



面试官问我 JMX 了解不，我说：什么？

追梦程序猿
JAVA架构

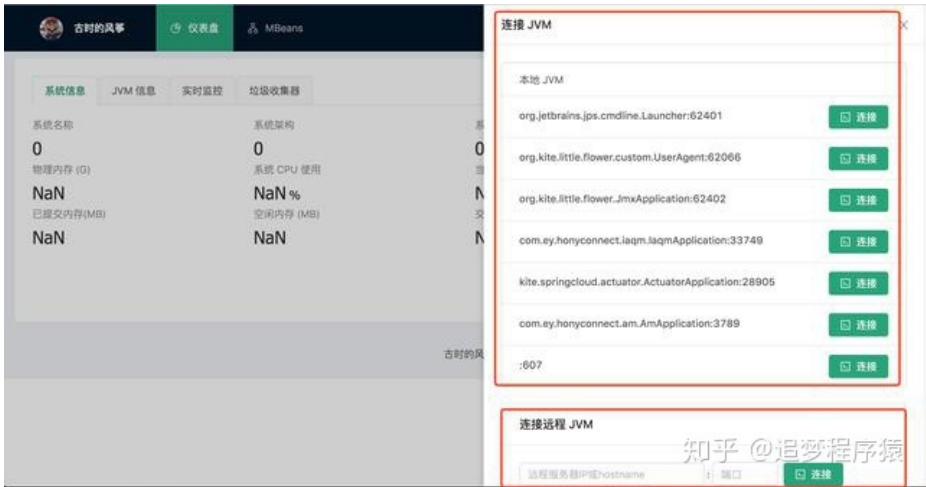
34 人赞同了该文章

如果你之前没接触过，一定会出现疑问三连击，"这是个什么玩意儿？干嘛的？有啥用？"。

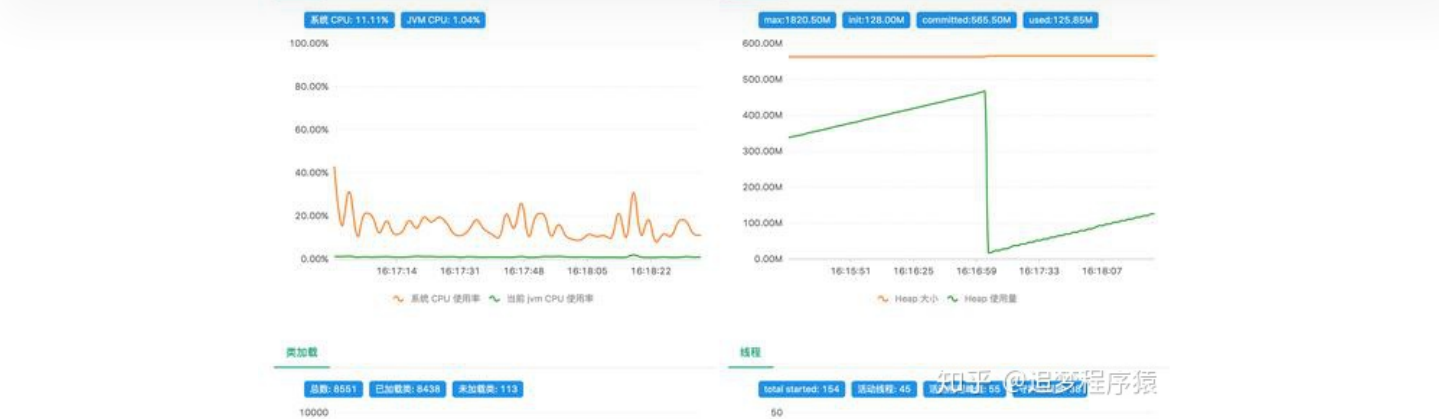
虽然可能不知道它，但是或多或少你肯定都接触过。

比如你如果用 Spring Boot，那你用过 Spring Boot Actuator 吧，它就用到了 JMX 。比如你用过 JConsole 或者 VisualVM 吧，它们也用到了 JMX。

