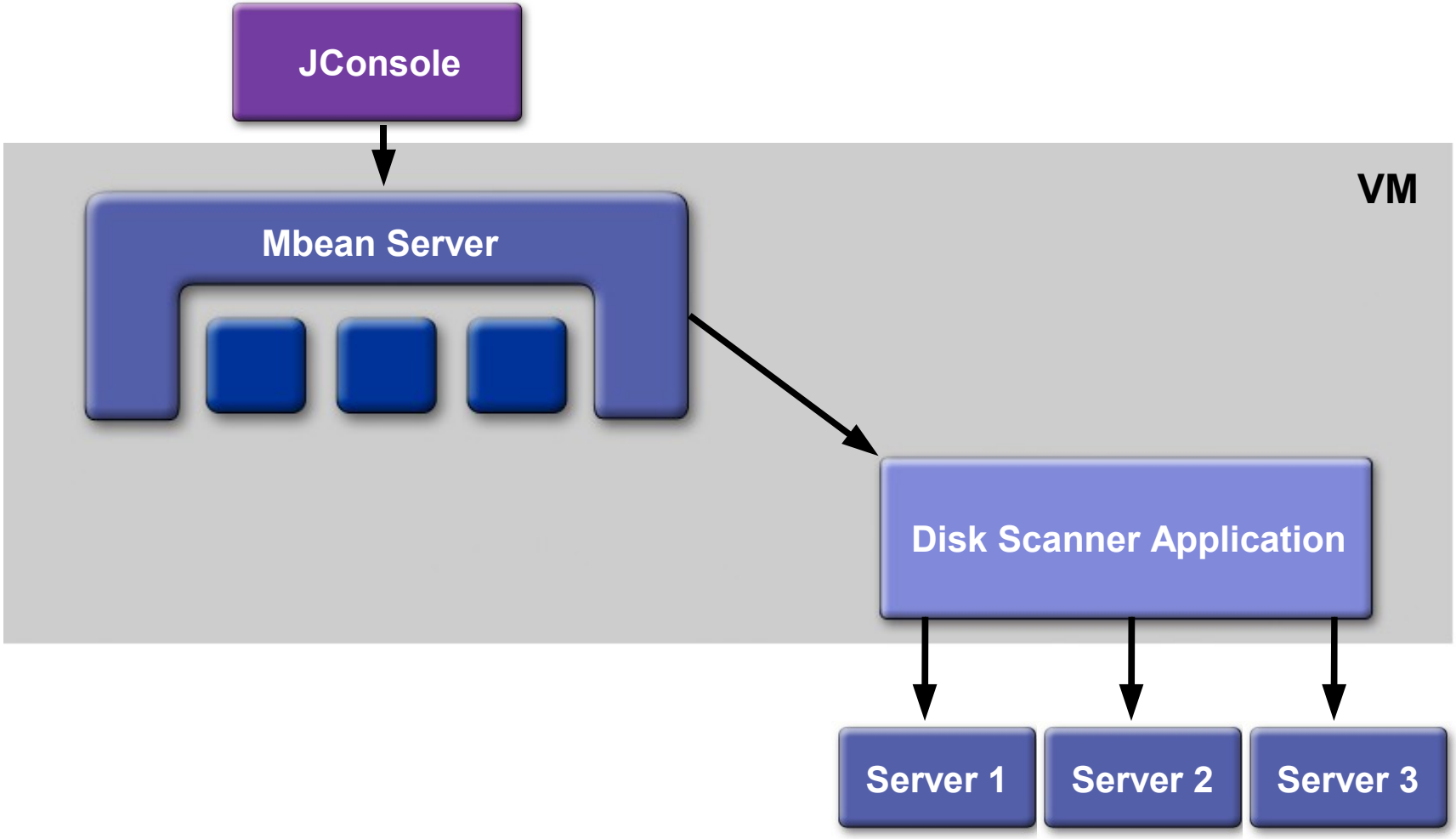
Linking MBeans to Information

JMX API Changes Coming Soon

JMX Web-Services Connector

JMX API Changes Coming Later

Scanner Application Architecture



DEMO

Monitoring With jconsole

Agenda

Demo

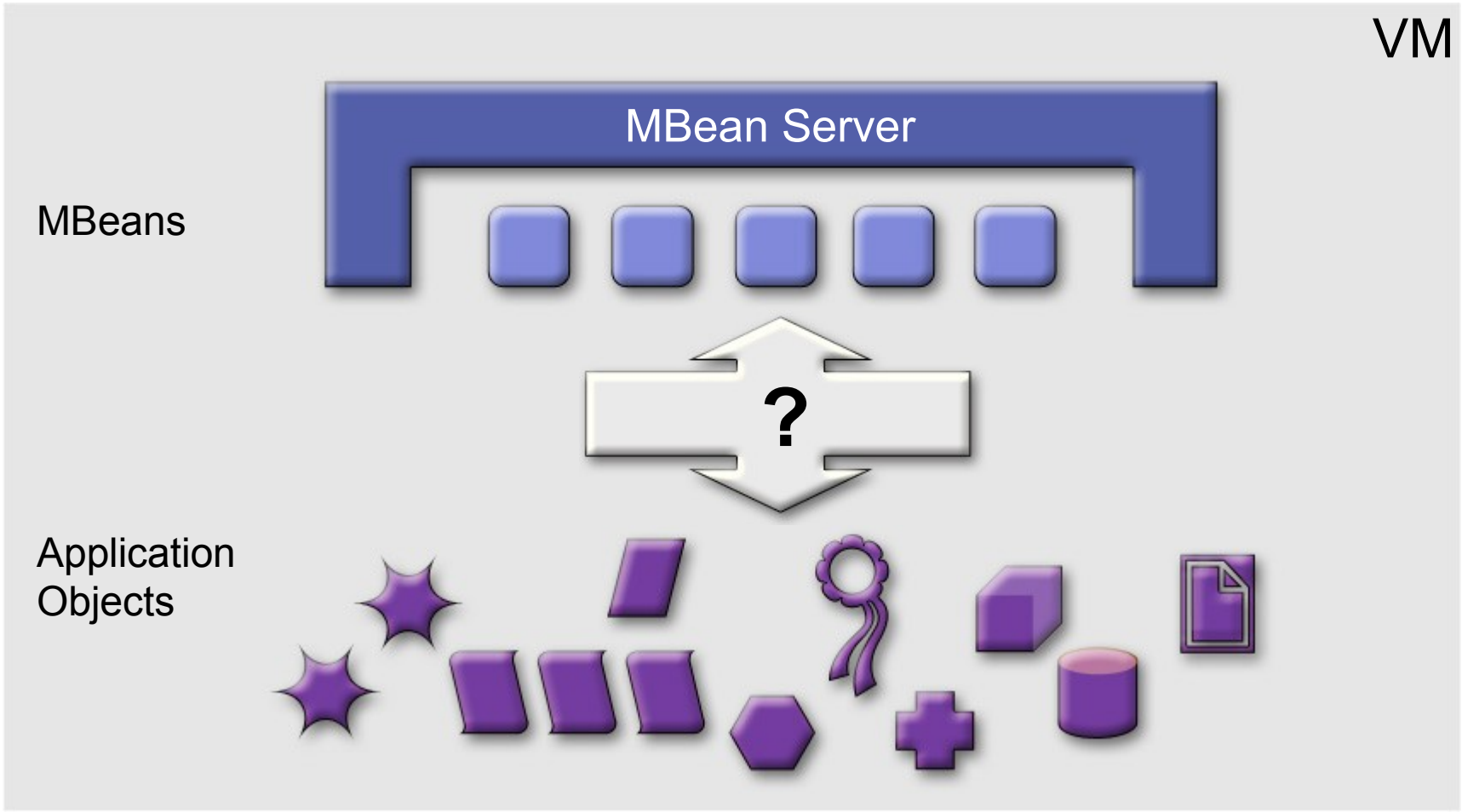
Linking MBeans to Information

JMX API Changes Coming Soon

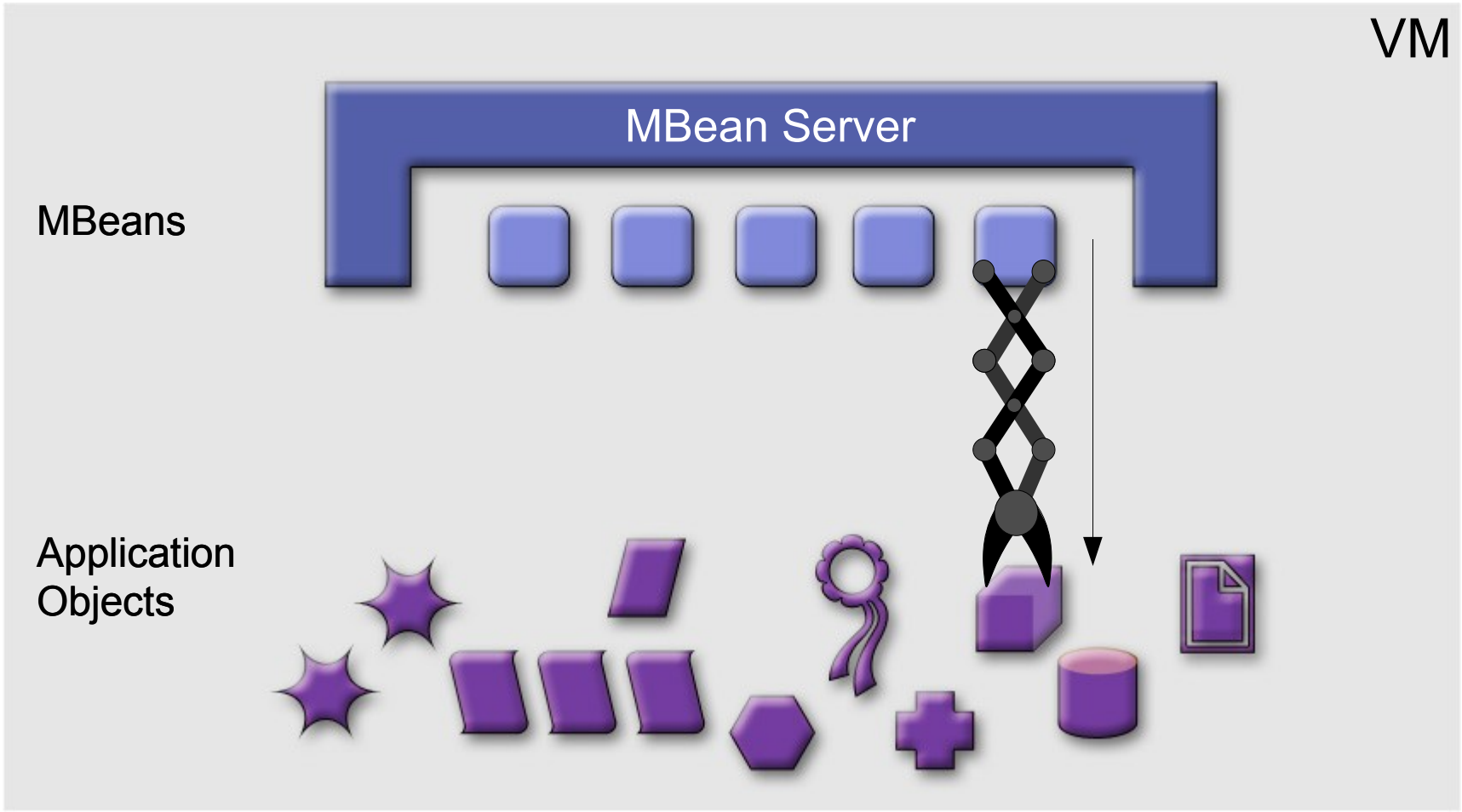
JMX Web-Services Connector

JMX API Changes Coming Later

Linking MBeans to Information



Method 1: Information Grabbing



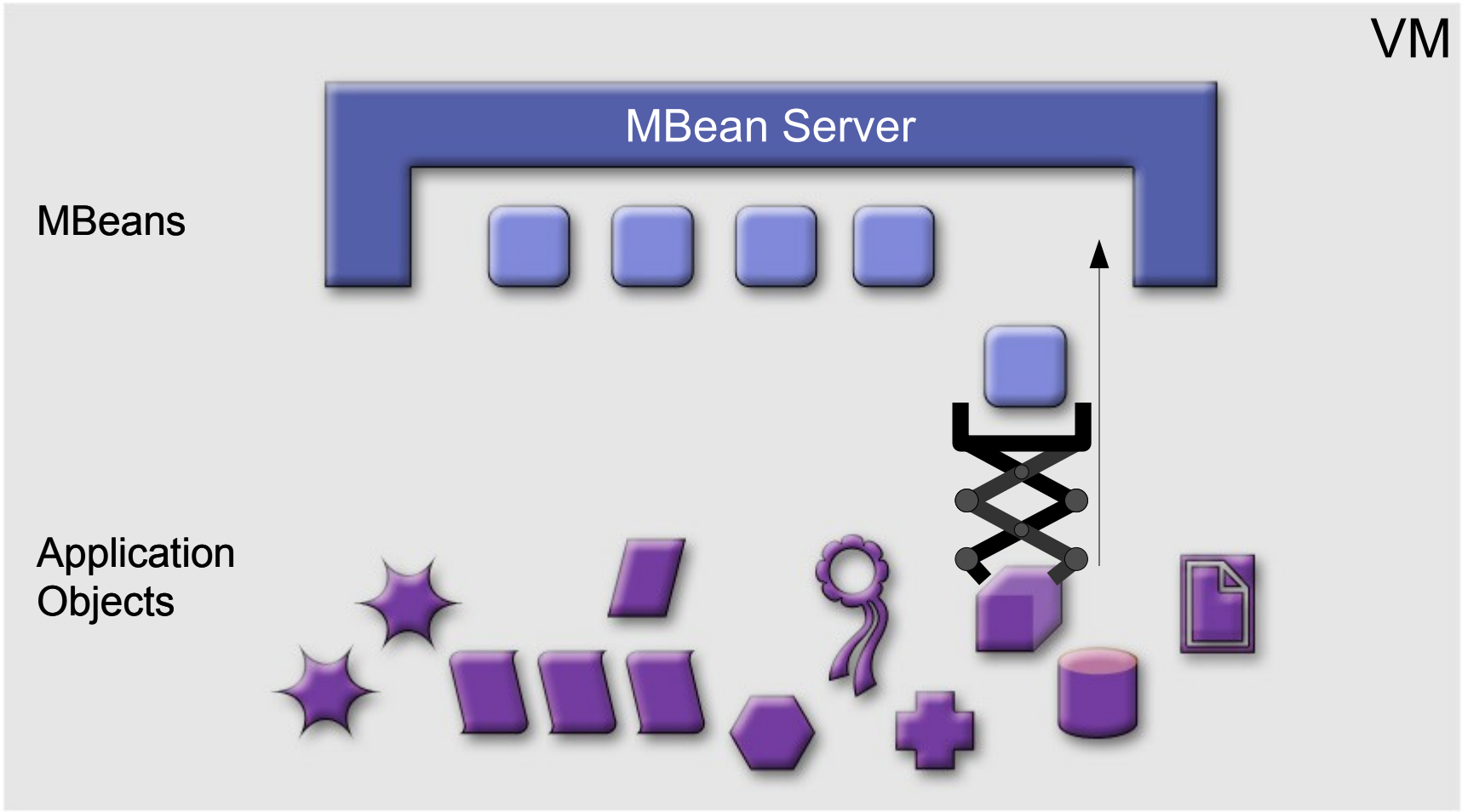
Method 1: Information Grabbing

- MBean consults global fields or methods
- e.g., AWT Event MBean:

```
Toolkit.getDefaultToolkit().addAWTEventListener(...)
```

- Simplest technique, but violates modularity
  - If the MBean can see the objects it needs, who else can?
  - What if we wanted one AWT Event MBean for every window?
