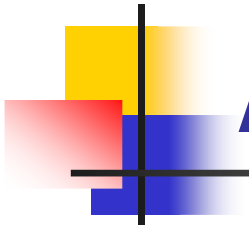
Java™ Management Extensions (JMX™)

Adam Quan



Agenda

- JMX Technology Updates
  - JMX Architecture (JSR 3)
    - MBeans
    - MBean server
    - JMX security
  - JMX Remoting (JSR 160)
  - JMX and J2SE (JSR 77)
  - JMX and J2EE (JSR 174)
- Real-World JMX Applications
- A Simple Demo

Why Management & Monitoring?

- Infrastructure Software is Getting More Complicated
  - Administration
  - Configuration
  - Monitoring
- Enterprise Business Application Characteristics:
  - Distributed
  - Complex
  - Mission Critical
  - High-Volume
  - Dynamic

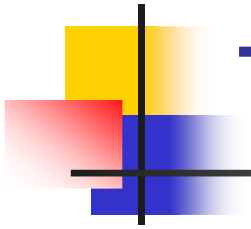
An Analogy First

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

JMX Benefits

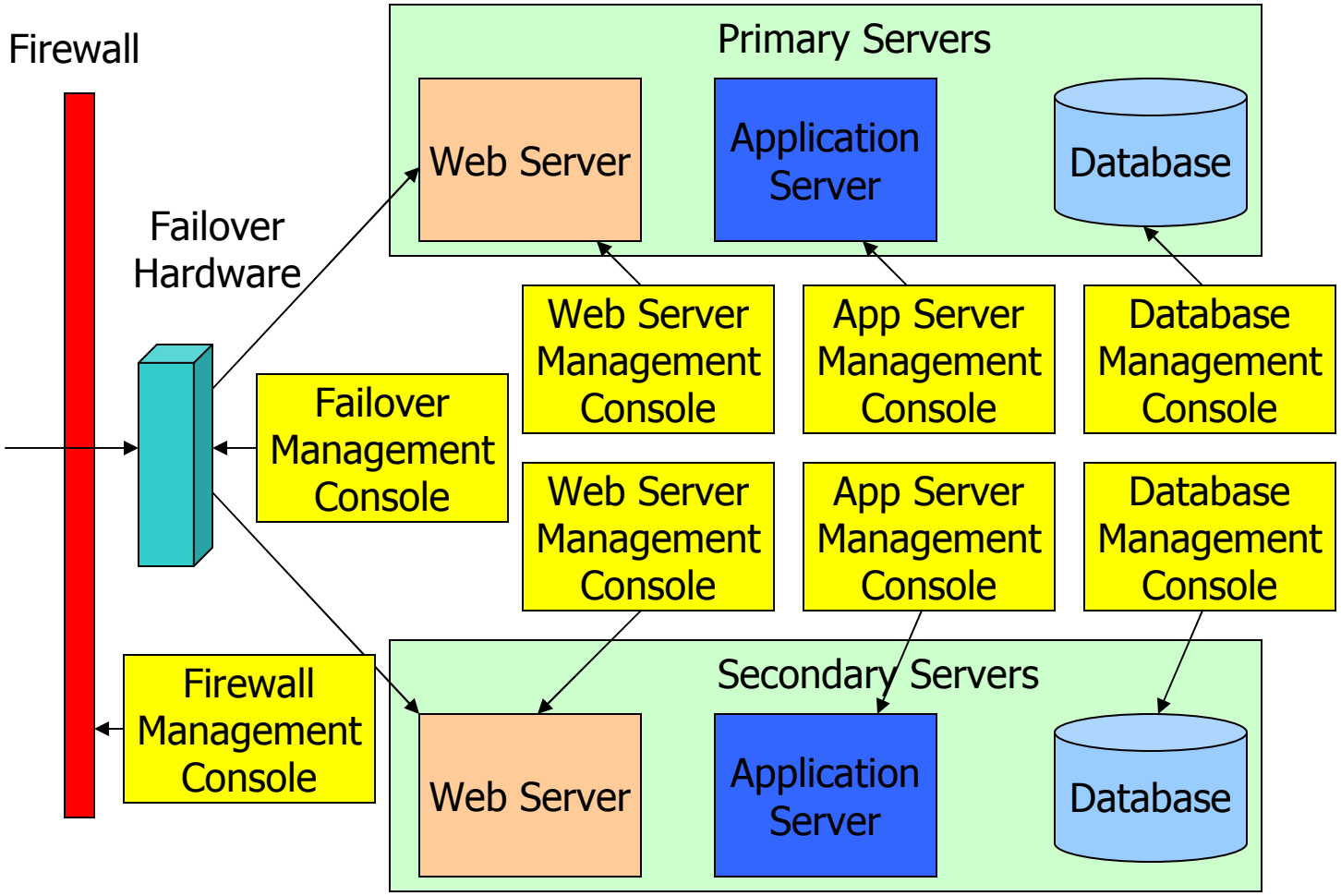
- Low Cost
- Scalable Management Architecture – Modularization of agent services
- Easy Integration – JMX smart agents manageable through various protocols
- Dynamic Management
- Integrates Existing Management Solutions
- Leverages Existing Standard Java Technologies
- Applicable to a Wide Range of Applications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