先上图吧，利用 JMX 你可以做出这样的 JVM 监控出来。

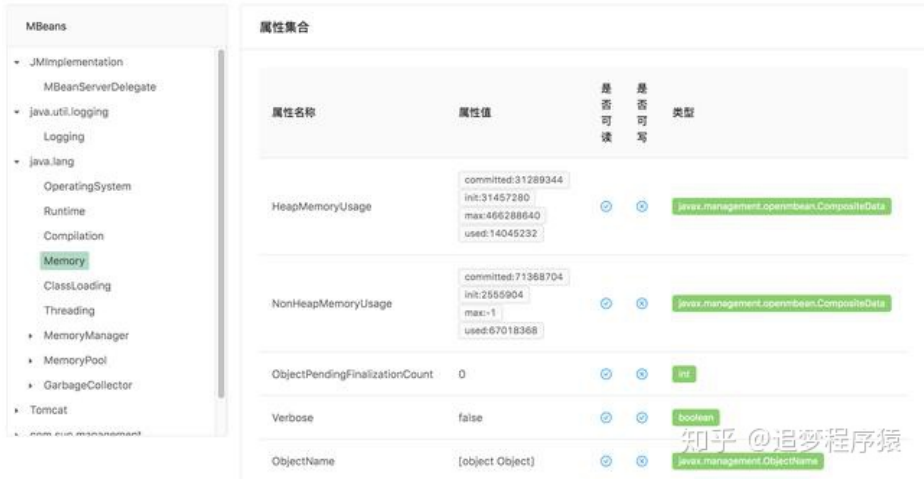


知乎
首发于
追梦程序猿



是不是挺有意思的。

上面是我自己实现的界面，下图是 JConsole 和 VisualVM 的界面。



它们是如何用到了 JMX 呢，下面一步一步说。

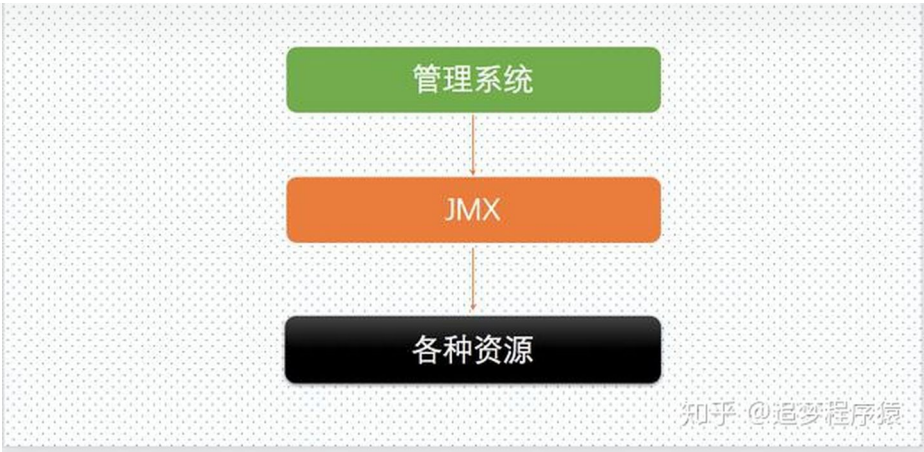
什么是 JMX

JMX 全称为 Java Management Extensions，翻译过来就是 Java 管理扩展，用来管理和监测 Java 程序。最常用到的就是对于 JVM 的监测和管理，比如 JVM 内存、CPU 使用率、线程数、垃圾收集情况等等。另外，还可以用作日志级别的动态修改，比如 log4j 就支持 JMX 方式动态修改线上服务的日志级别。最主要的还是被用来做各种监控工具，比如文章开头提到的 Spring Boot Actuator、JConsole、VisualVM 等。

JMX 既是 Java 管理系统的一个标准，一个规范，也是一个接口，一个框架。有标准、有规范是为了让开发者可以定制开发自己的扩展功能，而且作为一个框架来讲，JDK 已经帮我们实现了常用的功能，尤其是对 JVM 的监控和管理。

