Java和.NET互操作究竟有什么用 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/264/2021_2022_Java_E5_92_8CNE_c67_264827.htm 并非所有的开发者都清楚，时下最流行的两个程序运行环境（Java虚拟机JVM和.NET通用语言运行时CLR）事实上就是一组共享的类库。不论是JVM还是CLR，都为程序代码的执行提供了各种所需的功能服务，这其中包括内存管理、线程管理、代码编译（或Java特有的即时编译JIT）等等。由于这些特性的存在，在一个操作系统中，如果程序同时运行在JVM和CLR两种环境之上，由于任何一个进程都可以加载与之对应的任何共享类库，这使得相应的操作将变得非常繁琐。 然而，当话题讨论到这些问题的时候，大多数开发者都会停下来，向一侧仰着头，非常认真的问道“ 可是……这样的互操作对我们来说究竟有什么用？ ”近些年来，基于Java平台的程序开发，一直都有为数众多的API类库和新技术为其提供强大的支持。与此同时，.NET的通用语言运行时CLR，天生就具备Windows操作系统所提供的那些丰富的编程支持。在Windows操作系统环境下，常有许多Windows编程中易于实现的功能目前却很难使用Java语言编程实现，然而有的时候，使用Java语言实现特定功能较之Windows编程却更为简洁。这是在Java编程中，使用Java本地接口JNI技术实现互操作时的通常看法，同时这对于Java的开发者来说也应当是非常熟悉。可能会让开发者感觉有所陌生的，是那些尝试在Java虚拟机中实现.NET编程语言特性的想法，例如在最新的.NET 3.0中，包含工作流、WPF和InfoCard等广受关注的特性，或是在.NET过程中使用Java虚

拟机提供的工具，比如说部署Java语言编写的那些包含复杂业务逻辑的Spring组件，或者实现通过ASP.NET访问JMS消息队列这样的功能。 加载动态链接库以及与底层代码托管环境进行交互，是解决互操作问题所面临的两个不同问题，然而，每一项操作都为之提供了标准的应用程序接口来完成这样的功能。举例来说，下面列出的非托管C 代码来自于Java本地接口JNI的官方文档，目的是利用标准过程（相关的代码句柄在JNIHosting子目录里以InProcInterop方案的一部分存在，构建它的最好方法是在命令行里用指向JDK 1.6目录位置的JAVA_HOME环境变量来操作。）创建基于Java虚拟机的函数调用：#include "stdafx.h"#include int _tmain(int argc, _TCHAR* argv[]){ JavaVM *jvm. /* 表示一个Java虚拟机 */ JNIEnv *env. /* 指向本地方法调用接口 */ JavaVMInitArgs vm_args. /* JDK或JRE 6的虚拟机初始化参数 */ JavaVMOption options[4]. int n = 0. options[n ].optionString = "-Djava.class.path=.". vm_args.version = JNI_VERSION_1_6. vm_args.nOptions = n. vm_args.options = options. vm_args.ignoreUnrecognized = false. /* 加载或初始化Java虚拟机，返回Java本地调用接口 * 指向变量 env */ JNI_CreateJavaVM(amp.env, &amp.vm_args). // 传入C 所需的参数 /* 使用Java本地接口调用 Main.test 方法 */ jclass cls = env->FindClass("Main"). jmethodID mid = env->GetStaticMethodID(cls, "test", "(I)V"). env->CallStaticVoidMethod(cls, mid, 100). /* 完成工作 */ jvm->DestroyJavaVM(). return 0.} 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com