- Nevertheless, often the best approach

Method 2: Information Shoving



Method 2: Information Shoving

- Object creates its own MBean

```
public class Cache {
  public Cache(int size) {
    this.size = size;
    Object mbean = new CacheManager(this);
    ObjectName name =
      new ObjectName(":type=CacheManager");
    MBeanServer mbs =
      ManagementFactory.getPlatformMBeanServer();
    mbs.registerMBean(mbean, name);
  }
  ...
}
```

```
JMXBridge.registerMe(this);
```

Method 2: Information Shoving

- Classes polluted with explicit code to publish information
  - Might be just one or two lines
- How do we find the MBeanServer to use?

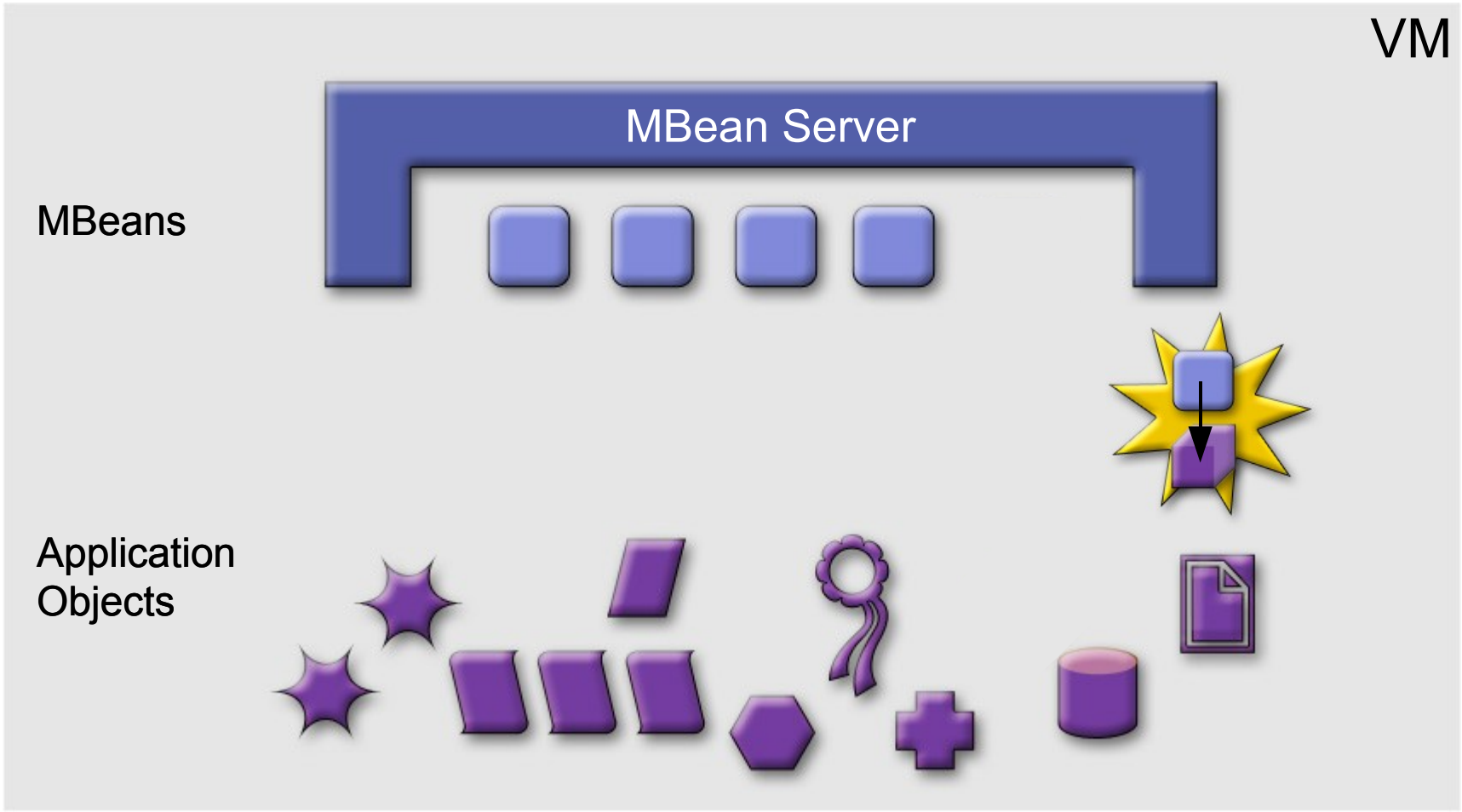
```
ManagementFactory.getPlatformMBeanServer();
```

- How do we know what ObjectName to use?

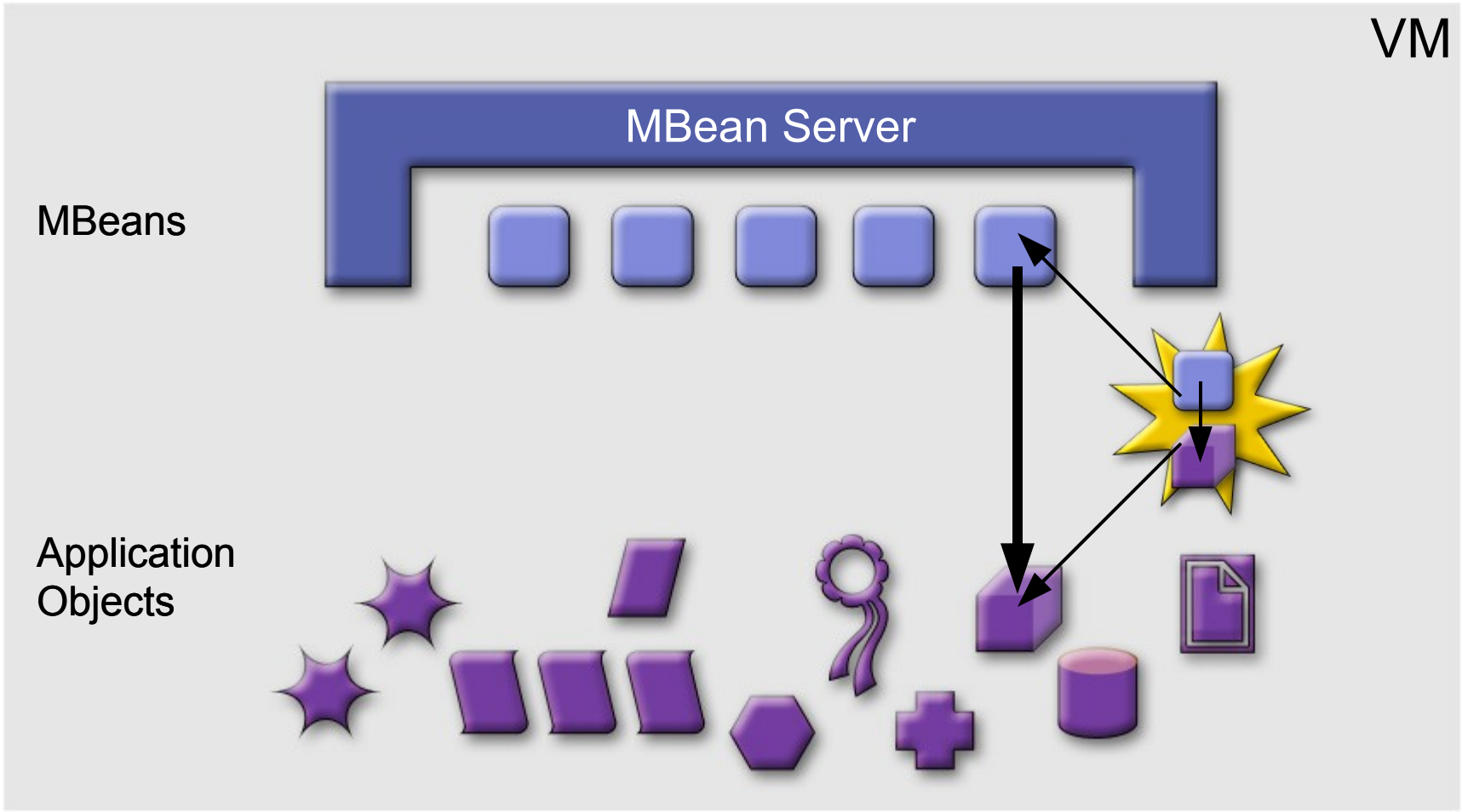
```
new ObjectName(":type=CacheManager");
```

