Developing Manageable Component-Based Systems Using JMX and the Spring Framework

Vladimir Vivien
Software Engineer

Objective

Explore design strategies for implementing manageable component-based systems using JMX and the Spring Framework

Agenda

- Components and Containers
- Introducing JMX
- Introducing Spring
- Spring and JMX
- Designing for Manageability
- Summary

Components
&
Component Containers

What Is A Component?

- Software Unit
- Concise Functionality
- Highly Re-Usable
- Loosely Coupled
- Controllable and Manageable

Component Characteristics

- Accomplish Discrete Tasks
- Functionally complete
- Temporally Durable
- Linkable to Others
- Invert Control to Universe
- Generate / React to Events
- Requires Operating Context

Component Containers

- Operating Environment for Components
- Component Registry Service
- Component Lookup / Discovery
- Instantiation / Life Cycle Services
- Event / Notification Model
- Data and Relationship Injection
- Inter Component Communication

Available Containers

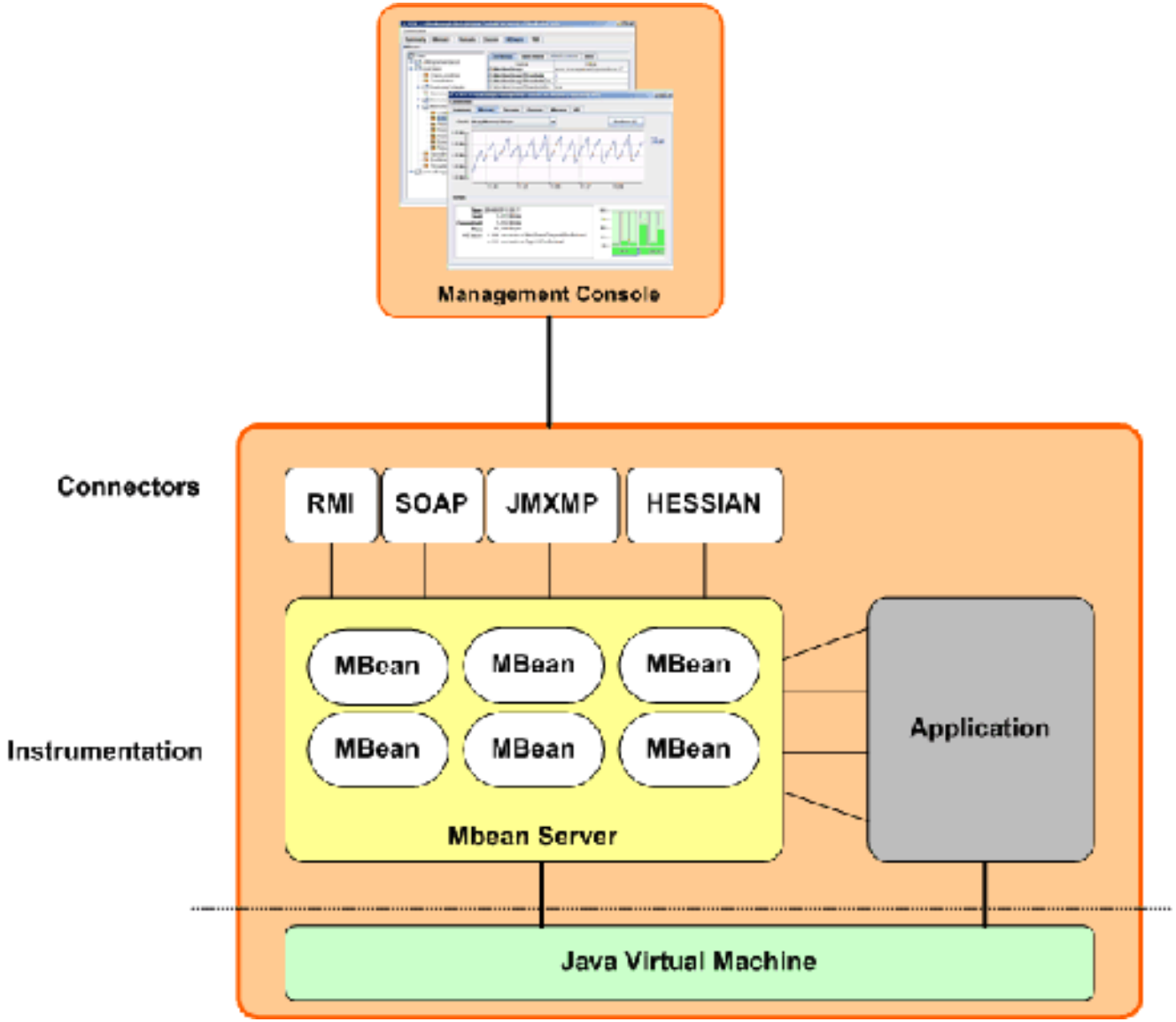
- The Java Virtual Machine
- Lightweight / Micro Containers
  - Single JVM Scope
  - JMX, Spring, OSGi, JbossMC, Mico/Pico, Hivemind
- Enterprise Containers
  - Distributed on multiple JVM's
  - Web containers, application containers, EJB containers, grids

