5 JMX and MBean interview questions & answers

Posted on March 11, 2015 by Arulkumaran Kumaraswamipillai

Q1. What is a JMX? What are the key components of JMX?
A1. **JMX** stands for Java Management Extensions (JMX), which is a technology to monitor and manage any Java applications are running in either a local or a remote Java Virtual Machine (JVM).

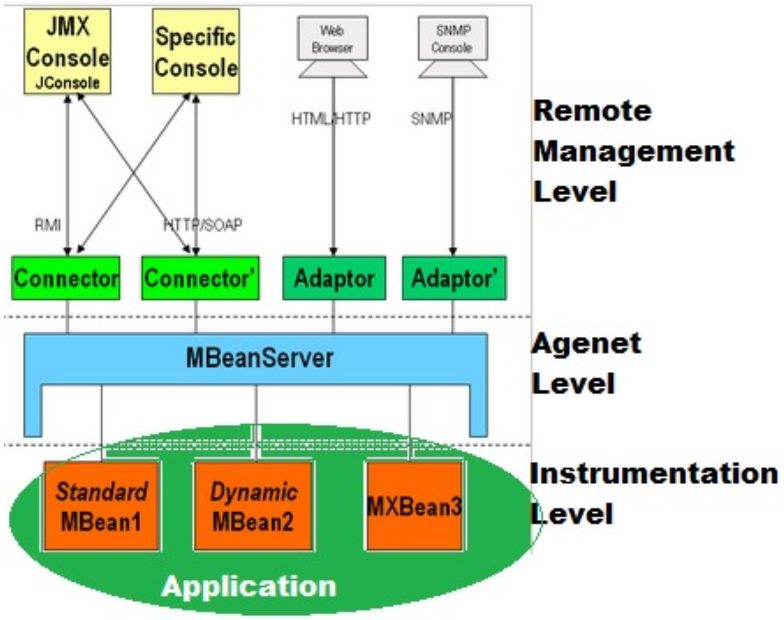
1) MBeanServer, which acts as a container for MBeans, providing remote access, namespace management, and security services.

2) MBean, which is a managed Java object that follows the design patterns set forth in the JMX (E.g. interface name must end with MBean, etc). It represents represent

manageable resources such as an application, service, a component, or a device.

3) JMX client, which connects to an MBeanServer. Jconsole is a JMX client. VisualVM is another JMX client.

Q2. What are the 3 levels of a JMX architecture?
A2.



JMX architecture – source wikepedia

From top to bottom:

1. **Remote Management:** Enables remote applications to access the MBeanServer through connectors and adaptors. A connector provides full remote access to the MBeanServer API using various communication protocols RMI, IIOP, JMS, WS-*, etc, whilst an adaptor adapts the API to another protocol (SNMP, …) or to Web-based GUI (HTML/HTTP, WML/HTTP, …).

2. **Agent Level:** The main component of a JMX agent is the MBean server. This is a core managed object server in which MBeans are registered. A JMX agent also includes a set of services for handling MBeans. JMX agents directly control

16 Technical Key Areas

resources and make them available to remote management agents.

3. **Instrumentation Level:** Resources, such as applications, devices, or services, are instrumented using Java objects called Managed Beans (MBeans). MBeans expose their management interfaces, composed of attributes and operations, through a JMX agent for remote management and monitoring.

Q3. What is an MBean & what conditions should an MBean or managed been satisfy?
A3. The MBean represents a resource running in the JVM, such as a stand alone or a JEE application service (transactional monitor, JDBC driver, etc.). They can be used

- for collecting metrics on concerns like performance, resources usage.
- for getting and setting application configurations or properties.
- for notifying events like faults or state changes.

An MBean exposes a management interface that consists of the following:

- A set of readable or writable attributes, or both.
- A set of invokable operations.
- A self-description.

An MBean is implemented as a Java class that meets the following conditions:

1. It cannot be a non-static inner class
2. A standard MBean is defined by writing a Java interface called XXXXMBean and a Java class called XXXX that implements that interface. Every method in the interface defines either an attribute or an operation in the MBean.
3. By default, every method defines an operation. **Attributes** and **operations** are methods that follow certain design patterns.

⊞ Design Concepts (7)
⊞ Design Patterns (11)
⊞ Exception Handling (3
⊞ Java Debugging (21)
⊞ Judging Experience Ir
⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

MBean Interface

```
1   package com.simple;
2
3  public interface HelloMBean {
4
5      public void sayHello();      //exposes operat
6      public v sayGoodNight();  //exposes operatio
7
8      public String getName();      //exposes read
9
10     public int getStreetName();   //exposes read
11     public void setStreetName(String streetName)
12 }
```

MBean Implementation

```
1  package com.simple;
2
3  public class Hello implements HelloMBean {
4
5      private final String name = "Peter";
6      private String streetName =  "Not Provided";
7
8      public void sayHello() {
9          System.out.println("hello" +  name);
10     }
11
12     public void sayGoodNight() {
13         System.out.println("goodnight" +  name);
14     }
15
16     public String getName() {
17         return this.name;
18     }
19
20     public String  getStreetName() {
21         return this.streetName;
22     }
23
24      public void setStreetName(String streetName
25         this.streetName =  streetName;
26      }
27 }
```

MBean Server stand alone

Once a resource has been instrumented by MBeans, the management of that resource is performed by a JMX agent. The core component of a JMX agent is the MBean server.

```
1  package com.simple;
```


How good are your .....?

```
2
3  import java.lang.management.*;
4  import javax.management.*;
5
6  public class Main {
7
8      public static void main(String[] args) throw
9
10         MBeanServer mbs = ManagementFactory.getP
11         ObjectName name = new ObjectName("com.si
12         Hello mbean = new Hello();
13         mbs.registerMBean(mbean, name);
14
15         //...wait for ever code
16     }
17 }
18
```

Whilst the server is running, you can connect to it using jconsole. You can interact with operations and attributes via the jconsole GUI. This is demonstrated elsewhere with non trivial tutorials.

Q4. What is an MXBean?
A4. An MXBean is a type of MBean that references only a predefined set of data types. In this way, you can be sure that your MBean will be usable by any client, including remote clients, without any requirement that the client have access to model-specific classes representing the types of your MBeans. An MXBean provides a convenient way to bundle related values together without requiring clients to be specially configured to handle the bundles.

Q5. Where did you use an MBean? Can you give some practical examples?
A5.

- JMX allows us to monitor local or remote applications. We can use it to detect memory and thread usage, and generate heap dumps.
- JMX allows us to generate events, alarms and notifications from an application running on the JVM.
- JMX can be used to gather application specific metrics like request counts, execution times, etc
- JMX can be used to parameterize or configure intial values for an application like inital thread count,

service retry duration, service retry count, etc. Any name/value pairs.

Practical examples tutorial style

1. Event Driven Programming in Java
2. Yammer metrics tutorial with JMX to gather metrics

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

766 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ ♥ What are the 16 technical key areas of Java programming to fast-track your career?

Q01-Q07 written test questions and answers on core Java ›

Posted in JMX, JVM, member-paid

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

 ↑