知乎

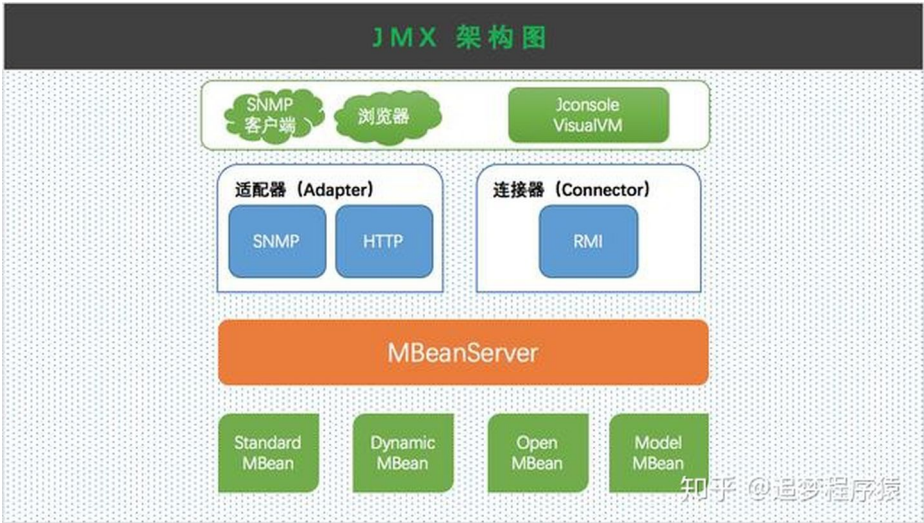
首发于
追梦程序猿



上图是 JMX 架构的简单示意图，简单理解就是管理系统通过 JMX 可以管理各种资源。

管理系统可以理解为管理客户端，比如上面说的 JConsole、VisualVM，还有 Metrics 这个非常知名的 Java 监控工具包，或者你自己通过 JMX 接口实现的客户端等。各种资源比如系统配置、JVM 指标等，或者你自己的项目中特定的资源等。

重点来了，JMX



这个架构图是把上面的架构示意图展开了，下面从底向上简单介绍一下。

MBean

JMX 是通过各种 MBean(Managed Bean) 传递消息的，MBean 其实就是我们经常说的 Java Bean，只不过由于它比较特殊，所以称之为 MBean。既然是个 Bean，里面就是一些属性和方法，外界就可以获取被管理的资源的状态和操纵MBean的行为。JMX 中共有四种类型的 MBean，分别是 Standard MBean, Dynamic MBean, Open MBean, Model MBean。JDK 提供的 MBean 主要在 java.lang.management 和 javax.management包里面。可以进去看一看，进去就能看到好多似曾相识的身影，比如 Memory 相关的、Thread 相关的，这不就是我们在 VisualVM 上看到的内容吗，没错，数据就是从这里来的。

说实话，并不用太关心这几种 MBean 的区别，但还是简单介绍下。

Standard MBean 就是普通的 Java Bean 没有区别，它也是 JMX 中最简单、使用最多的一种。主要在java.lang.management包里面

javax.management.DynamicMBean 这里，里面定义一些接口方法，比如动态获取属性、设置属性等。

另外还有两类 MBean：Open MBean 和 Model MBean，实际上它们也都是动态 MBean。

Open MBean 与其它动态 MBean 的唯一区别在于，前者对其公开接口的参数和返回值有所限制——只能是基本类型或者 javax.management.openmbean 包内的 ArrayType、CompositeType、TabularType 等类型。这主要是考虑到管理系统的分布，很可能远端管理系统甚至 MBeanServer 层都不具有 MBean 接口中特殊的类。

MBeanServer

MBeanServer 是负责管理 MBean 的，一般一个 JVM 只有一个 MBeanServer，所有的 MBean 都要注册到 MBeanServer 上，并通过 MBeanServer 对外提供服务。一般用 ManagementFactory.getPlatformMBeanServer() 方法获取当前 JVM 内的 MBeanServer。

适配器和连接器

写好的 MBean 注册到 MBeanServer 上之后，功能已经具备了。适配器和连接器就是将这些功能开放出来的方式。比如 HTTP 协议适配器，就是将功能以 HTTP 协议开放出去，这样我们就可以在浏览器使用了。但是 JDK 只是提供了适配器的实现标准，并没有具体的实现，比较常用的是 HtmlAdaptorServer，需要 jmxtools.jar 包的支持。

