基于JMX监控下的JBoss数据库连接池计算机二级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/578/2021_2022__E5_9F_BA_E4_BA_8EJMX_E7_c97_578608.htm 2009年下半年全国计算机等级考试你准备好了没?考计算机等级考试的朋友,2009年下半年全国计算机等级考试时间是2009年9月19日至23日。更多优质资料尽在百考试题论坛 百考试题在线题库 一、JMX简介 JMX(Java Management Extensions，Java管理扩展)是一个为应用程序植入管理功能的框架。JMX是一套标准的代理和服务，实际上，用户可以在任何Java应用程序中使用这些代理和服务实现管理。 Jboss的成功就在于采用了JMX，从零开始、模块化开发了Jboss服务器和容器，实现了模块化、嵌入式的技术架构。JMX作为集成中心(总线)，可以很方便的热插拔新的模块和组件。JMX服务可以通过HTTP、RMI、SNMP等多种协议进行访问，使其适合作为一个网络管理、监控平台的技术架构。 二、JMX构架中的各层及相关的组件 1.工具层(Instrumentation Level) (a)MBeans(标准的，动态的，开放的和模型MBeans) (b)通知模型：Notification、NotificationListener等类 (c)MBean元数据类：Attribute、Opreator等类 2.代理层(Agent Level) (a)MBean Server (b)代理服务。如jboss jmx-console下的HtmlAdaptorServer等。 MBean：是Managed Bean的简称。在JMX中MBean代表一个被管理的资源实例，通过MBean中暴露的方法和属性，外界可以获取被管理的资源的状态和操纵MBean的行为。事实上，MBean就是一个Java Object，同JavaBean模型一样，外界使用自醒和反射来获取Object的值和调用Object的方法，只是MBean更为复杂和高

级一些。 MBeanServer：MBean生存在一个MBeanServer中。MBeanServer管理这些MBean，并且代理外界对它们的访问。并且MBeanServer提供了一种注册机制，是的外界可以通过名字来得到相应的MBean实例。 JMX Agent：Agent只是一个Java进程，它包括这个MBeanServer和一系列附加的MbeanService。当然这些Service也是通过MBean的形式来发布。 Protocol Adapters and Connectors JMX Agent通过各种各样的Adapter和Connector来与外界(JVM之外)进行通信。同样外界(JVM之外)也必须通过某个Adapter和Connector来向JMX Agent发送管理或控制请求。 Adapter和Connector的区别在于：Adapter是使用某种Internet协议来与JMX Agent获得联系，Agent端会有一个对象(Adapter)来处理有关协议的细节。比如SNMP Adapter和HTTP Adapter。而Connector则是使用类似RPC的方式来访问Agent，在Agent端和客户端都必须有这样一个对象来处理相应的请求与应答。比如RMI Connector。JMX Agent可以带有任意多个Adapter，因此可以使用多种不同的方式访问Agent。 三、监控jboss数据库连接池的实现 TABLE cellSpacing=0 borderColorDark=#ffffff cellPadding=2 width=400 align=center borderColorLight=#999999 border=1&gt. import java.util.Iterator. import java.util.Properties. import java.util.Set. import javax.management.MBeanInfo. import javax.management.MBeanOperationInfo. import javax.management.MBeanParameterInfo. import javax.management.ObjectInstance. import javax.management.ObjectName. import javax.naming.InitialContext. import

org.jboss.jmx.adaptor.rmi.RMIAdaptor. public class DataSourceManger { public static void main(String[] args) { // TODO 自动生成方法存根 try { // Get RMIAdaptor Object Properties pro = new Properties(). pro.setProperty("java.naming.factory.initial", "org.jnp.interfaces.NamingContextFactory"). pro.setProperty("java.naming.provider.url", "jnp://localhost:1099"). pro.setProperty("java.naming.factory.url.pkgs", "org.jboss.naming:org.jnp.interfaces"). InitialContext ic = new InitialContext(pro). RMIAdaptor server = (RMIAdaptor) ic.lookup("jmx/rmi/RMIAdaptor"). ObjectName name = new ObjectName("jboss.jca:name=jdbc/baosigpo,service=ManagedConnectionPool") . ObjectName lname = new ObjectName("jboss.jca:name=jdbc/baosigpo,service=LocalTxCM"). String AvailableConnectionCount = server.getAttribute(name, "AvailableConnectionCount").toString(). System.out.println("==========avlide==========" AvailableConnectionCount). String InUseConnectionCount = server.getAttribute(name, "InUseConnectionCount").toString(). System.out.println("==========InUseConnectionCount========" InUseConnectionCount). String ConnectionCreatedCount = server.getAttribute(name, "ConnectionCreatedCount").toString(). System.out.println("==========ConnectionCreatedCount==========" ConnectionCreatedCount). String

ConnectionDestroyedCount = server.getAttribute(name, "ConnectionDestroyedCount").toString(). System.out .println("==========ConnectionDestroyedCount========== ConnectionDestroyedCount). ConnectionDestroyedCount = server.getAttribute(name, "ConnectionDestroyedCount").toString(). System.out .println("==========ConnectionDestroyedCount========== ConnectionDestroyedCount). String[] argTypes = new String[0]. int i=0. i=Integer.parseInt(AvailableConnectionCount). Object opReturn=null.