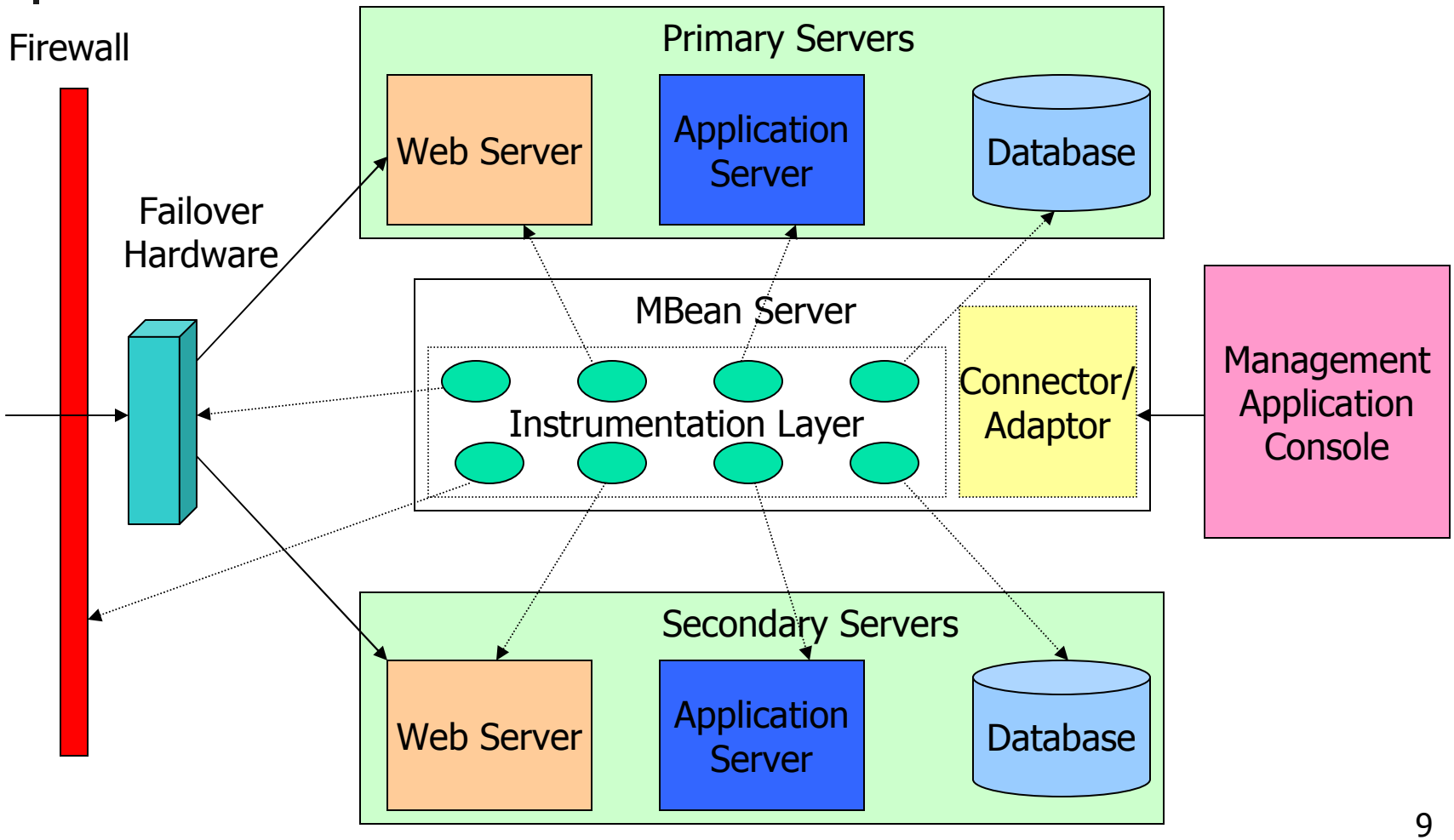
Typical JMX Usage

- 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

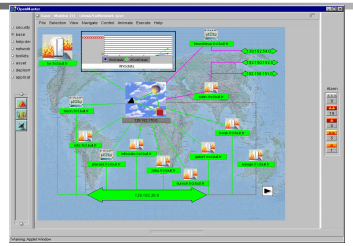


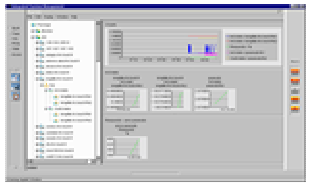
Management With JMX

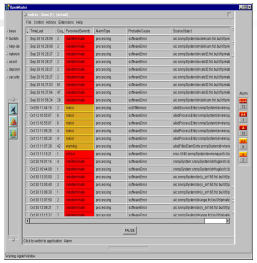


JMX Architecture

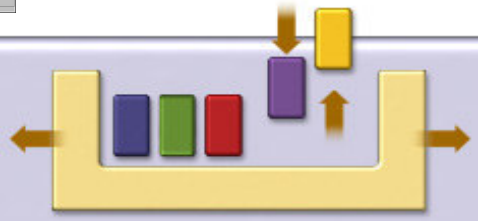
Management Applications





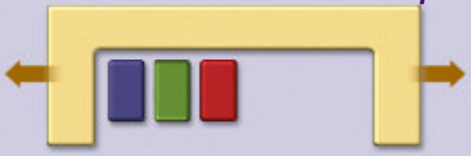


JMX Distributed Services



Protocol and InfoModel independent

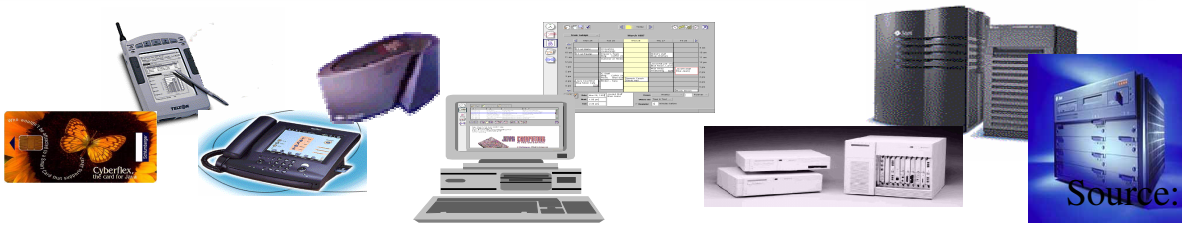
JMX Agent Services



JMX Instrumentation

Instrumentation MBeans

Managed Resources



Source: JavaSoft

What Is an MBean?

- A Java object that implements a specific interface and conforms to certain design patterns.
- The **management interface** specifies:
  - Attributes (fields) which may be accessed
  - Operations (methods) which may be invoked
  - Notifications (events) which may be emitted
  - Constructors of the MBean's Java class

MBean Types

- Four Types of MBeans:
  - Standard MBean - Simple, Static
  - Dynamic MBean - Flexible, Dynamic
  - Open MBean - Basic Data Types Only
  - Model MBean - Run Time Support
- Differ in Implementations, Not the Way They Are Managed

Standard MBean

- For static management information.
- Follows naming conventions.
- The MBean server constructs the **`MBeanInfo`** for standard MBeans via reflection.
- Constructors – Any public constructors
- Attributes – getters/setters
- Operations – Any remaining public methods
- Notification

Standard MBean Example – The Management Interface

```
public interface PrinterMBean {
  public int getPrintJobCount();
  public String getPrinterName();
  public int getPrintQuality();
  public void setPrintQuality(int q);
  public void cancelPrintJobs();
  public void performSelfCheck();
}
```