连接器是各种客户端最常用的，JDK 提供的默认连接器是 RMI 连接器，JConsole、VisualVM 都是使用它。

实现并使用一个 MBean

虽然 Java 提供了实现 MBean 的标准和规则，但平时我们几乎不需要开发 MBean。绝大多数的开发者接触到的也仅仅是使用 JDK 或者第三方定义好的 MBean，即便是第三方有实现 MBean，也是非常少的。我们知道的有 Tomcat 和 Spring Boot Actuator。

定义 MBean 接口 和 实体类

```
// 接口
public interface UserMBean {

    String getName();

    String getPassword();

    String getPhone();

    void say();
}

public class User implements UserMBean {

    @Override
    public String getName() {
        return "风筝";
    }
}
```



```

    }

    @Override
    public String getPhone() {
        return "18900000000";
    }

    @Override
    public void say() {
        System.out.println("Hello JMX");
    }
}

```

实体类需要继承 MBean 接口类，而接口类的命名规则是「实体类名+MBean」，这是固定的规则。

将定义好的 MBean 注册到 MBeanServer

```

public void main(String[] args) throws Exception {
    MBeanServer server = ManagementFactory.getPlatformMBeanServer();
    ObjectName userName = new ObjectName("FengZheng:type=customer,name=customerUserBean");
    server.registerMBean(new User(), userName);

    {
        //这个步骤很重要，注册一个端口，绑定url后用于客户端通过rmi方式连接JMXConnectorServer
        LocateRegistry.createRegistry(8999);
        //URL路径的结尾可以随意指定，但如果需要用Jconsole来进行连接，则必须使用jmxrmi
        JMXServiceURL url = new JMXServiceURL(
            "service:jmx:rmi:///jndi/rmi://localhost:8999/jmxrmi");
        JMXConnectorServer jcs = JMXConnectorServerFactory.newJMXConnectorServer(url, null, null);
        System.out.println("begin rmi start");
        jcs.start();
        System.out.println("rmi start");
        try {
            Thread.sleep(60 * 60 * 1000);
        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```