JMX

JMX

- Java Management Extension
- Specialized Component Container
- Exposes Management Units (MBeans)
- Application Instrumentation
- Control & Configuration
- Event / Notification Model
- Monitoring / Timer Services
- Part of Java 5

JMX Management Architecture



JMX MBean

- Management Beans
- Standard MBean
  - Easy to implement
  - Follows naming pattern
  - Requires an associated interface
- Other MBeans
  - Suited for tools / API's development
  - Dynamic MBean
  - Model MBean
  - Open MBean

JMX MBean Registration

- Obtain Instance of MBean Server
- Create MBean Component
- Register with ObjectName Object
- ObjectName Object
  - Provide naming strategy for MBean
  - Specially formatted string
  - **`domain:key=value[,key2=value2]*`**
  - **`house:room=livingroom`**

JMX Standard Bean Definition

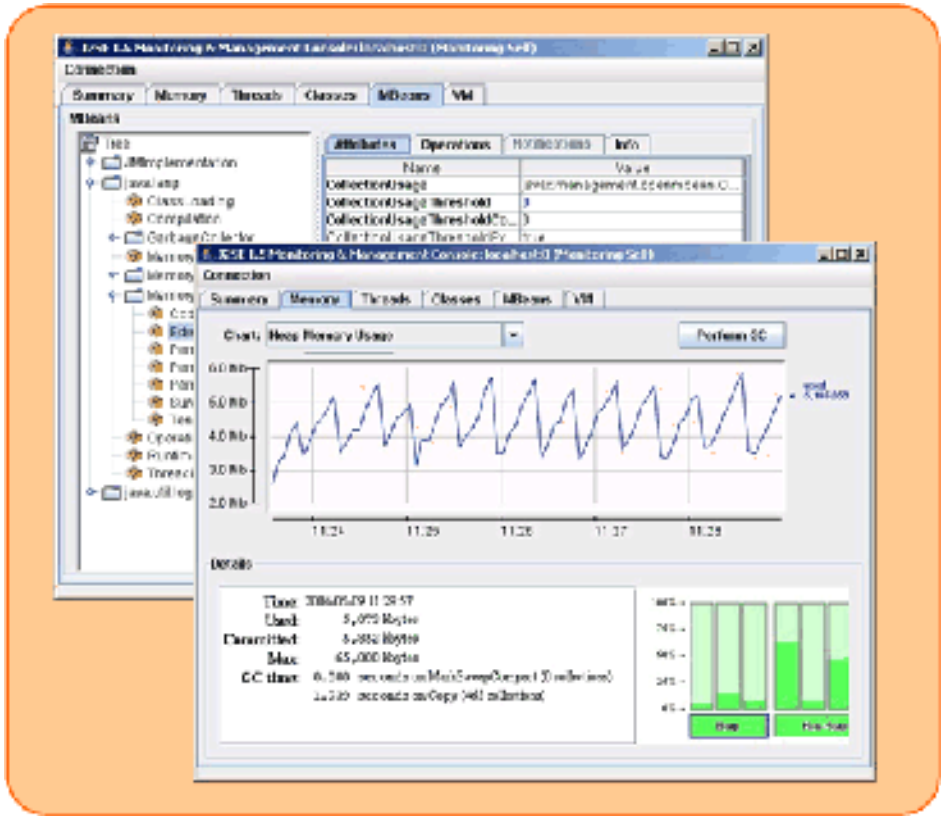
```
public interface LightMBean {
    public void turnOn();
    public void turnOff();
    public boolean isOn();
}
```

```
public class Light implements LightMBean {
    private boolean onState;
    private int onCount;

    public Light(){ turnOn() }

    public void turnOn() {
        onState = true;
        onCount ++
    }

    public void turnOff(){ onState = false; }
    public boolean isOn() { return onState }
    public int getOnCount(){ return onCount; }
}
```

JMX MBean Registration

```
public class LightingManager {

    public void RegisterLight () throws Exception {

        MbeanServer svr = ManagementFactory.getPlatformServer();

        String name = "home.lighting:type=light,location=porch";

        ObjectName name = new ObjectName(name);

        svr.registerMBean(new Lightbulb());
    }
}
```

JMX JConsole



- Desktop Management Console
- Graphically Expose MBeans
- Part of Java 5
- Info on GC, Memory, Threads
- Support Remote Connection
- Built-in Security
- Requires VM-Argument Switch

```
-Dcom.sun.management.jmxremote
```

Spring

Spring

- Application Framework
- Component Container
- Provide Operating Context
- Uses POJO-Based Component
- Dependency Injection
  - Creates object instances
  - Injects object data
  - Injects object relationships
  - Follows JavaBeans convention