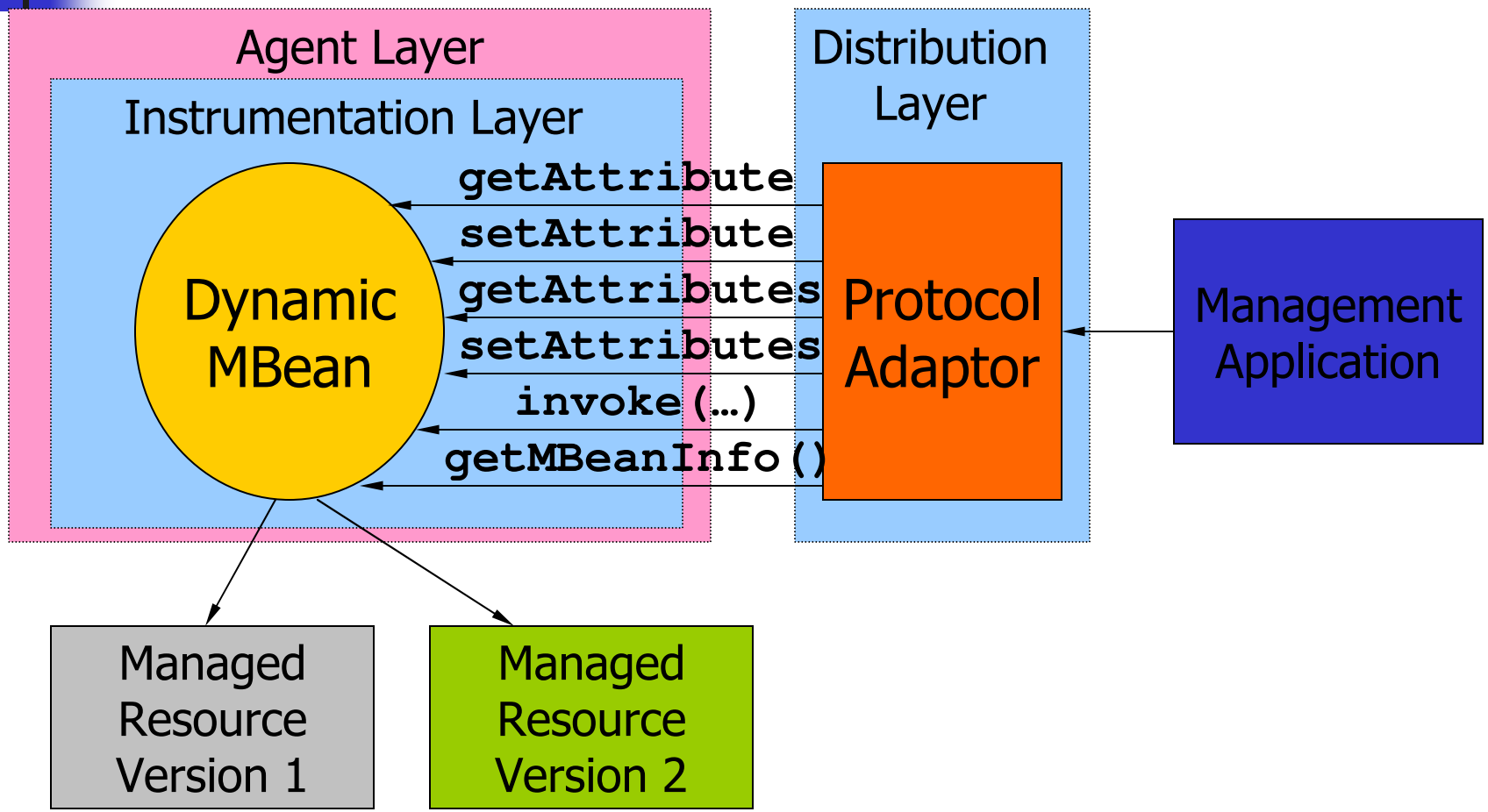
Standard MBean Example – The MBean Implementation

```
public class Printer extends
  NotificationBroadcasterSupport
  implements PrinterMBean {
  private int printJobCount;
  private String printerName;
  private int printQuality;
  …
  public void performSelfCheck(){
      …
      if ( error ) {
            sendNotification(notification);
      }
  }
```

Dynamic MBean

- Supports management information that is known only at runtime
- Implements `DynamicMBean` interface
- Responsible for building its own `MBeanInfo` structure
- Generic methods expose:
  - Attributes
  - Operations
- For changing resources

Dynamic MBean: Generic Method Access



Open MBean

- Why are Open MBeans "Open"?
  - Dynamic MBeans that use a subset of universal Java types
  - Descriptively rich metadata
  - "Open" to widest range of management applications
- Does Not Require Serialization – supports management applications not written in Java
- No Custom Classloading
- Required in 1.2

Model MBean

- Generic, Configurable Dynamic MBean
- Common Template for Different MBeans
- Configured at Runtime: Descriptors
- Default Implementation – RequiredModelMBean
- Rapid Instrumentation – No need to write an MBean
- Model MBean Features
  - Persistence – survive JMX agent crash
  - Notification Logging
  - Attribute Value Caching – better performance
  - Operation Delegation – multiple manageable resources

Model MBean Example

```
ObjectName oName = new ObjectName(…);
1 createMBean("javax.management.modelmbe
    an.RequiredModelMBean", oName );
Object managedResource = …;
2 setManagedResource(…);
ModelMBeanInfo info =
    buildModelMBeanInfo( mbeanDesc );
3 setModelMBeanInfo(…);
```

JMX Notification

- **Notification** – Java objects emitted by MBeans and the MBean server to encapsulate events, alerts, or general information.
- Based on the Java event model
- Specifies notification objects, the broadcaster, the listener and filter interfaces that notification senders and receivers must implement.
- Management applications listen to the MBean and MBean server notifications remotely.