- Giving out a reference to this from a constructor can lead to data races

Method 3: Quantum Entanglement



Method 3: Quantum Entanglement



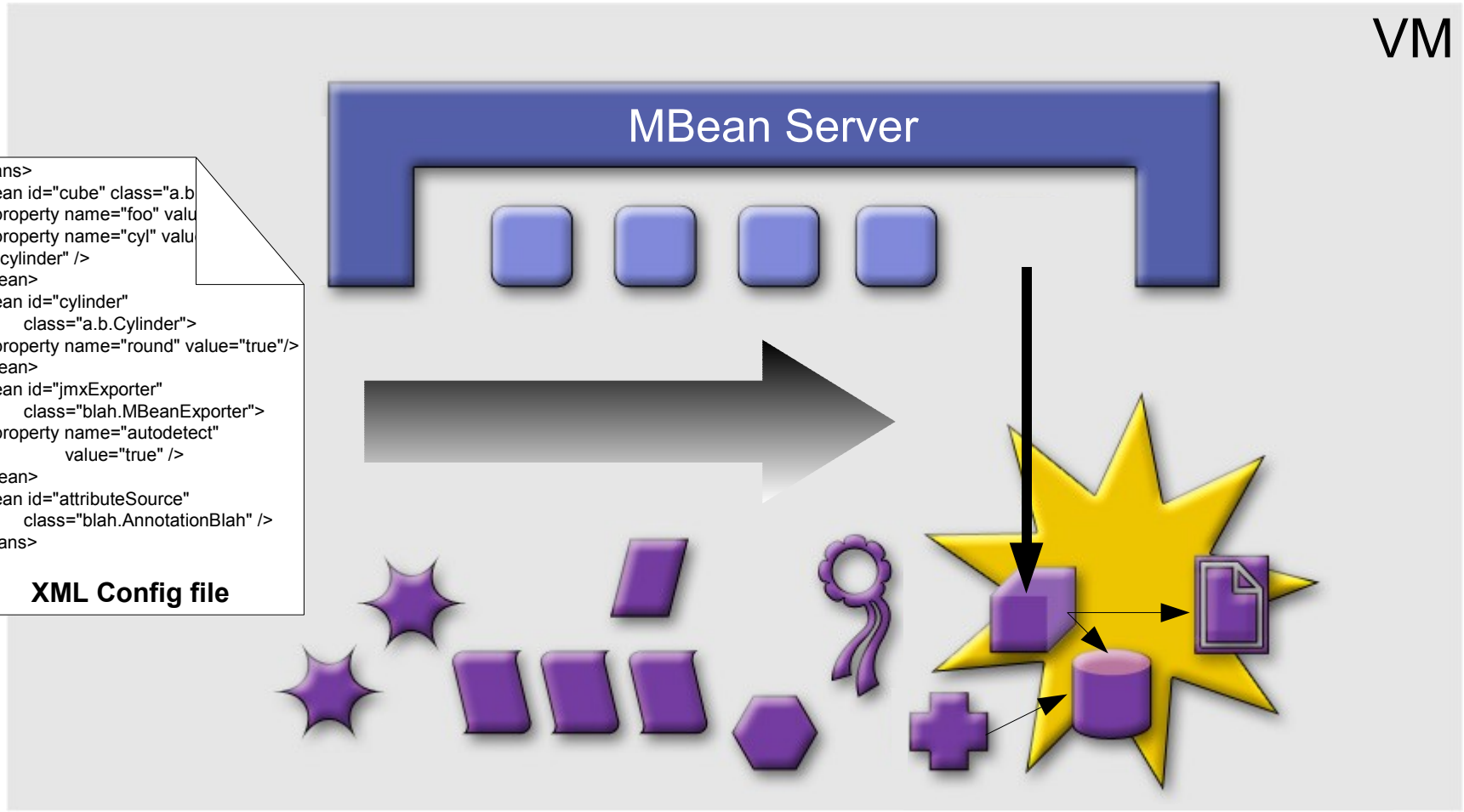
Method 3: Quantum Entanglement

- Creator code constructs object and its MBean

```
Cache cache = new Cache();
Object mbean = new CacheManager(cache);
ObjectName name =
  new ObjectName(":type=CacheManager");
mbeanServer.registerMBean(mbean, name);
```

- Can create many objects and their MBeans together
  - Centralizes JMX API “pollution”
  - Instrumented objects don’t see JMX API

Method 4: Dependency Injection



Method 4: Dependency Injection

- Obtain objects by asking container rather than with new Foo(...)
- Dependencies (references) between objects resolved by container
- Object classes and dependencies typically expressed in XML config files
- Cache depends on MBean Server and ObjectName
- Or MBean depends on Cache

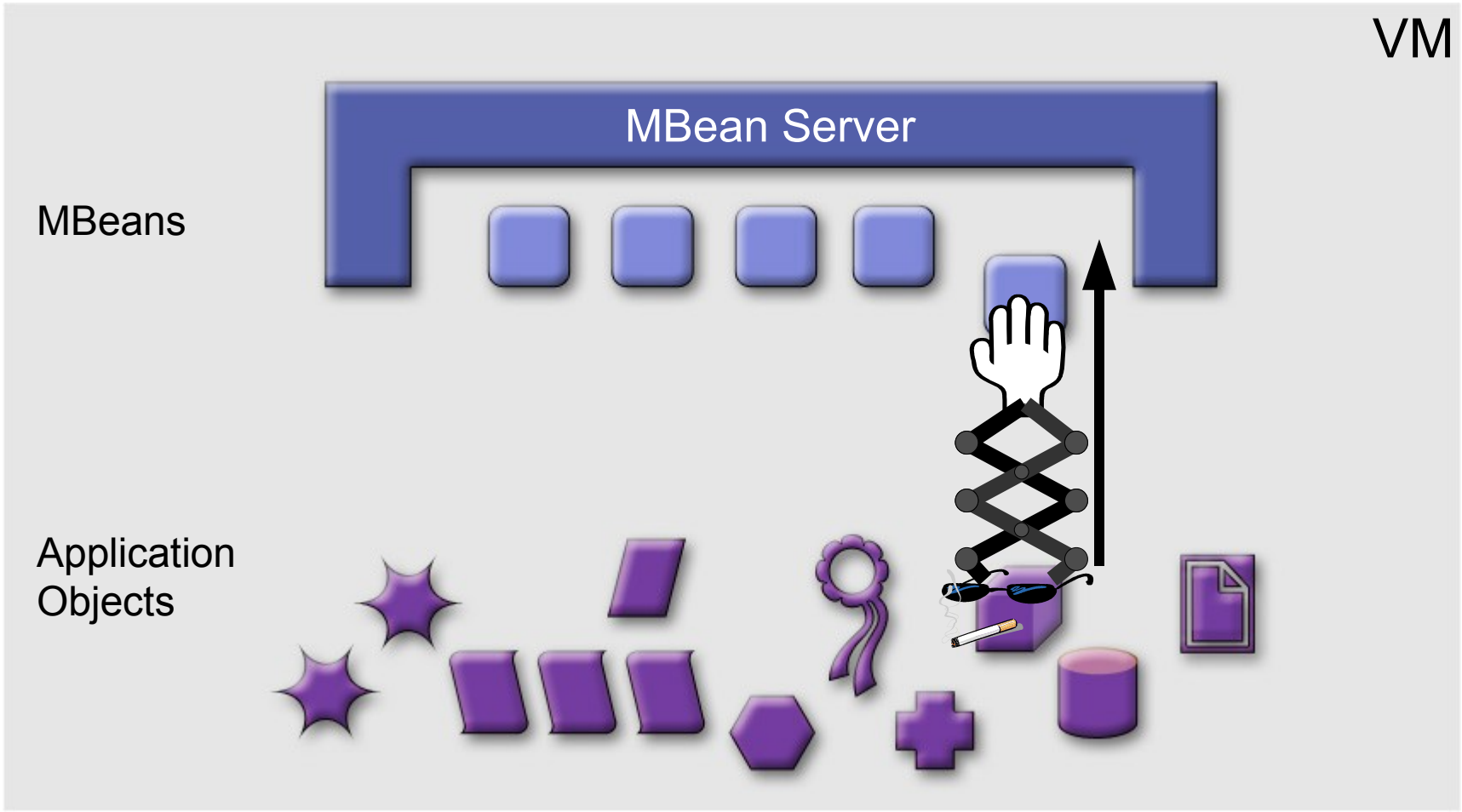
Spring Can Conjure Up MBeans

```
@ManagedResource
public class Cache {
   ...
   @ManagedAttribute(description = "Size in bytes")
   public int getSize() {...}
}
```

```
<bean name="domain:type=Cache"
      class="com.example.cache.Cache">
   ...
</bean>
```

- Spring creates the cache object and fabricates an MBean `"domain:type=Cache"` that exports some of the object's methods

Method 5: AOP



Method 5: AOP

```
public class Cache {
   public Cache(int size) {
      this.size = size;

   }
}
```

```
Object mbean = new CacheManager(this);
ObjectName name =
  new ObjectName(":type=CacheManager");
MBeanServer mbs =
  ManagementFactory.getPlatformMBeanServer();
mbs.registerMBean(mbean, name);
```

Method 5: AOP

- Class is modified “from outside” to add JMX API instrumentation
  - Details like ObjectName and MBeanServer are stored in this outside aspect
- Works well for monitoring (e.g. statistics)
  - http://glassbox-inspector.dev.java.net/
- Somewhat questionable for management
  - e.g., configuration changes
  - What you see is not what you get

