
# Meeting #47


# JMX

## Java Management Extensions

Dominik Dorn

# Overview

- JMX - Definition
- MBean
- MBean-Server
- Connectors
- Adaptors
- JMX in J2EE / JavaEE
  - Location Transparency
  - Server Management

# Definition of JMX

"... provides the **tools for building**

- distributed,
- Web-based,
- modular and
- dynamic

**solutions** for

- managing and monitoring
  - *devices*, *applications*, and *service-driven networks*."

(oracle.com)

# MBean

- MBean = Managed Bean
- Different Types
  - Standard MBean
    - Notification Mechanisms
  - Dynamic Mbean
  - Model MBeans
  -

# MBean

- Standard MBean = Plain **POJO**
  - attributes (any valid Java type),
  - getters/setters / (state changing) methods
- pre JMX 2.0: Interface + Implementation
- JMX 2.0: POJO annotated with **@javax.management.MBean** and **@javax.management.ManagedOperation** **@javax.management.ManagedAttribute**

# Standard MBean Notifications

- Event-System for Managed Beans
- Interfaces javax.management.
  - Notification (extends java.util.EventObject)
  - NotificationBroadcaster
  - NotificationFilter
  - NotificationListener
- Server provided events
  - Attribute enabled / disabled, Bean added/removed
- Own event types

# Dynamic MBean

- Generic Interface
- Use it to create dynamic clients
- Basically “**reflection for JMX**”

```
package javax.management;

public interface DynamicMBean  {

    Object getAttribute(String attribute) throws AttributeNotFoundException, MBeanException,
ReflectionException;

    void setAttribute(Attribute attribute) throws AttributeNotFoundException,
InvalidAttributeValueException, MBeanException, ReflectionException;

    AttributeList getAttributes(String[] attributes);

    AttributeList setAttributes(AttributeList attributes);

    Object invoke(String actionName, Object[] params, String[] signature) throws MBeanException,
ReflectionException;

    MBeanInfo getMBeanInfo();
}
```

- Nice feature: Hot Deploying Resources

# Model MBean

- Extending Dynamic MBeans
- Adds Persistence: **automatic (periodic) loading/saving bean properties**
- Allows to specify **MetaData** (types, constructors, etc.) for dynamic MBeans (instead of if/else blocks in normal dynamic Mbeans)
- **Attribute Caching** / performance improvements

# MXBean

- Normal Mbean would basically allow a “inconsistent read”
- MXBeans allow to “bundle” attributes, to provide consistent reads.

example: a statistical MemoryMBean with attributes

min, current, max, average

# MBean-Server

- Manages MBeans
- Has services
  - Timer Service
  - Monitoring Service
  - M-Let Service
  - Relation Service
- Allows to register/lookup beans

# Timer Service

- Interface **TimerMBean** (broadcaster MBean)