Object Name

- Uniquely identifies MBean
- Two parts
  - domain name
  - key/value property list (unordered)
- `[domainName]:name=value[,name=value]*`
- `ObjectName` class

JMX Agent Layer

- JMX Agent – A Java process that provides a set of services for managing a set of MBeans. The container for an MBean server.
- JMX Agent Level
  - MBean Server(s)
  - Agent Services
- Four Mandatory Standard Services (MBeans)
  - M-Let Service
  - Timer Service
  - Monitoring Service
  - Relation Service

MBean Server

- A **registry** for MBeans – MBeans have to be registered.
- Exposes management interfaces:
  - Creates MBeans
  - Registers MBean
  - Handles notifications – add/remove listeners.
  - Handles access to MBeans – get/set attributes, invoke operations.
  - Handles MBean queries – based on object names and attributes.
- MBeanServer mbeanServer = MBeanServerFactory.createMBeanServer("HelloAgent");

Standard Agent Services: M-Let Service

- Expand JMX agent CODEBASE.
- Dynamically load new components to the MBean server.
- Allows network-enabled application to load its MBean components from remote hosts.
- Enables hot deployment.
- <MLET
  CODE = “com.foo.HelloWorld”
  ARCHIVE = “helloWorld.jar"
  CODEBASE = “http://www.foo.com/jars”
  NAME = “helloWorld:name=hello,url=www.foo.com”>
  </MLET>

Standard Agent Services: Timer Service

- Schedule notifications for tasks that need to be run once or at regular intervals.
  - Specific Times
  - Intervals
- More elegant solution than background threads in a J2EE environment.

Standard Agent Services: Monitoring Service

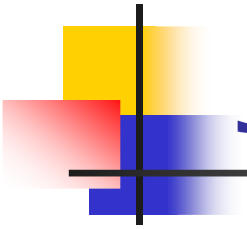
- Common Monitor Management Interface Attributes
  - ObservedObject
  - ObservedAttribute
  - GranularityPeriod
  - Active
- **`MonitorNotification`**
- Three monitor implementations
  - Counter monitor: *threshold*
  - Gauge monitor: *high|low threshold*
  - String monitor: *matches | differs*

Standard Agent Services: Relation Service

- Define relations between MBean components and react to changes in case of MBean dependencies.
- Consistency
- Cardinality
- Conceptually relate MBeans
- Allows to manage MBeans as related groups
- Typically grouped based on managed resources or workflows they participate in.

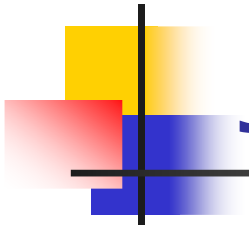
Distributed Services Level

- Connectors
  - Contains *connector client* and *connector server*
  - Hide the actual protocol being used to contact the agent.
  - Can use any convenient transport – RMI, RMI/IIOP, HTTP, SOAP, JMS, JINI, "Raw" TCP/UDP, …
- Protocol Adaptors
  - Listen for incoming messages that are constructed in a particular protocol like HTTP, SNMP
- JMX Remoting API (JSR-160) – The Standardization



JMX Remote API (JSR 160)

- Allows Any Java Client To:
  - **Discover** JMX Connectors
  - **Connect** to Any Running JMX Server
  - **Access** to a Running JMX Server via a Protocol-Independent Client API
- JMX Remoting API Goals
  - Interoperable – Completely defines standard protocols
  - Transparent – Remote access like local
  - Secure – Built on JSSE, JAAS, SASL
  - Flexible – New transport protocols can be added easily



JMX Remoting Connectors

- RMI Connector – Required
- JMXMP Connector – Based on Java serialization over TCP
- Generic Connector – Configurable by plugging in modules to define *Transport Protocol* and *Object Wrapping*
- Connector Server Address
  - service:jmx:rmi://host/…
  - service:jmx:jmxmp://host:port

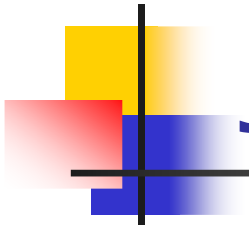
JMX Remoting Example: Remote Access Like Local

- Local Access:

```
MBeanServer mbs = MBeanServerFactory.createMBeanServer();
mbs.createMBean( className, obName );
Object a = mbs.getAttribute( obName, "attr" );
Set names = mbs.queryNames(…);
```

- Remote Access:

```
JMXConnector c = JMCConnectorFactory.connect(url);
MBeanServerConnection mbs = c.getMBeanServerConnection();
mbs.createMBean( className, obName );
Object a = mbs.getAttribute( obName, "attr" );
Set names = mbs.queryNames(…);
c.close();
```



JMX Security

- JMX Security Risks:
  - MBeanServer API Access
  - MBean API Access
  - Trusted MBean Sources
- JMX Security introduced in JMX 1.2
- Based on Standard Java Security Model by defining Permissions (**`javax.security.Permission`**)
  - **`MBeanServerPermission`**
  - **`MBeanPermission`**
  - **`MBeanTrustPermission`**
- Why not J2EE Role-base Security Model?