Agenda

Demo

Linking MBeans to Information

JMX API Changes Coming Soon

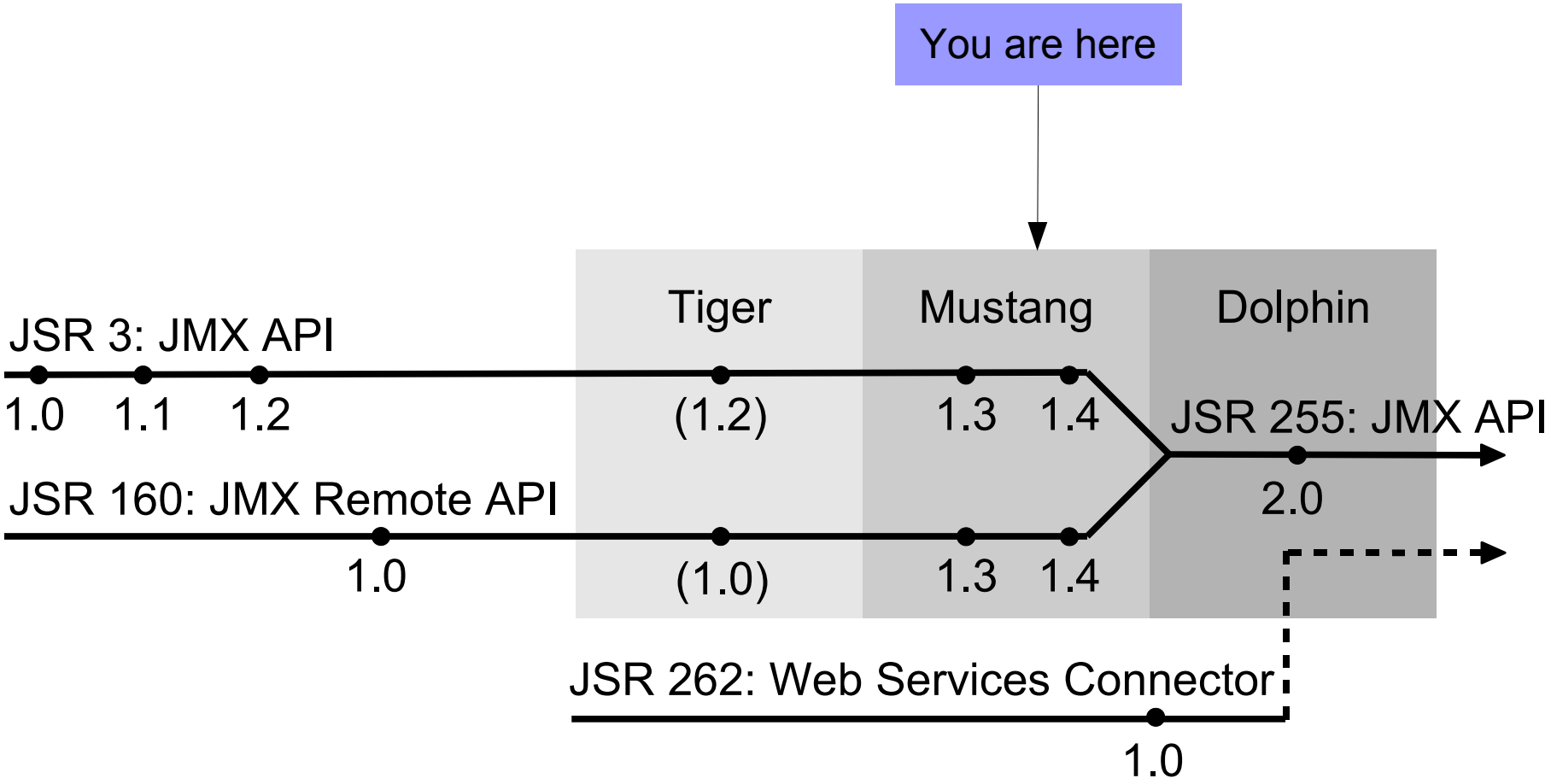
JMX Web-Services Connector

JMX API Changes Coming Later

JMX API Versions

- JSR 3 defined the JMX API
  - Updated in Java Platform, Standard Edition 6 (code-named Mustang)
- JSR 160 defined the JMX Remote API
  - Also updated in Mustang
- JSR 255 merges and updates JSRs 3 and 160
  - It will produce JMX API version 2.0 in Java SE 7 (code-named Dolphin)

JMX API Versions



http://mustang.dev.java.net

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();
  ...
}
```

- When values must be grouped automatically

MXBeans: Problem Statement (2)

- An MBean interface can include arbitrary Java programming language types

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

- 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)

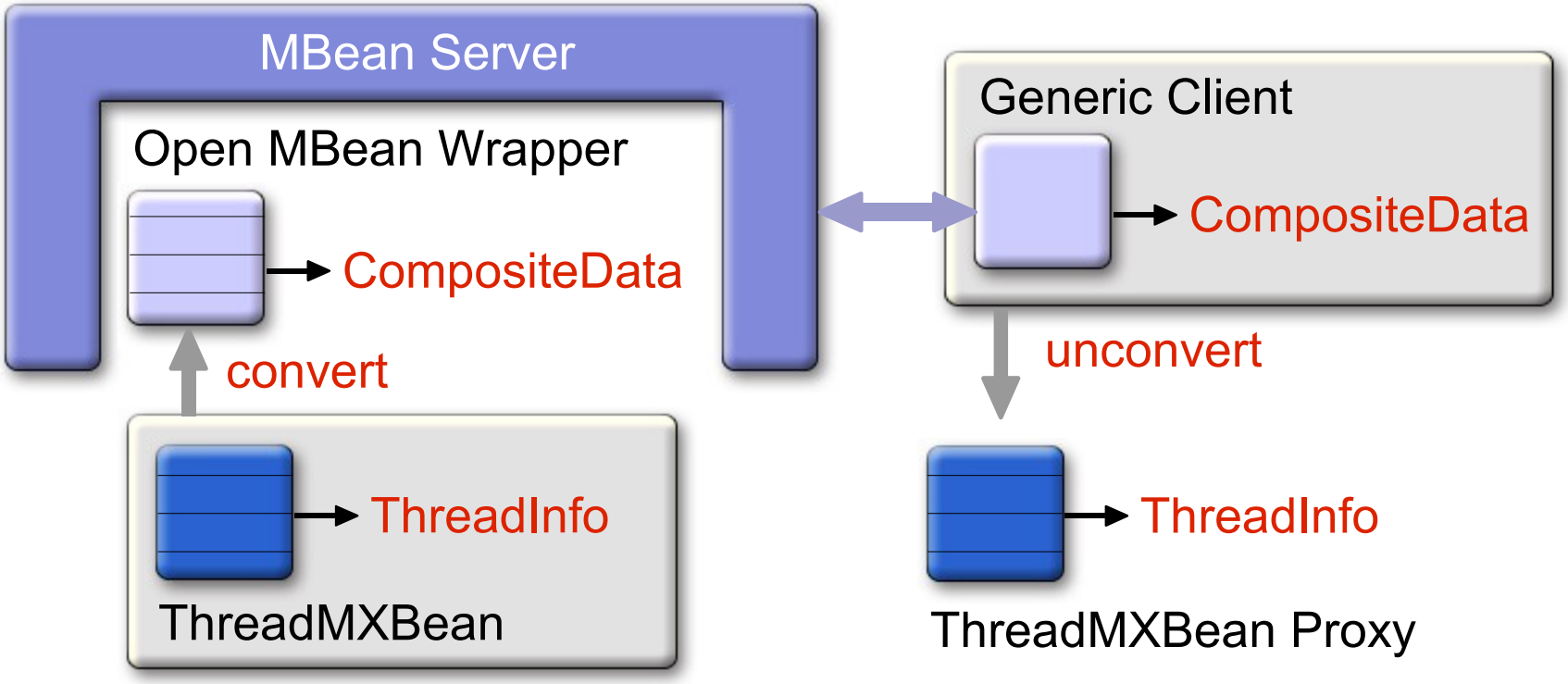
- MXBeans were designed for the instrumentation of the VM itself (JSR 174)
  - Already exist in java.lang.management
  - User-defined MXBeans are new in Mustang
- Management interface still a bean interface
- Can reference arbitrary types, with some restrictions
- JMX API wraps an instance of this interface 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);
```

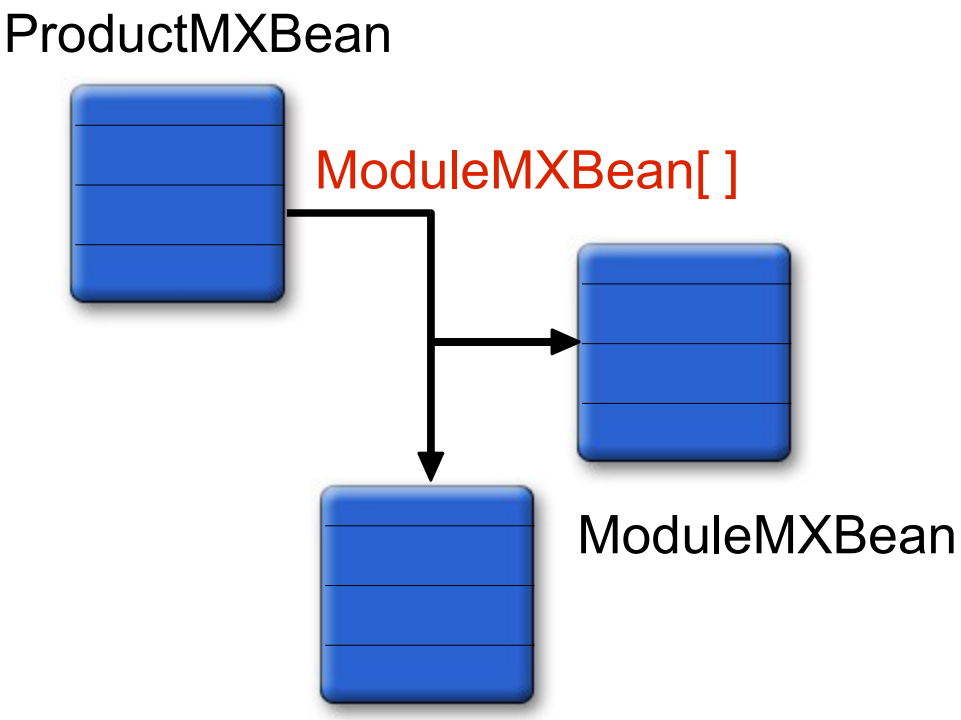
MXBeans (3)

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

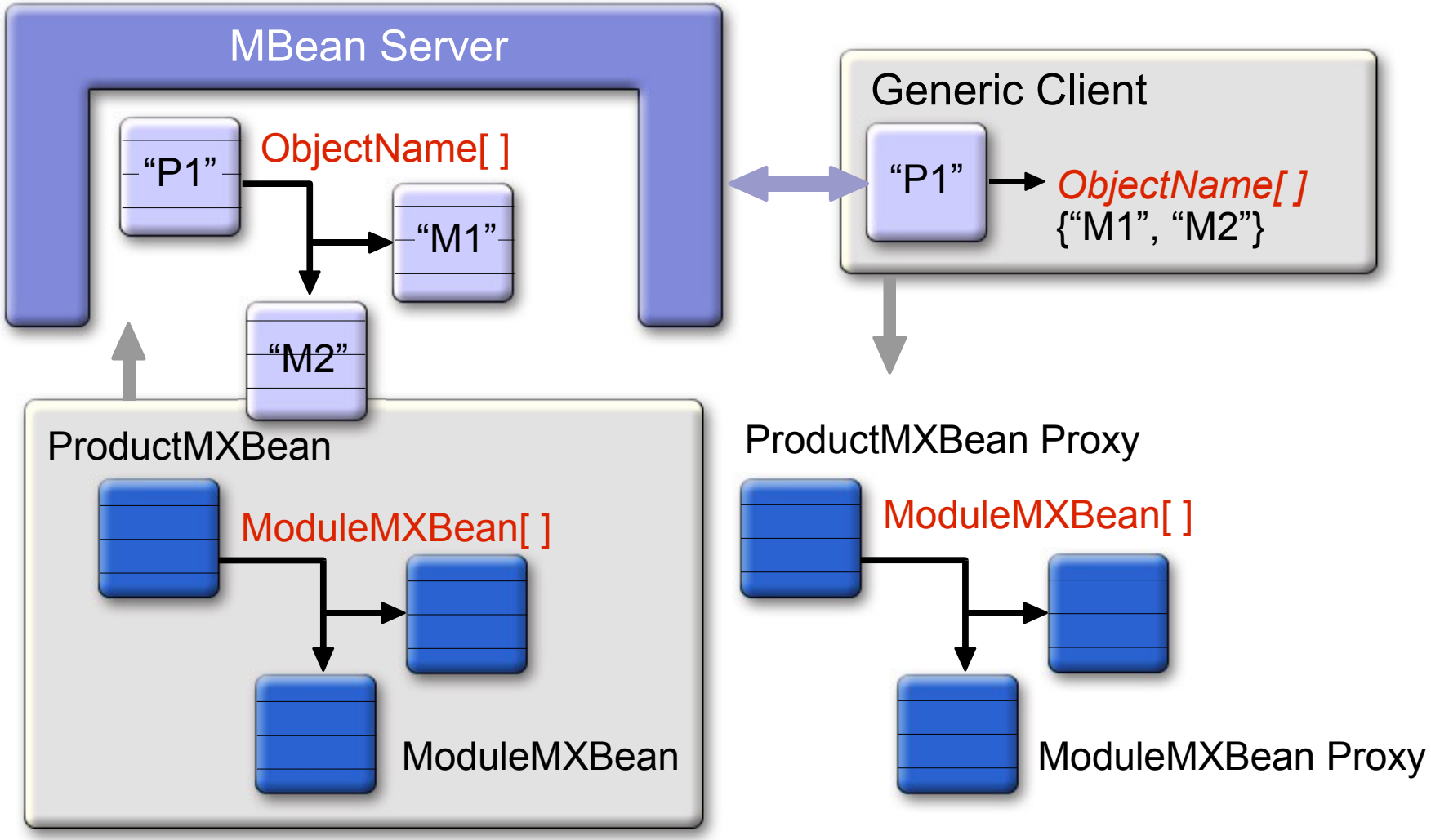


MXBean References (1)

```
public interface ProductMXBean {
    ModuleMXBean[] getModules();
}
```