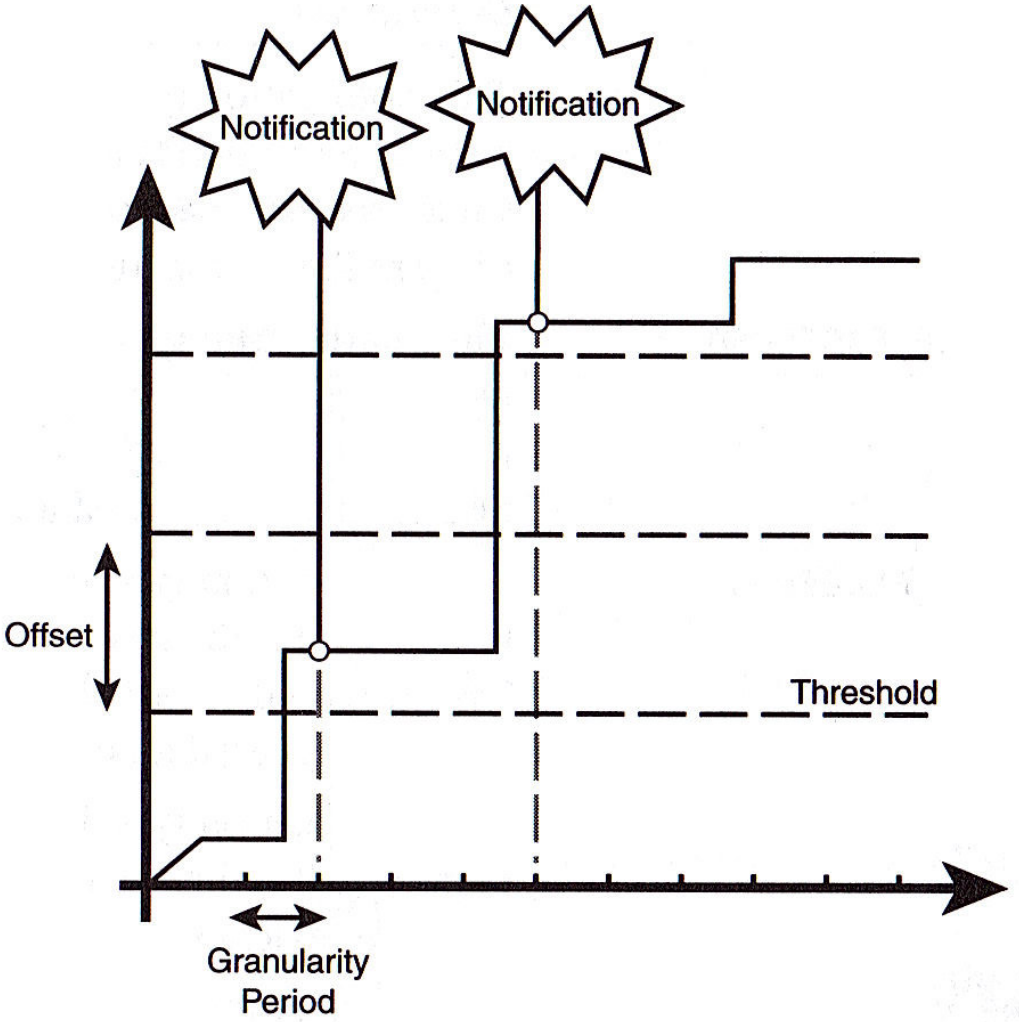
```
addNotification(String type,
String message,
Object userData,
Date date)
```

- Class **Timer** (implements TimerMBean) “sends out an alarm at a specified time that wakes up all the listeners registered to receive timer notifications”
- Class **TimerNotification** (extending EventObject)
- Subscribe using JMX Notification System.