MBean Permission Example

```
grant applone.jar { permission
    javax.management.MBeanServerPermission
    "createMBeanServer, releaseMBeanServer"; };

grant applone.jar {
    permission javax.management.MBeanPermission
    "com.Foo#doIt[d1]", "invoke"; };

grant signedBy "MyOrg" { permission
    javax.management.MBeanTrustPermission "register";};
```

JMX and J2SE

- JMX is going into J2SE 1.5
- JSR 174 – Monitoring and Management Specification for Java

Health Indicators:

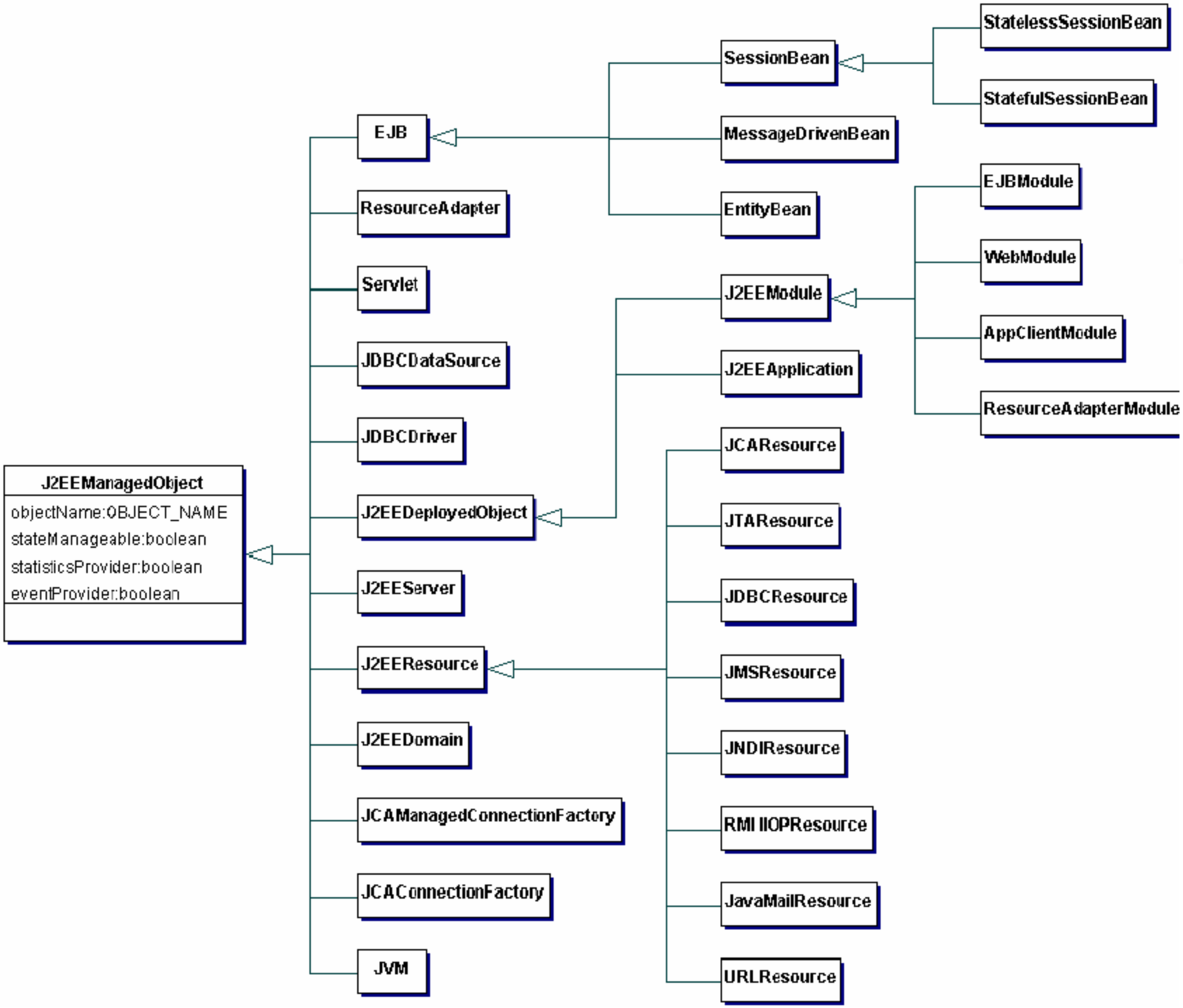
Class load/unload
Memory allocation statistics
Garbage collection statistics
Monitor info & statistics
Thread info & statistics
Object info
...

Runtime Control:

Heap size
Verbose GC on demand
Garbage collection control
Thread creation control
Just-in-time compilation control
...