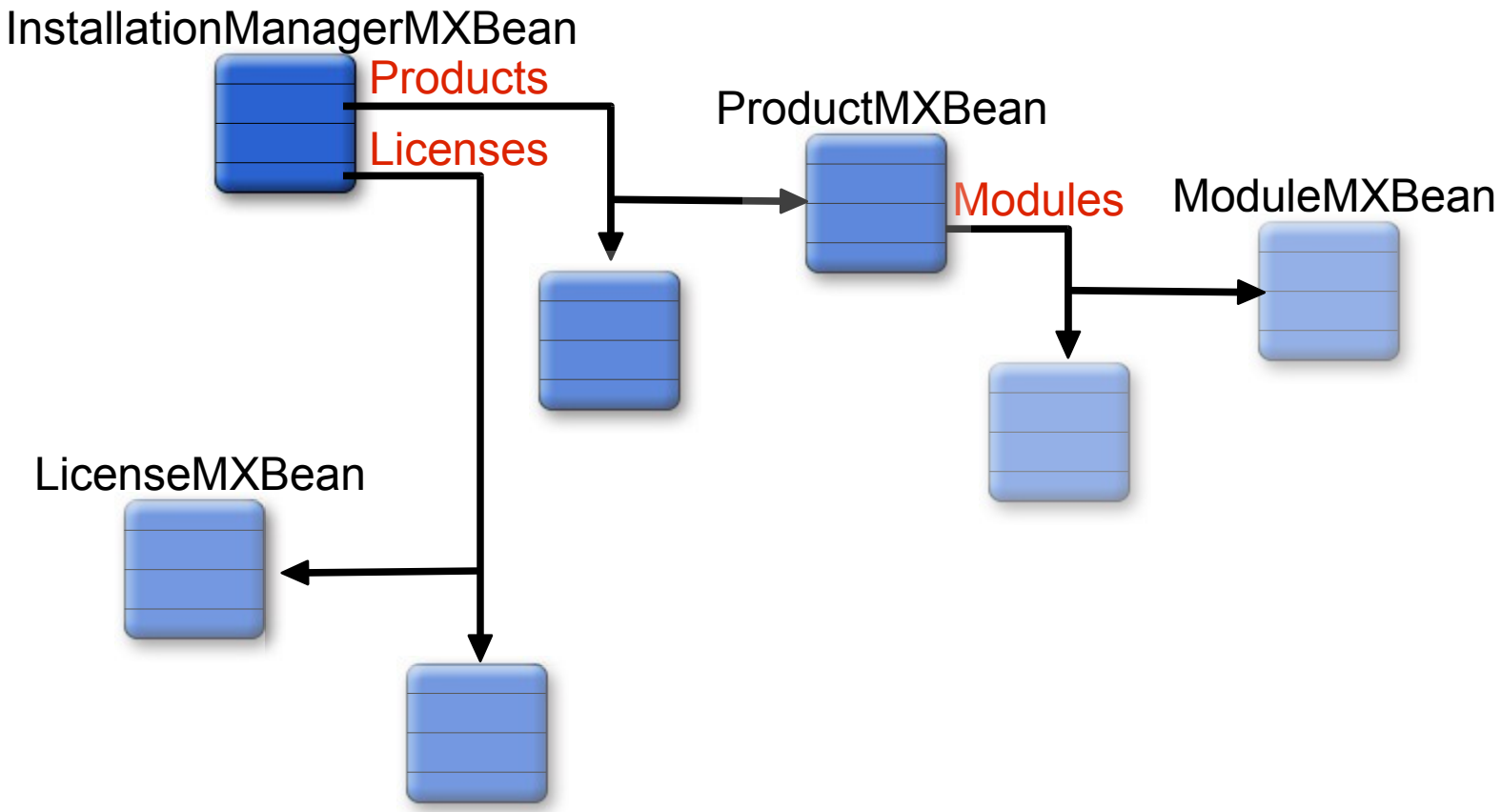


MXBean References (2)

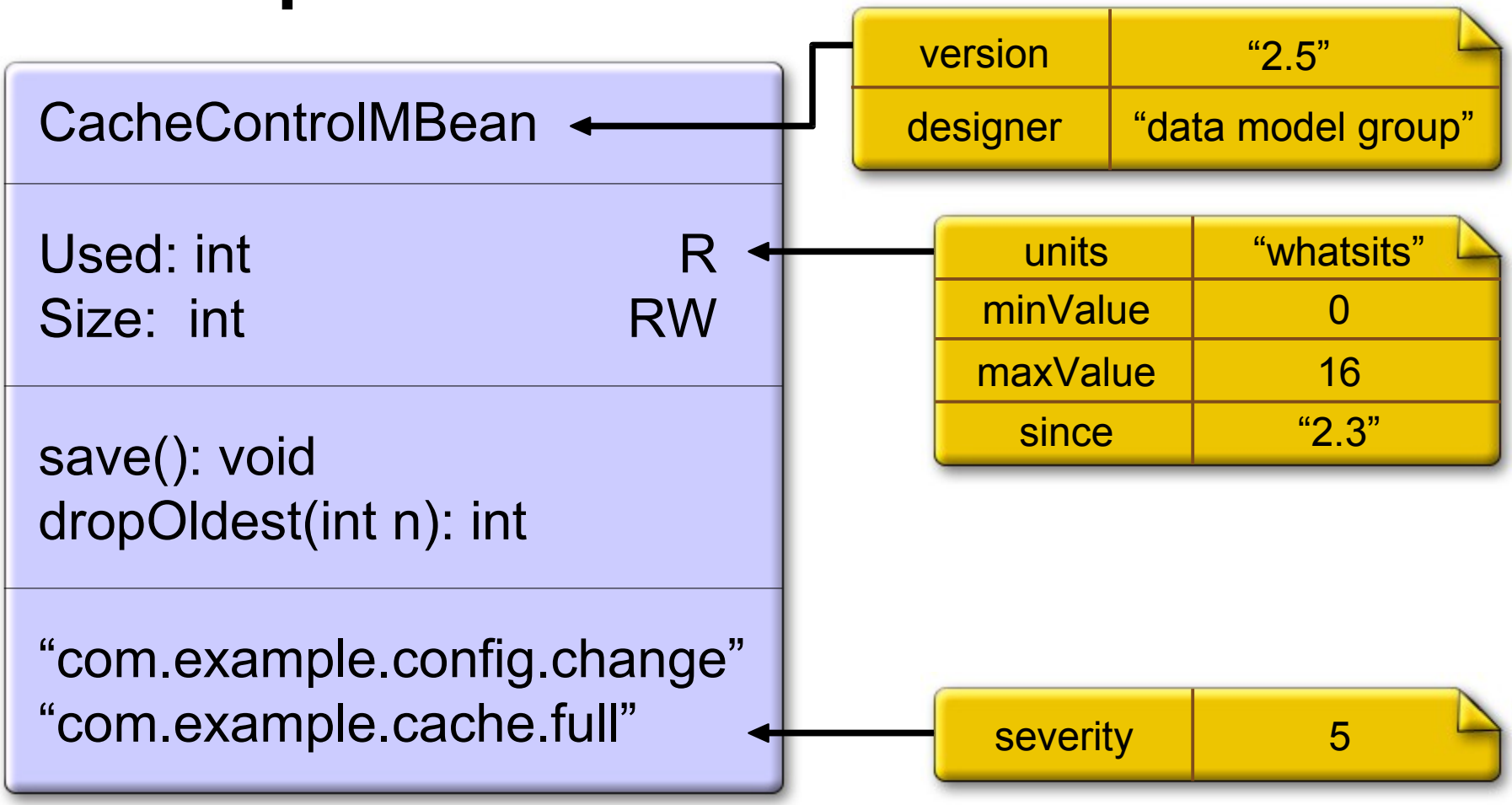


MXBean References (3)

Navigating From a Starting Point



Descriptors

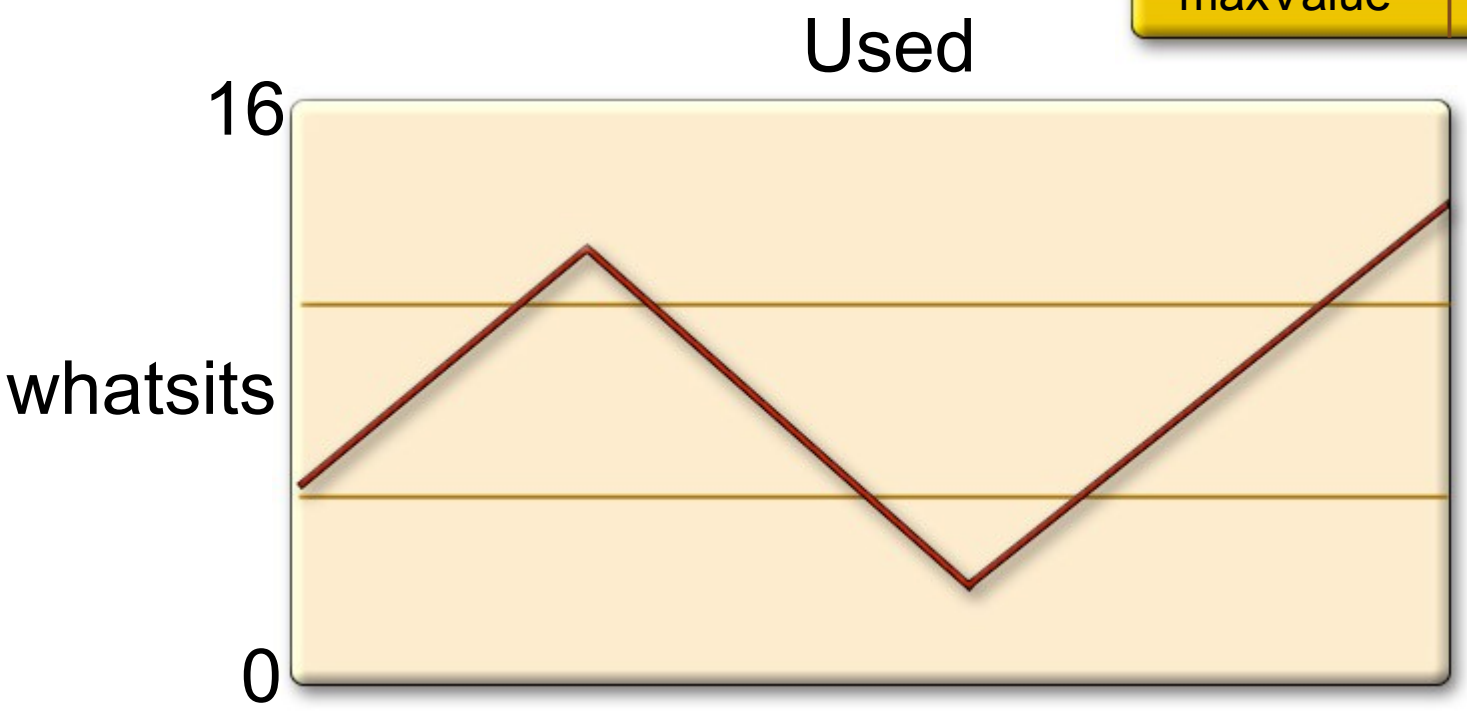


Descriptors and Generic Clients

(like jconsole)

Used: int R

units	“whatsits”
minValue	0
maxValue	16

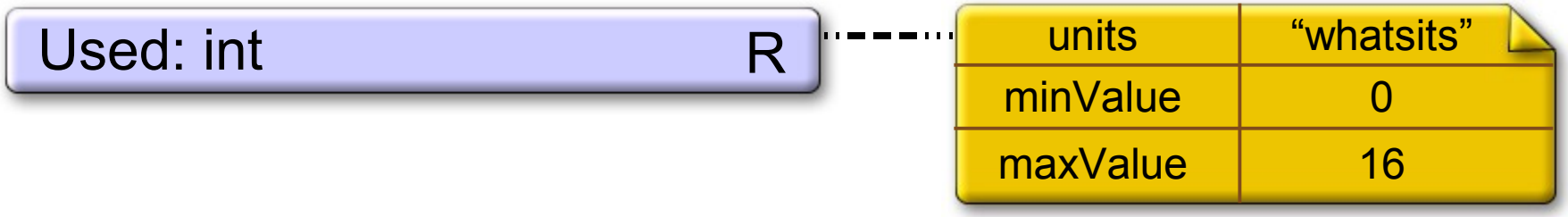


Descriptor Details

- 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

Descriptor Annotations

```
public interface CacheControlMBean {
    @Units("whatsits") @Range(minValue=0, maxValue=16)
    public int getUsed();
}
```