# Monitoring Service

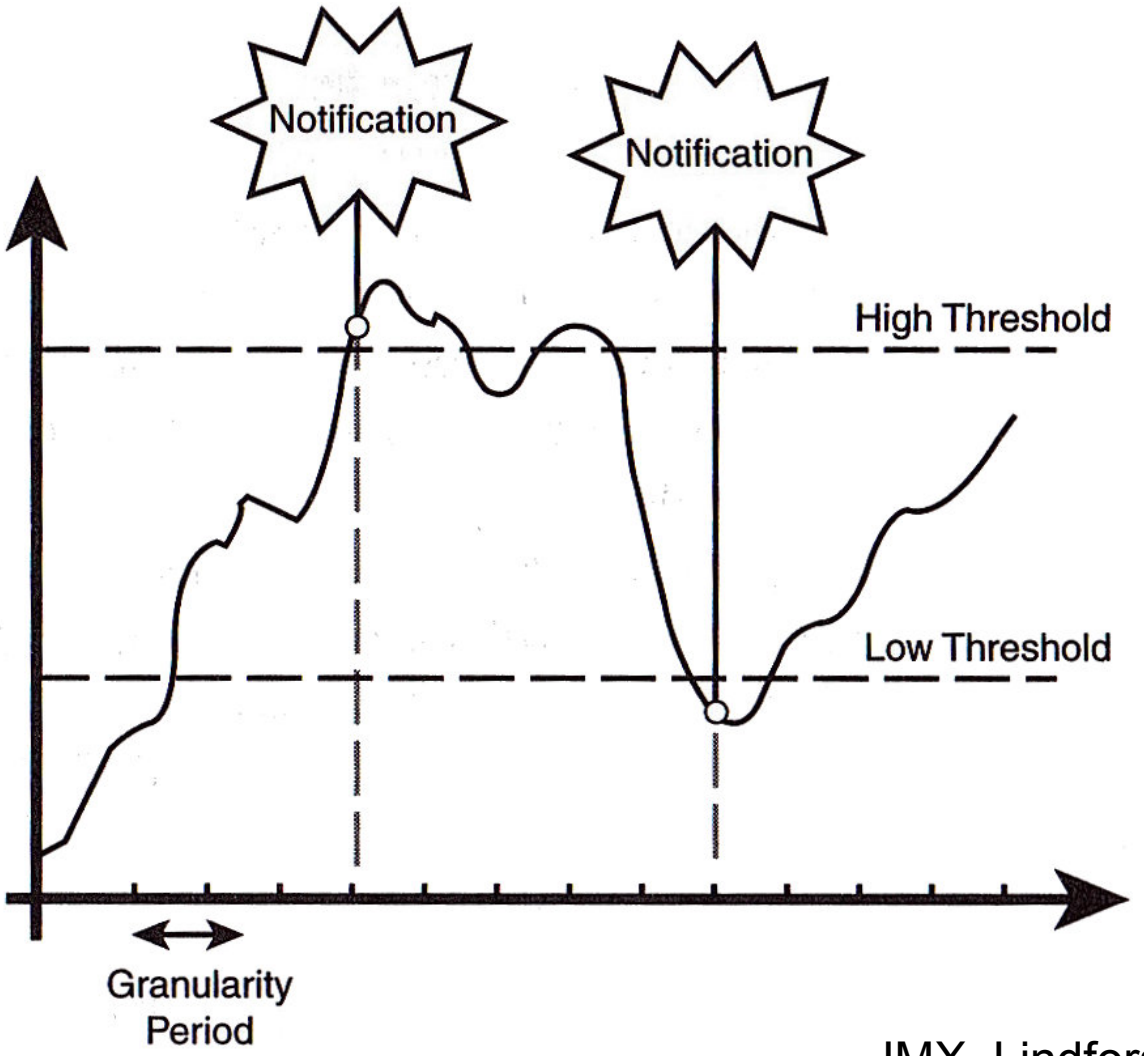
- Allows to monitor attributes of MBeans
- javax.management.monitor.**CounterMonitor**
  "A counter monitor sends a *threshold notification* when the value of the counter reaches or exceeds a threshold known as the comparison level."
- javax.management.monitor.GaugeMonitor
  "observes an attribute that is continuously variable with time"
- javax.management.monitor.StringMonitor
  "Allows to monitor MBean attributes of type java.lang.String"
  (basically an AttributeListener)

# CounterMonitor




JMX, Lindfors & Feury, 2001

# GaugeMonitor




JMX, Lindfors & Feury, 2001

# Relation Service

- Allows to specify relations and role meta data between MBeans.
- Example:
  Observable Bean: ThreadMonitor
  Observer Bean: GaugeMonitor

# M-Let Service

“a mechanism .. to
**dynamically load new Java classes**
to the MBean server. The new
**classes can be loaded** from
**a local machine** or
**can be downloaded from a remote host**
accessible from the network.” [JMX, Lindfor + Fleury, 2002]

# Connectors

- Allow to access the MBeanServer from other environments/JVMs
- Interface javax.management.remote.**JMXConnector**
- Default impl. included in JavaSE: RMIConnector
- Other samples:
  - SOAP Connector
  - JMS Connector (async.)

# Adaptors

- “translate” between protocols.
- Example: SNMPAdaptor allows using JMX through SNMP applications.
- HTTPAdaptor: RESTful configuration (e.g. included in Glassfish)

# Tooling

- Frameworks
  - Part of JavaSE since JavaSE 5
  - SpringFramework has helpers
- JConsole
- Glassfish Web Console

# Links / Literature

- The Java Tutorials:
  http://docs.oracle.com/javase/tutorial/jmx/
- JMX 2.0 Annotations Example
  http://marxsoftware.blogspot.com/2008/08/playing-

- JMX – Managing J2EE with Java... (Lindfors, Fleury, 2002)

That's it!

twitter: @domdorn

mail: {firstname}@{firstname}{lastname}.com ;)