JMX and J2EE: J2EE Management (JSR 77)

- JMX is Going into J2EE 1.4
- Further J2EE Standardization: Vendor Independent Management of J2EE Servers
- Defines a Model of J2EE Managed Objects
- No Java Classes
- Managed EJB (MEJB)
- Management Capability:
  - Event Handling – state/attribute change, creation/destruction
  - State Management – manage an object's state
  - Performance Monitoring – statistics data

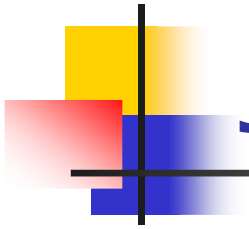

J2EEManagedObject
objectName:OBJECT_NAME
stateManageable:boolean
statisticsProvider:boolean
eventProvider:boolean
EJB
ResourceAdapter
Servlet
JDBCDataSource
JDBCDriver
J2EEDeployedObject
J2EEServer
J2EEResource
J2EEDomain
JCAManagedConnectionFactory
JCAConnectionFactory
JVM
SessionBean
MessageDrivenBean
EntityBean
J2EEModule
J2EEApplication
JCAResource
JTAResource
JDBCResource
JMSResource
JNDIResource
RMI IIOPResource
JavaMailResource
URLResource
StatelessSessionBean
StatefulSessionBean
EJBModule
WebModule
AppClientModule
ResourceAdapterModule

Managed EJB (MEJB)

```
Context ctx = new InitialContext();
ManagementHome home =
    (ManagementHome)PortableRemoteObject.narrow(
    ctx.lookup("ejb/mgmt/MEJB"), ManagementHome.class);
1 Management mejb = home.create();
String domain = mejb.getDefaultDomain();
2 Set names = mejb.queryNames(
    new ObjectName(domain+":j2eeType=EJBModule,*"),null);
Iterator itr = names.iterator();
while( itr.hasNext() ) {
    ObjectName name = (ObjectName)itr.next();
3   ObjectNames[] ejbs = (ObjectName[])mejb.getAttribute(name, "ejbs");
}
```

All EJB Modules

All EJBs in a Module

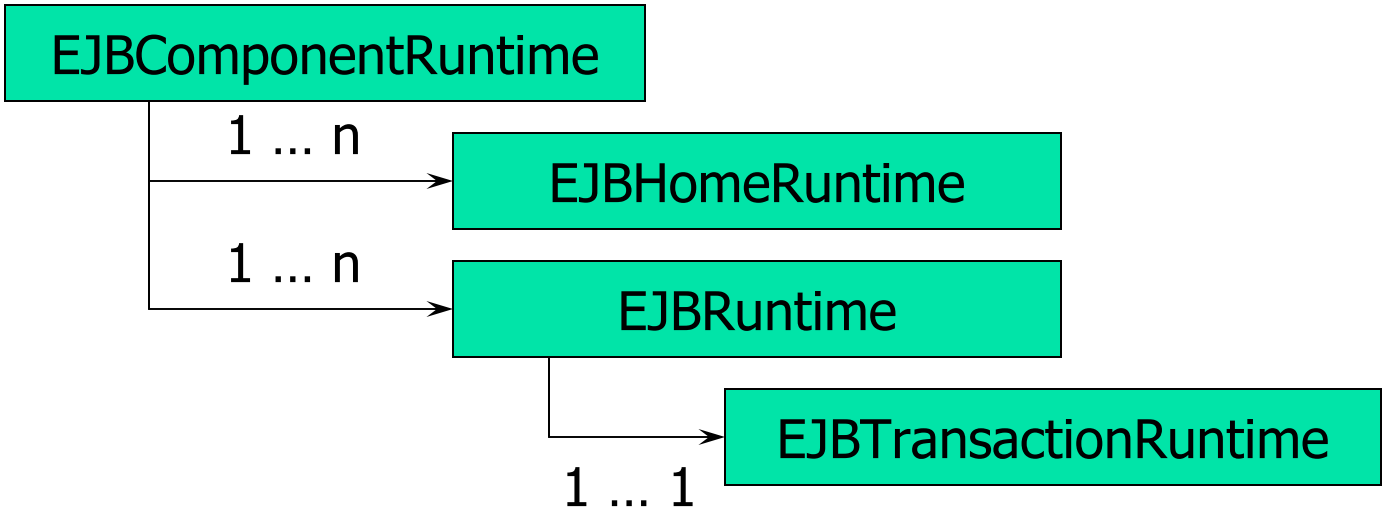


JMX In Use Today

- All Major Application Servers
  - BEA WebLogic
  - IBM WebSphere
  - JBoss
- Management Applications
  - Tivoli Web Component Manager
  - AdventNet Middleware Manager
  - Dirig Softare
- Performance Management Tools
  - Existing instrumentation
  - Custom instrumentation

JMX Use Case: WebLogic

- Administration and Configuration MBeans
- Runtime MBeans – information about open servlet sessions, active JDBC connections, JTS transactions, pooled EJBs, JMS messages
- Anything deployed into WebLogic is automatically manageable