ManagementFactory.getPlatformMBeanServer()获取当前 JVM 的 MBeanServer，ObjectName 是 MBean 的唯一标示，一个 MBeanServer 不能有重复。完整的格式「自定义命名空间:type=自定义类型,name=自定义名称」。当然你可以只声明 type，不声明 name。

使用 JConsole 查看

JConsole 是 JDK 自带的工具，在\${JAVA_HOME}的 bin 目录下，启动即可。启动后在本地进程找到上一步启动的 main 方法所在的进程。

知乎

首发于
追梦程序猿



新建连接

本地进程(L):

名称	PID
com.ey.honyconnect.iaqm.laqmApplication	33749
kite.springcloud.actuator.ActuatorApplication	28905
org.ietfbrains.ips.cmdline.Launcher /Applicat...	58730
org.kite.little.flower.custom.UserAgent	58731
com.ey.honyconnect.am.AmApplication	3789
org.netbeans.Main --cachedir /Users/fengz...	37310

注：将对此进程启用管理代理。

在 JConsole 上方有内存、线程、类等选项卡，点击最后一个 MBean，通过这个选项卡可以看到当前 JVM 所有已定义的 MBean。可以看到系统定义的 MBean，以及刚刚我们定义的这个 MBean。

Java 监视和管理控制台

连接(C) 窗口(W) 帮助(H)

pid: 58731 org.kite.little.flower.custom.UserAgent

概览 内存 线程 类 VM 概要 MBean

FengZheng

- customer
 - customerUserBean
 - 属性
 - Name
 - Phone
 - Password
 - 操作
 - say
 - JMImplementation
 - com.sun.management
 - java.lang
 - java.nio
 - java.util.logging

属性值

名称	值
Name	风筝

刷新(R)

MBeanAttributeInfo

名称	值
属性:	
名称	Name
说明	Attribute exposed for management
可读	true
可写入	false
是	false
类型	java.lang.String

描述符

名称	值
----	---

点击属性可以在右侧查看属性值，并且在操作菜单项下，可以看到我们定义的方法，并且可以调用。

赞同 34

4 条评论

分享

喜欢

收藏

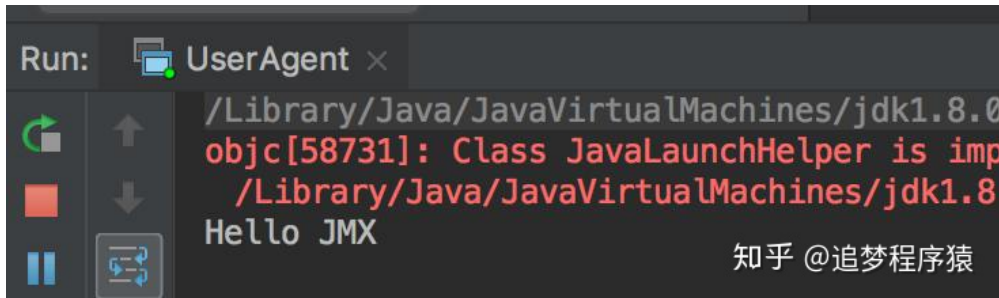
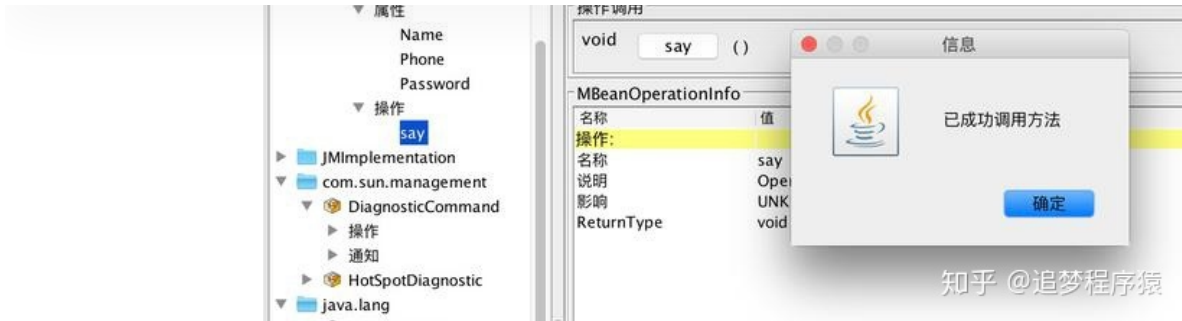
申请转载

...

https://zhuanlan.zhihu.com/p/166530442

6/11

知乎

首发于
追梦程序猿

使用 RMI 方式连接

RMI 一般是用来连接远程服务的，当然本地进程也可以。这也是实现连接远程服务客户端的第一步。我们在注册 MBean 的时候，有没有注意到注册完成后，还有一大段代码，那段代码就是用来开启 RMI 连接的，开启 8999 端口作为 RMI 访问端口，然后客户端就可以用固定的连接串连接了。

连接串的格式 `service:jmx:rmi:///jndi/rmi://host:port/jmxrmi`