Spring Container

- General Purpose Container
- Component Creation
- Component Registration
- Component Lookup / Discovery
- Broadcasts System Events
- Life Cycle Services

Spring Framework API's

- Full Service Stack
  - Data Access (JDBC, ORM API's)
  - Enterprise Service (Web, EJB, Web Service)
  - Scheduling (Quartz)
  - Messaging (JMS)
  - Transaction Abstraction
  - Manageability and Monitoring (JMX)
  - AOP (Spring AOP)

Spring Configuration File

- XML File
- Simple Syntax
- Declares Java Beans
- Specifies Instantiation Policies
- Injects Bean Property Values
- Injects Constructor Values
- Injects Component Relationships

Spring Bean Configuration

```
<beans>

    <bean id="porchlight" class="home.lighting.OutdoorLight"/>

    <bean id="floodlight"  class="home.lighting.OutdoorLight"/>

    <bean id="interiorSwitch" class="home.lighting.LightSwitch">
        <property name="lights">
            <set>
                <ref bean="porchlight"/>
                <ref bean="floodlight"/>
            </set>
        </property>
    </bean>

</beans>
```

Using Spring Context

```
public class Lighting {
    private ApplicationContext context;

    public void Lighting() {

        context = new ClassPathXmlApplicationContext(
            "META-INF/spring-context.xml"
        );

    }

    public void switchLightOn() {

        Switch sw = (Switch) context.getBean("interiorSwitch");

        for(Light l : sw.getLights) {
            l.turnOn();
        }
    }

}
```

JMX and Spring

Spring JMX

- Provides Full JMX Support
- Simplifies API
  - No special Interface
  - No naming pattern
  - Registers POJO's as Model MBeans
- Declarative MBean Registration
- Flexible Management Interfaces
- JMX Connector API Support

Spring Beans Exposed

- MBean Exporter Component
- Dynamically exports POJO's as MBeans
- Exposes properties and methods
- Requires JMX Naming Format
- Controls Registration Policy
- MBean Server Discovery / Instantiation

Controlling Manageability

- Several Methods of Manageability
- Exports All Properties & Methods (Default)
- Through Code-Level Meta Data
  - Jakarta Commons Attributes
  - Java 5 Annotations
  - Granular control of JMX API
- Through Java Interfaces Export
- Specifying Method Names

Spring Bean JMX Export

```
...
<bean id="porchlight" class="home.lighting.OutdoorLight"/>
<bean id="floodlight"  class="home.lighting.OutdoorLight"/>

<bean id="interiorSwitch" class="home.lighting.LightSwitch">
    <property name="lights">
        <set><ref bean="porchlight"/><ref bean="floodlight"/></set>
    </property>
</bean>

<bean id="exporter"
    class="org.springframework.jmx.export.MBeanExporter"
    lazy-init="false">

    <property name="beans">
        <map>
            <entry key="home.lighting:type=switch"
                value-ref="interiorSwitch"/>
        </map>
    </property>

</bean>
...
```

Design Strategies

Strategies for Component Design

- Functionally Complete Components
- Design for Portability
- Design for Manageability
- Event / Notification Model
- Provide Fault Tolerance

Functionally Complete

- Single Work Unit
- Concise Functionality
- Separation of Responsibilities
- Reduce Complex Components
- Increase Functionality through Composition
- Expose Clear Control

Design for Portability

- Reduce Dependency on Container API
- Separate Core Logic from Container
- Provide Dependency Integration Layers
- Select Portable Containers

Design for Manageability

- Identify Manageability Concerns Early
- Integrate Directly in Component Model
- Separate Management Interfaces
- Provide Life Cycle Hooks
- Aggregate States & Controls

Event & Notification Model

- Establish Event Mechanism
- Observer / Listener Pattern
- Ability to Register Event Handler
- Provide Rich Event Context

Fault Tolerance

- Expect Unavailable Components
- Components Misconfiguration
- Outline Tolerance Policies
- Design for Failure
- Componentize Exception Handling
- Implement Graceful Degradation
- Avoid Container Shutdown

Conclusion

Summary

- Component-based design provides powerful extension to the OO paradigm when coupled with lightweight containers such as Spring.
- JMX, an API included in Java 5, exposes component management using instrumented beans.
- Developers can integrate manageability directly in their code by coding against a simple interface and name pattern model.
- The Spring Framework provides a set of API that makes facilitates manageability using declarative XML configuration files.
- When building manageable systems, designs for portability, provided event model, remove container dependency, and expect component unavailability.

References

- Sun JMX
  - **http://java.sun.com/products/JavaManagement/**
- Management Consoles
  - **JConsole**
    http://java.sun.com/j2se/1.5.0/docs/guide/management/jconsole.html
  - **MC4J - http://sourceforge.net/projects/mc4j/**
- MX4J – JMX Implementation
  - **http://mx4j.sourceforge.net/**
- Spring Documentation
  - **http://www.springframework.org/documentation**