- With definitions like:

```
public @interface Range {
    @DescriptorKey("minValue")
    public int minValue();
    @DescriptorKey("maxValue")
    public int maxValue();
}
```

Some Other Mustang changes

- Generified at last!
  - Set<ObjectName> queryNames(...)
- More-general ObjectName wildcards
  - domain:type=Dir,path="/root/*"
- Simpler Notification use
  - NotificationBroadcasterSupport(MBeanNotificationInfo[])
  - class StandardEmitterMBean extends StandardMBean
- Monitor attributes of complex type
  - MonitorMBean.setObservedAttribute("ThreadInfo.size")

http://mustang.dev.java.net

Agenda

Demo

Linking MBeans to Information

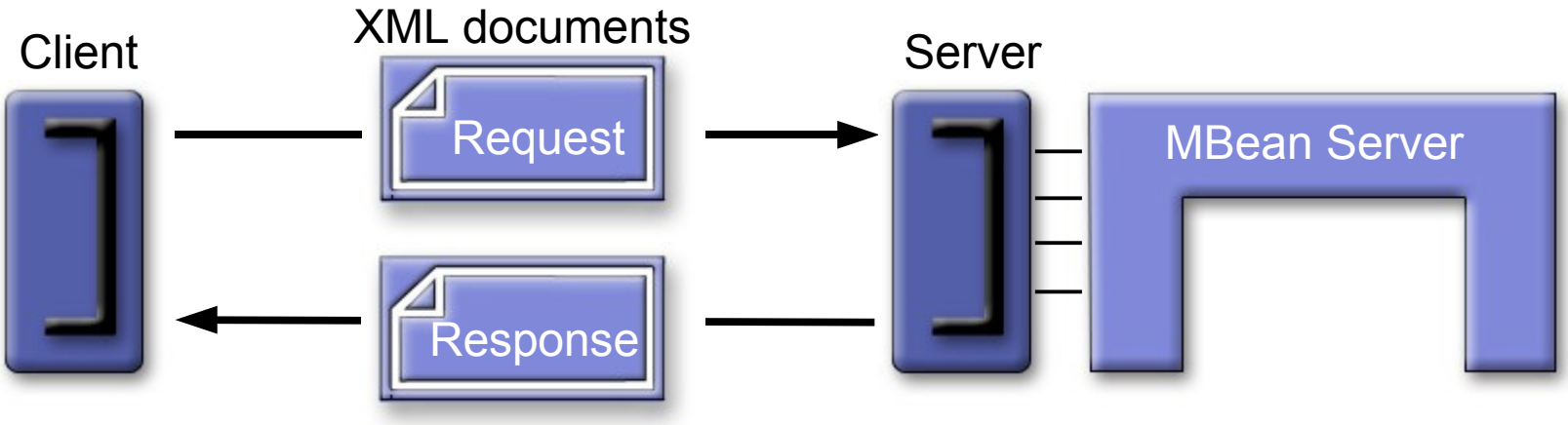
JMX API Changes Coming Soon

JMX Web-Services Connector

JMX API Changes Coming Later

Web-Services Connector

- 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



Web-Service Connector Details

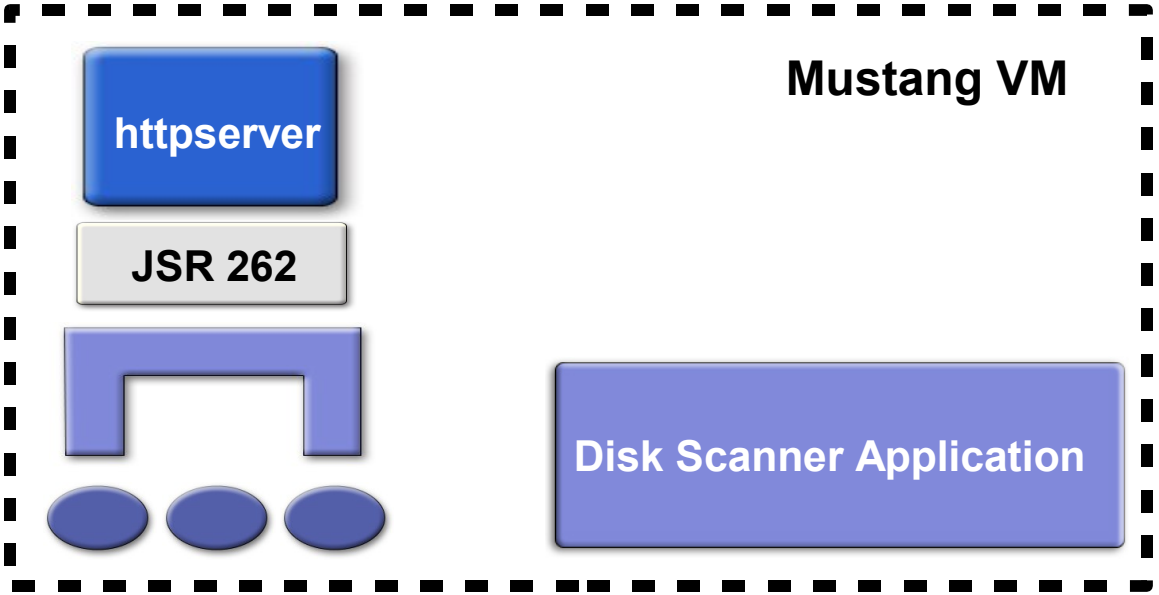
- JMX Remote API compliant connector
  - service:jmx:**ws**://<host>:<port>/<http context>
- JAX-WS 2.0 Endpoint
- SOAP 1.1 binding / HTTP transport
- Interoperable with non JMX client
  - WSDL at http://<host>:<port>/<http context>?WSDL
  - JavaScript, C, C#, perl, ...
- Basic Authentication and SSL

AJAX/JSR 262 Demo

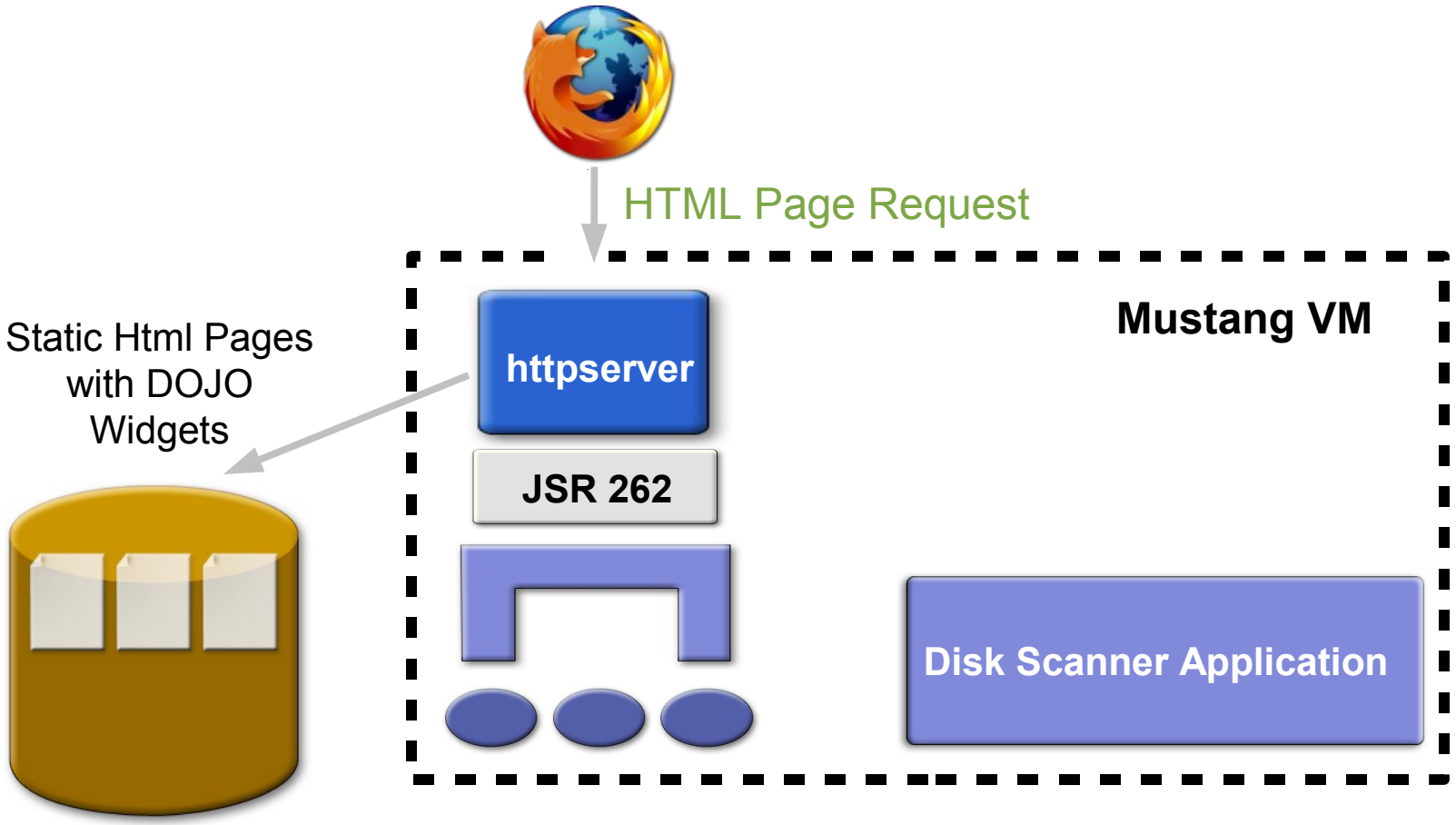
- Web-based management console
  - Non Java client
  - Deployed in Mustang HTTP Server
  - Could have been deployed in any JAX-WS container
- Prototype of AJAX Dojo widgets
  - Display MBean Attributes Table
  - Invoke MBean Operation
- Prototype of a NetBeans support