```
public class Client {
    public static void main(String[] args) throws IOException, Exception, Nullf
        String jmxUrl = "service:jmx:rmi:///jndi/rmi://localhost:8999/jmxrmi";
        monitor(jmxUrl);
    }

    public static void monitor(String url) throws Exception{
        JMXServiceURL jmxServiceURL = new JMXServiceURL
            (url);
        JMXConnector jmxnc = JMXConnectorFactory.connect(jmxServiceURL, null);

        MBeanServerConnection msc = jmxnc.getMBeanServerConnection();
        String[] domains = msc.getDomains();
        for (String domain : domains) {
            System.out.println(domain);
        }
    }
}
```

首先根据连接串获得一个 `JMXServiceURL` 对象，之后通过 `JMXConnectorFactory.connect` 方法获取 `JMXConnector`，再通过 `getMBeanServerConnection` 拿到 `MBeanServerConnection`。之后就可以获取所有的 MBean 了。其中 `getDomains` 是获取所有命名空间的方法，也就是我们上面定义的 "FengZheng"，以及 `JMImplementation`、`java.lang` 等，我们在 JConsole 看到的的就是这些。

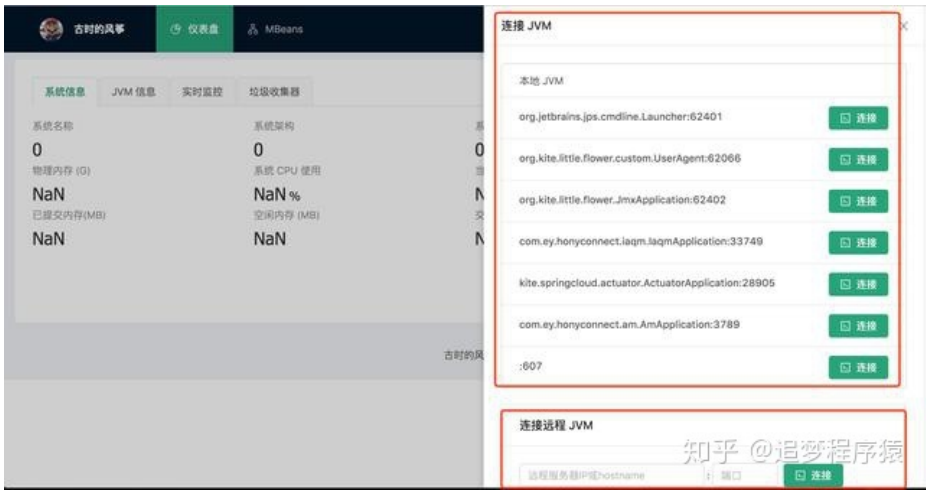
知乎

首发于
追梦程序猿

突然发现，再稍微改改就能当个 web 版的简易监控端用了。

此工具只在 hotspot JVM 8 环境下测试过。可支持查看本地 JVM 和 远程 JVM 实时监控。

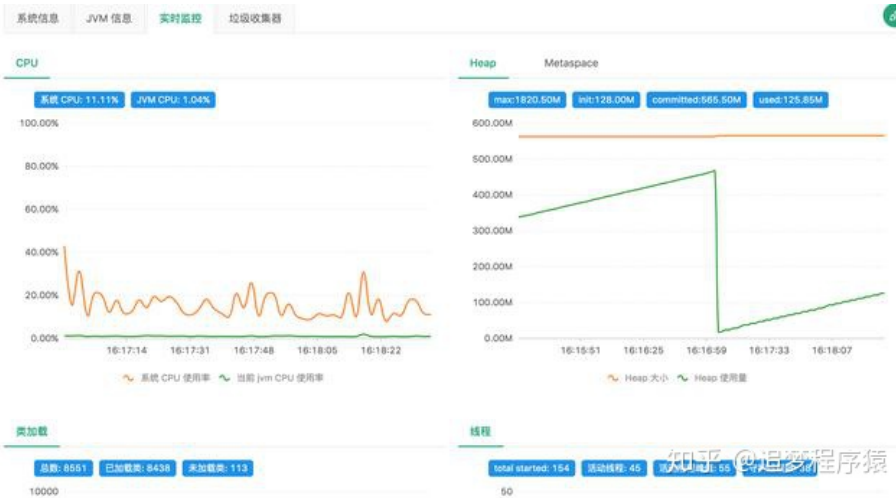
前端采用 React 16 + Antd + Yarn ，后端 Spring Boot + Java 1.8。具体使用方式可以到 github 仓库 README 页面查看。并且提供了一个在线预览版本，文末有源码地址和在线版本的使用方式。



支持本地 JVM 和远程 JVM 链接。



所有 MBean 的树形展示以及属性、操作的展示。



JVM 运行情况实时展示

- 3. JVM 参数，包括命令行参数和 systemProperties；
- 4. CPU、Heap、Metaspace、类加载、线程的实时折线图；
- 5. 垃圾收集器的种类和回收次数；

下面结合这几部分，说一下 JMX 的使用方式。

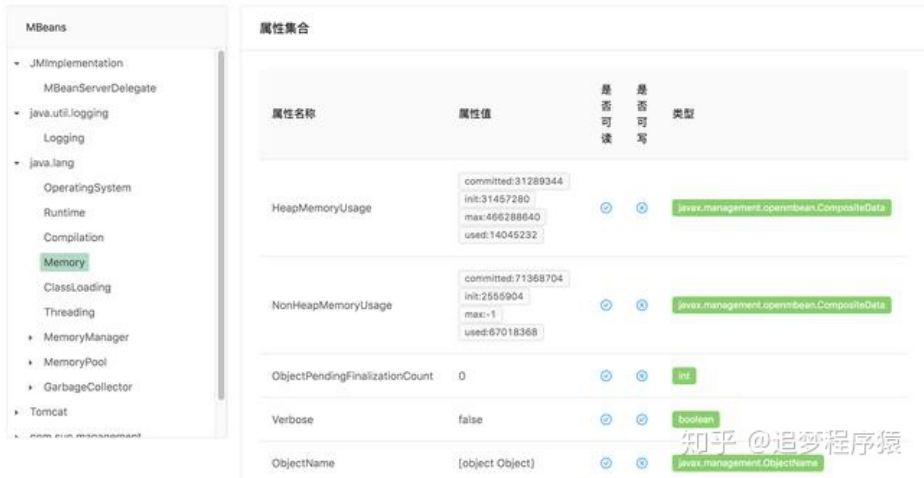
MBean 的获取

正如各种工具里的 MBean 的树形展示方式一样，MBean 本身就是以这种层级关系存在的。

MBean 包含在 Domain 里，Domain 相当于是是一套独立的空间，这个空间里可以定义各种 type，各种 name 的 ObjectName。比如前一篇 JMX 文章里自定义的那个。