Real-World JMX Usage

- Log4j Runtime Control
- JDBC Connection Pool Management
- HTTP Session Management
- Generic Cache Control
  - Cache Size
  - Cache Policy
  - Cache Hit Rate
  - Cache Refresh

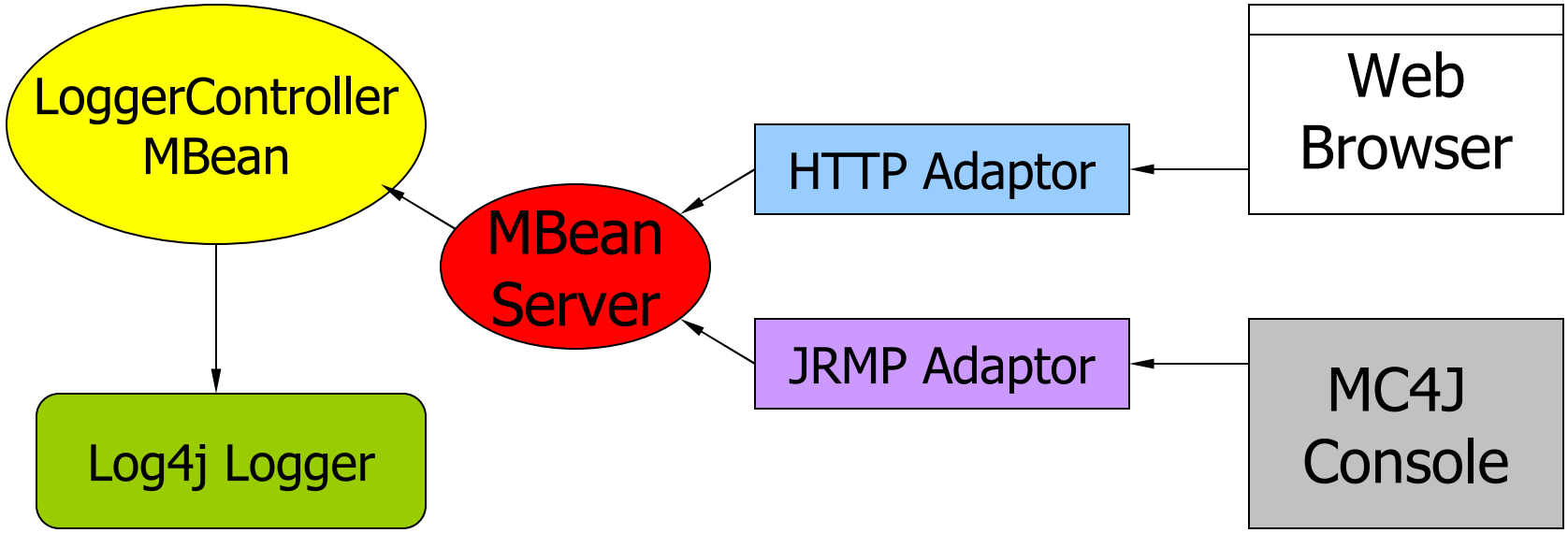
Real-World JMX Usage (cont.)

- Socket Adaptor for Integration
- Monitoring Tool to Collect Data
- Key Application Transaction Monitoring
- Pluggable Security Providers
- Timer
- Queue Management
  - Custom Queue
  - JMS Queue
- …

Summary

- JMX technology is a **standard**, **mature** and **complete** open solution to manage and monitor both Java applications and underlying infrastructure.
- Developers should have application management and monitoring in mind during design phase, not as an after-thought.
- Developer's main task will be writing interfaces and reporting tools that make use of the instrumentation already provided by Application Servers.

A Simple Demo: Log4j Control



A Simple Demo: Log4j Control (cont.)

```
public interface LoggerControllerMBean {
    public String getLevel(String loggerName);
    public void setLevel(String loggerName,String level);
}

public class LoggerController implements LoggerControllerMBean{
    public String getLevel(String loggerName) {
        Logger logger = Logger.getLogger(loggerName);
        return logger.getLevel().toString();
    }

    public void setLevel(String loggerName, String level) {
       Level level = Level.toLevel(level);
       Logger.getLogger(loggerName).setLevel(level);
    }
}
```

References

- ***JMX in Action***
  Benjamin G. Sullins, Mark Whipple
  Publisher: *Manning Publications Company*
- ***JMX:*** *Managing J2EE with Java Management Extensions*
  Marc Fleury, Juha Lindfors
  Publisher: *Sams*
- ***Java and JMX:*** *Building Manageable Systems*
  Heather Kreger, Ward K. Harold, Leigh Williamson, Ward Harold
  Publisher: *Pearson Education*
- ***Sun JMX Page****: http://java.sun.com/products/JavaManagement*
- ***AdventNet****: www.adventnet.com*
- **MC4J:** http://mc4j.sourceforge.net
- **MX4J:** http://mx4j.sourceforge.net