将Java程序注册成系统服务Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E5_B0_86Java_E7_A8_8B_c104_644501.htm 你是不是在找将java程序注册成系统服务的方法？试试java Service Wrapper这个工具吧，你可以从这个网站上面下载你喜欢的版本：http://wrapper.tanukisoftware.org/ ，java Service Wrapper提供了适合市面上流行的操作系统的版本。 使用Wrapper将java程序注册成系统服务有三种方式可供选择： 第一种是使用WrapperSimpleApp 这个帮助类来运行你的程序，这个是最简单的方法，也是官方推荐使用的方式，但是这样可能会对你的程序有改动，如果你在项目初期就开始考虑的话，这个方法还是不错的。像JBoss也是使用这种方式。 第二种方式是使用WraperStartStopApp这个类来实现功能，这个方法适合那些通过ClassA类来负责启动服务，ClassB类来负责停止服务的应用场景。 我使用的是第三种方式，这种方式好处是对程序改动比较小，只要让你的启动类实现WrapperListener接口，并实现接口中的 start（String[] args） 和 stop（String [] args）方法，然后通过WrapperManager来启动。其他的一些配置比如要运行的主类全名、java类路径、依赖java库的路径、还有服务显示的名称，都可以通过配置文件conf/wrapper.conf来配置，相对来说比较灵活，像我目前在做的RCP项目有自动更新功能，更新下来的插件要比那些原来的插件的版本号要更新，虽然说会定期删除那些过期的插件，但有时还是会产生延迟，那么配置文件里面配置的java类路径必须也要链接到最新的插件的地址，我是通过一个java类来管理这个wrapper.confg

文件，如果有更新的插件，通过java类来得到最新插件的路径，将这些信息写入到 wrapper.confg文件中，这样就能保证配置文件中的类路径是最新的了。 下面是程序的结构 这里主页介绍一下wrapper.conf的配置，这个配置文件是java常用的属性文件格式， wrapper.java.command=java： 指定要运行的java，如果你不想设置环境变量的话，你也可以指定JDK的bin文件路径 wrapper.java.mainclass=test.Main：指定要运行的类，这个类必须实现WrapperListener接口和接口中的start和stop方法，通过WrapperManager类来初始化服务。如果启动服务过程中出现与不能取得JVM信息的情况，可能是接口实现的问题。 wrapper.java.classpath.1=……/lib/wrapper.jar：配置java的类路径，这里的将wrapper.jar也包含在内，这里可以设置参数的位置，而且这个位置必须得从1 开始，不能跳过，必须顺序指定，指定类路径的时候还有根据依赖关系来排列，被依赖的排在前面，否则会出现ClassNotFoundException的错误，这里支持觉得路径和相对路径，也支持通配符"*"，比如wrapper.java.classpath.1=……/lib/wrapper*，不过这个通配符只能用于匹配文件名，不能用于匹配文件夹名称。 wrapper.java.library.path.1=……/lib：指定Wrapper自带的类库文件存放文件夹，比如Wrapper.DLL文件等，只要指定到对应的上级目录名称就行，支持通配符。
wrapper.java.library.path.1=……/lib：指定Wrapper自带的类库文件存放文件夹，比如Wrapper.DLL文件等，只要指定到对应的上级目录名称就行，支持通配符。 wrapper.app.parameter.1=：指定运行类的main方法参数。 wrapper.daemonize=TRUE：将服务注册成守护线程，就算程序关闭的话不影响服务的运

行 wrapper.ntservice.hide-console=false：不显示控制台 wrapper.filter.trigger.1= ， wrapper.filter.action.1 ：指定过滤器和触发器，可以对控制台的输出信息进行监听，然后触发相应的操作 wrapper.disable_shutdown_hook=TRUE：是否禁用 "关闭Hook" ， 关闭的话在出现一般异常的情况下面可以忽略掉异常继续执行 wrapper.console.loglevel=INFO：配置控制台的显示信息的级别，NONE不显示任何输出信息，FATAL只显示致命的错误消息，ERROR显示所有的错误消息，STATUS显示服务状态的改变，包括服务启动和停止等信息，INFO显示所有程序输出的信息和JVM显示的信息，如果程序无法正常启动，可以使用DEBUG显示详细的调试信息。 wrapper.logfile.loglevel=INFO：配置日志记录文件要记录的输出信息的级别，参数值和wrapper.console.loglevel功能一致 wrapper.logfile.maxsize=0：配置日志文件的最大大小，如果为0表示不限制日志文件的大小，支持标记符，“ k ”代表KB，“ m ”代表MB，如果要设置最大大小为100KB的话可以这样：wrapper.logfile.maxsize=100k wrapper.console.title=Wrapper Demo ：控制台窗口显示标题， wrapper.ntservice.name=testwrapper： 系统服务的名称， wrapper.ntservice.displayname=Wrapper Demo：在服务管理中显示的名称 wrapper.ntservice.description=Wrapper Demo的介绍信息： 在服务管理器显示服务的描述信息 wrapper.ntservice.starttype=AUTO_START： 配置服务启动方式，可以选择AUTO_START（自动）和DEMAND_START（手动）两种方式。默认为自动。 前几天在看Jetty源代码的时候发现它也是使用Wrapper注册成系统服务，使用的是第三种方

式，可以参考一下 import java.io.PrintStream. import org.mortbay.jetty.Server. import org.mortbay.start.Main. import org.tanukisoftware.wrapper.WrapperListener. import org.tanukisoftware.wrapper.WrapperManager. public class JettyServiceWrapperListener implements WrapperListener { private static Server __server = null. public void controlEvent(int event) { if ((WrapperManager.isControlledByNativeWrapper()) || ((event != 200)