通过 ObjectName 可以获取到 MBean 的各种信息，包括属性、操作、通知。

有些属性是简单数据类型，比如 int、long、double、String 类型，另外有些是比较复杂的，比方说 com.sun.management:type=HotSpotDiagnostic 的属性 DiagnosticOptions 就是 javax.management.openmbean.CompositeData 类型。还有的属性的数据类型是 javax.management.openmbean.TabularData。这些都要单独处理。



常用的 MBean

有些指标是监控会用到的，比如内存、CPU、堆空间、线程、类加载情况相关的 MBean。

JDK 提供了一个 ManagementFactory，帮助我们方便的获取常用的 MBean。可以到 java.lang.management 包下找到这个类看一下注释和代码。

OperatingSystemMXBean

可以获取操作系统相关的信息，机器名称、内存使用、CPU使用等信息。

可通过 ManagementFactory.getOperatingSystemMXBean() 方式获取。

RuntimeMXBean

可以获取当前 JVM 的信息，包括 JVM 参数和 JVM 相关的系统参数。

可以通过 ManagementFactory.getRuntimeMXBean() 方式获取。

知乎

首发于
追梦程序猿

可以通过 `ManagementFactory.getMemoryMXBean()` 获取

ThreadMXBean

获取 JVM 线程使用情况，包括活动线程、守护线程、线程峰值等。

可以通过 `ManagementFactory.getThreadMXBean()` 获取。

ClassLoadingMXBean

获取 JVM 类加载情况，包括已加载类、未加载类等。

可以通过 `ManagementFactory.getClassLoadingMXBean()` 获取。

GarbageCollectorMXBean

获取 JVM 垃圾收集器的情况，包括使用的哪种垃圾收集器以及回收次数等等。

可以通过 `ManagementFactory.getGarbageCollectorMXBeans()` 获取，注意，这里获取到的是一个集合，因为垃圾收集器分为老年代和新生代两个。

除了以上几个常用的 MBean，还有很多其他的。有些在 `ManagementFactory` 类里已提供了，另外还有很多需要自己通过 `ObjectName` 获取。

作者：古时的风筝

链接：juejin.im/post/68569495...

发布于 2020-08-04 12:09

[JMX](#)[Java 虚拟机 \(JVM\)](#)[Java 程序员](#)

文章被以下专栏收录



追梦程序猿

长期分享Java架构、面试以及技术文章

推荐阅读



JVisualVM 进行性能分析及调优

ITGeGe



面试官问Java垃圾收集工作原理，是看你能否有效管理JVM...

知乎用户Di4RhU

听说 JVM 性能优化很难？今天我小试了一把！

对于 Java 开发的同学来说，JVM 性能优化可以说是比较难掌握的知识点。这不仅因为 JVM 性能优化需要掌握晦涩难懂的 JVM 知识，还因为 JVM 性能优化很难有使用场景。这导致了许多人对 JVM 性...

陈树义

发表于架构师指南



写下你的评论...



经济的本质是劳动

02-13

非常给力，清晰易懂



赞



知乎用户

01-17

学习了👍



赞



kenshine

2021-12-13

学习了



赞



lwq1989

2021-10-29



赞