AJAX/JSR 262 Demo Architecture

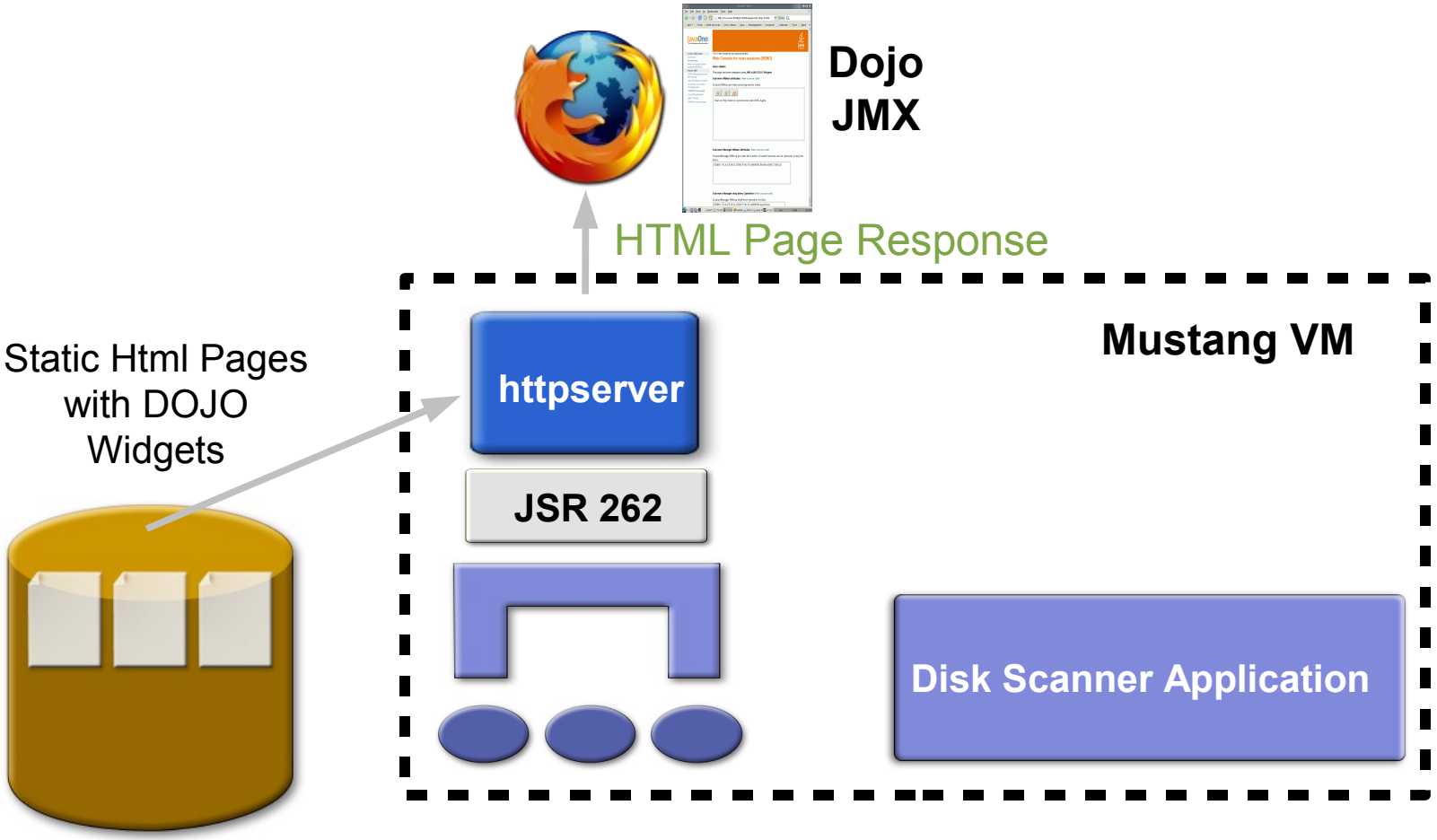
Static Html Pages with DOJO Widgets



AJAX/JSR 262 Demo Architecture



AJAX/JSR 262 Demo Architecture

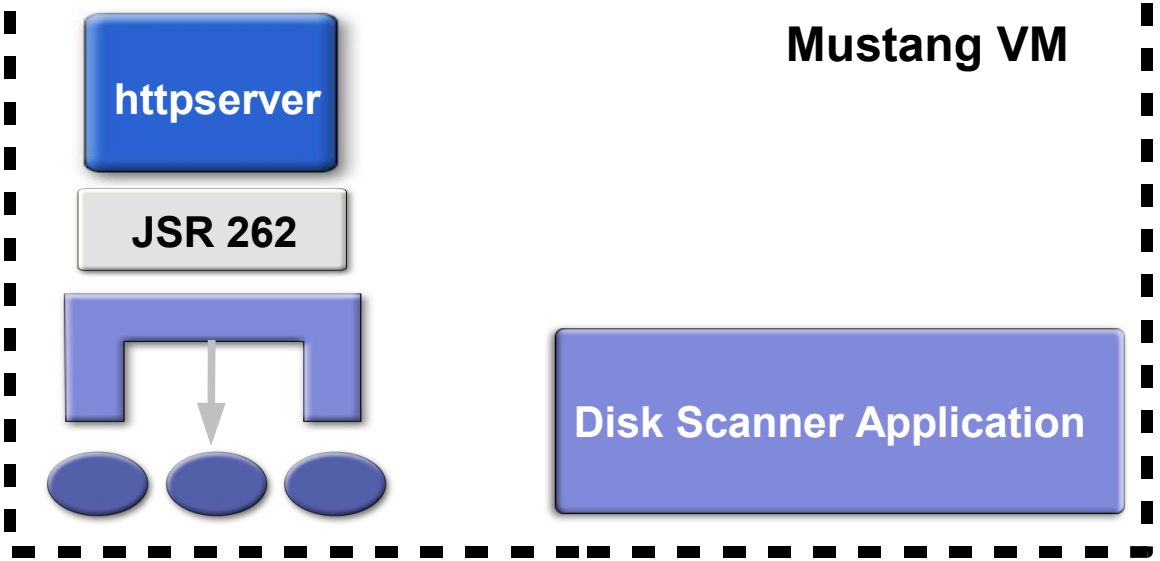


AJAX/JSR 262 Demo Architecture

Dojo
JMX

JavaScript Sends jsr262 XML Requests

Static Html Pages
with DOJO
Widgets



AJAX/JSR 262 Demo Architecture

Dojo
JMX

JavaScript Receives jsr262 XML Responses

Static Html Pages
with DOJO
Widgets

Mustang VM

httpserver

JSR 262

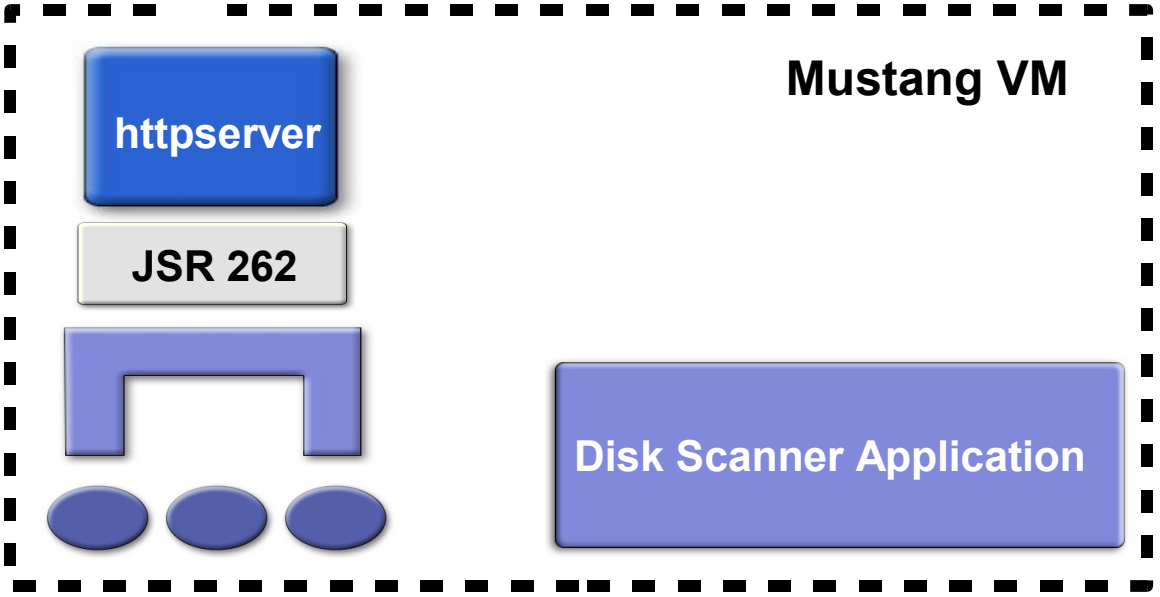
Disk Scanner Application

AJAX/JSR 262 Demo Architecture

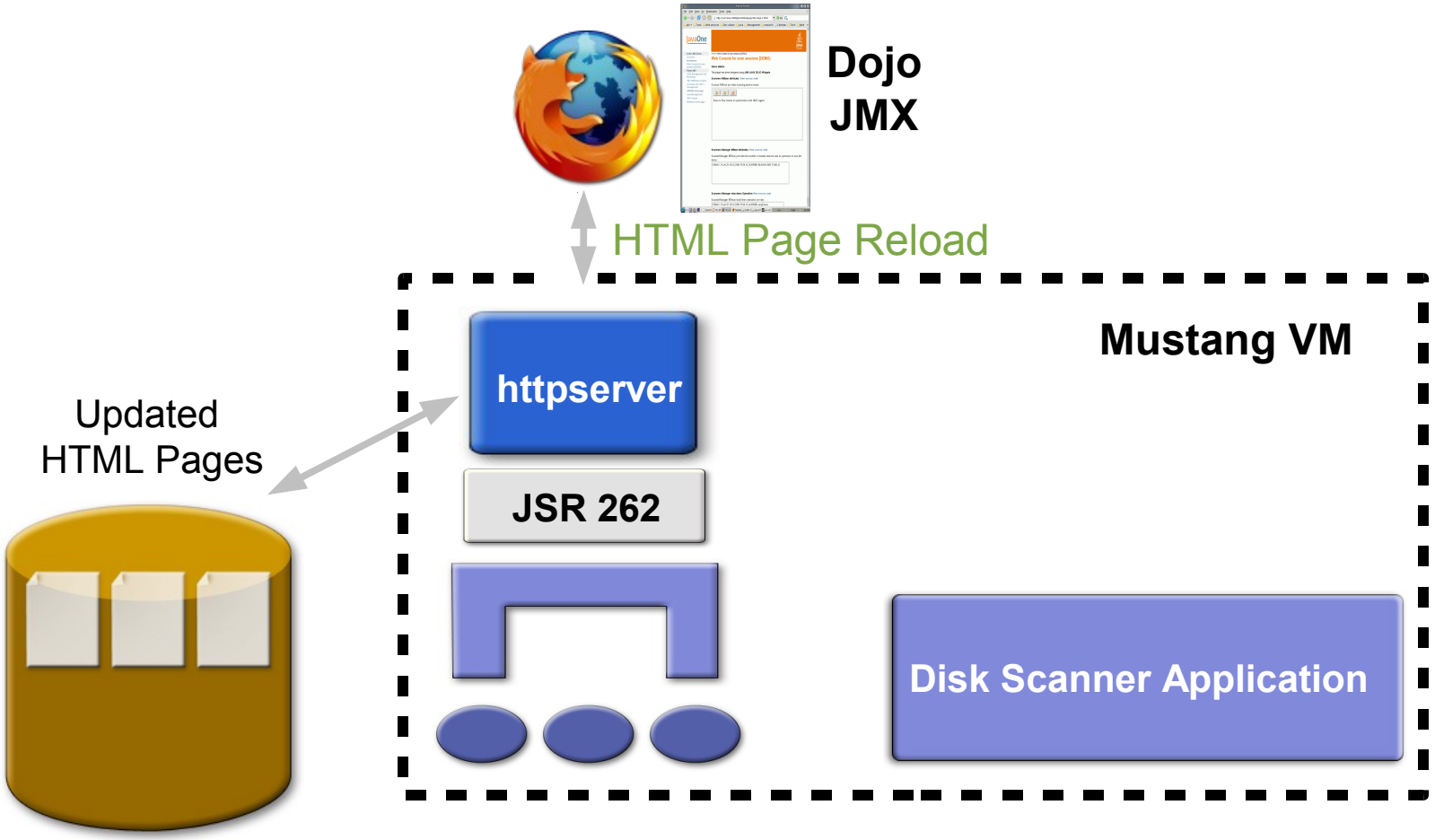
NetBeans 5.0

Dojo
JMX

Add DOJO
Widgets



AJAX/JSR 262 Demo Architecture



DEMO

AJAX/Web-Services Access

JSR 262 Summary and Opportunities

- JMX™ Remote API Connector
  - Existing client such as JConsole
- Allows new actors to enter the JMX scene
- Integrates naturally with AJAX
- Building block for WSDM and WS-MAN convergence?
- http://ws-jmx-connector.dev.java.net/
  - Early specification draft, sample code and RI

Agenda

Demo

Linking MBeans to Information

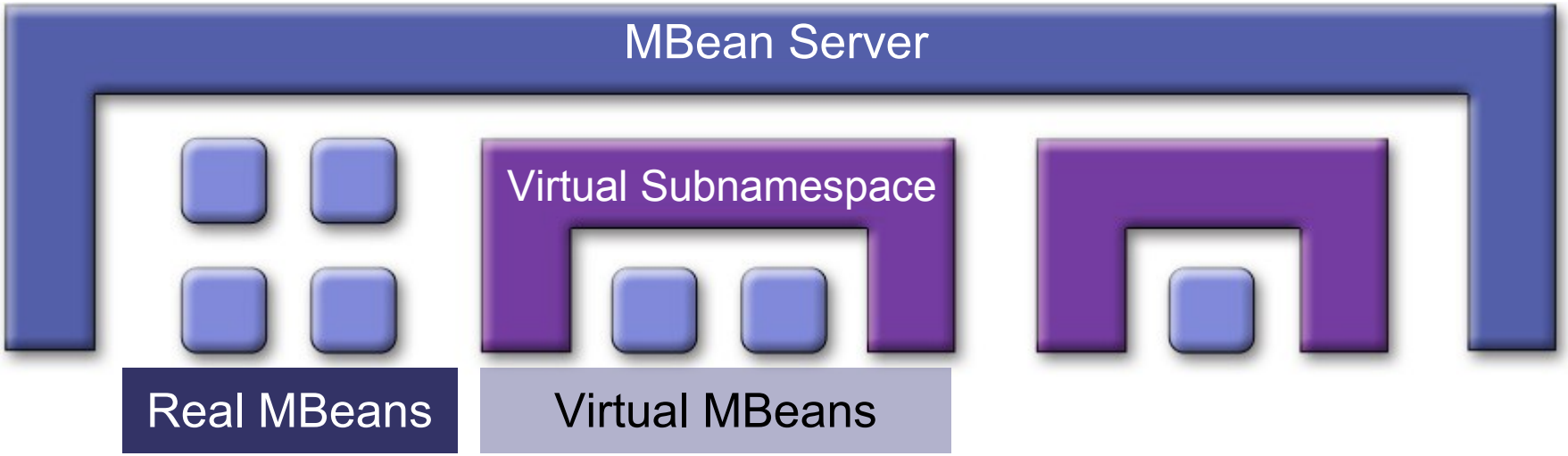
JMX API Changes Coming Soon

JMX Web-Services Connector

JMX API Changes Coming Later

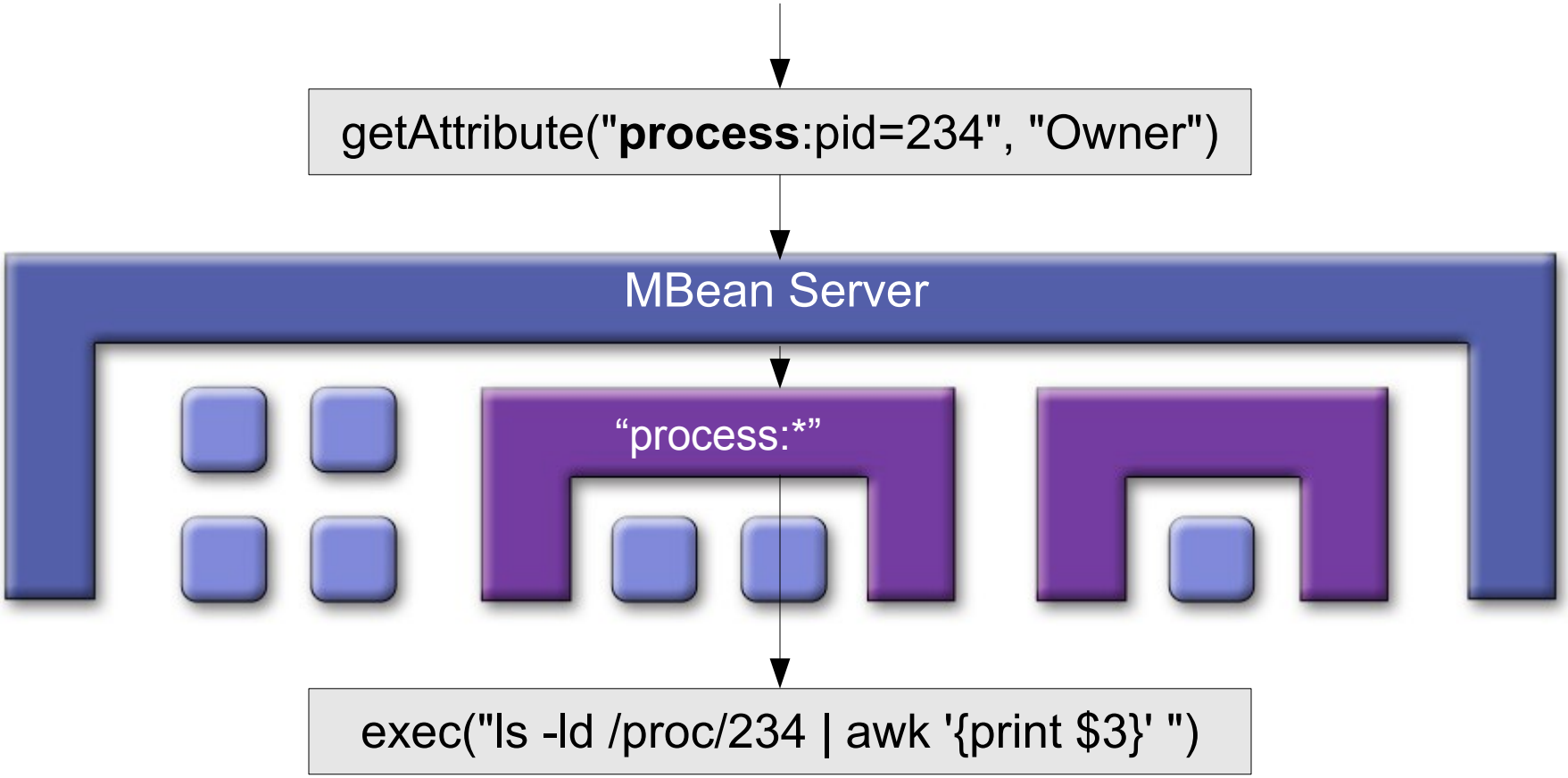
SPECULATION

Virtual MBeans



- Handler can be set for part of MBean namespace
- Accesses to names in the space directed to handler
- MBeans in that space don't have to be real objects

Virtuality Example: Processes

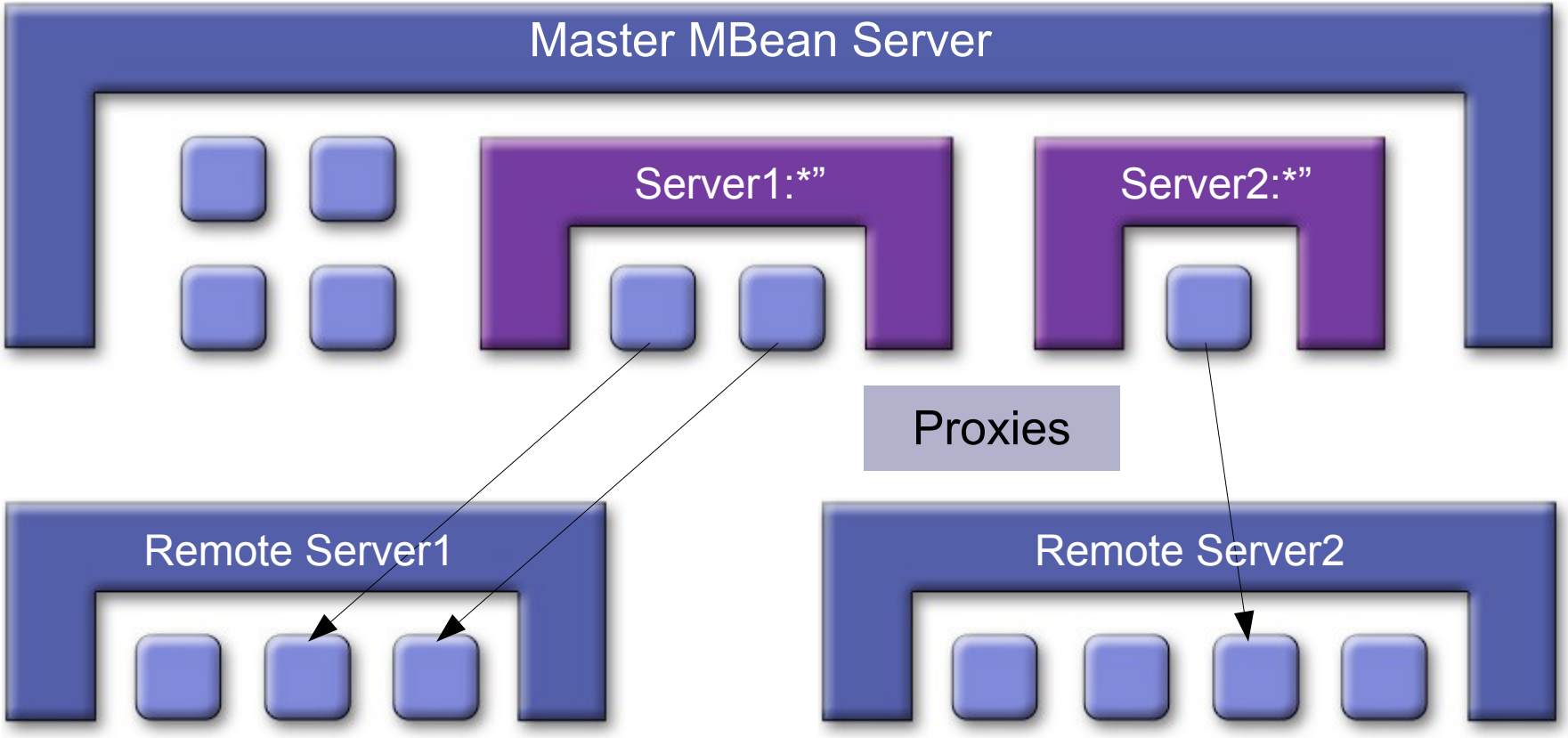


Don’t take the names literally!

Virtual MBeans: What For?

- When it is expensive to discover a set of MBeans, or to track their creation and deletion
  - MBeans are system resources and listing them means running a slow external program
  - MBeans are remote and listing them means making a network connection
- When you want to layer some additional behaviour on a set of MBeans
  - log accesses, impose security checks

Cascading


Master MBean Server
Server1:*”
Server2:*”
Proxies
Remote Server1
Remote Server2

Cascading: What For?

- Provide a single access point for all instrumentation in a distributed system
- Clients only have to know how to reach the Master Server
- Example: managing a clustered app server

MBean Annotations (1)

```
@ManagedResource
public class Cache {
    ...
    @ManagedAttribute
    public int getSize() {...}
    ...
}

Cache c = ...;
ObjectName name = new
ObjectName("com.example:type=Cache");
mbeanServer.registerMBean(c, name);
```

MBean Annotations (2)

```
@ManagedResource(type="Cache")
public class Cache {
    ...
    @ObjectNameKey("name")
    public String getCacheName() {...}
    ...
}

Cache c = ...;
ObjectName name = new ObjectName("com.example:");
mbeanServer.registerMBean(c, name);
// name is "com.example:type=Cache,name=Something"
// where "Something" is what's returned by
c.getCacheName()
```

Summary

- Several ways to link MBeans to the rest of the application
- Good stuff coming up in Mustang
  - Try it out now!
- More good stuff coming up in Dolphin

For More Information

- http://java.sun.com/jmx
  - Documentation, articles, best practices
- http://weblogs.java.net/blog/emcmanus
  - Blog on JMX technology
- http://mustang.dev.java.net
  - Download snapshot or Beta

For More Information

- Session TS-1956, “Designing Manageable J2EE™ Apps with JMX Technology”
- BOF-0442, “Using JConsole to Monitor and Manage Your Application”
- Lab HOL-5120, “Application Monitoring and Management with NetBeans™ 5.0”
- Lab HOL-1510, “Explore J2SE™ 6.0 (Mustang) Features”
- Peabody ask-the-experts
  - Serviceability—Thursday 12:00pm
  - JMX API—Thursday 4:00pm

Q&A

Java™ Management Extensions (JMX™) Technology Today and Tomorrow

Éamonn McManus

JMX Specification Lead
Sun Microsystems
http://java.sun.com/jmx

Jean-François Denise

JMX Technology Team
Sun Microsystems
http://java.sun.com/jmx

TS-3523