Interoperability Between JVM & CLR

Abstract

The real concept driving this article is to develop solutions using the .NET or Java Framework that interoperate with heterogeneous systems or even mutually communicate with each other. Java Virtual Machine (JVM) is exposing Java Native Interface (JNI), which allows other programs to control JVM in a manner, for instance, load classes, create instances and run methods.

This example shows the amazing tactics of mingling both C# and Java code in one source code file and producing the desired result. Nowadays, organizations work on multiple projects simultaneously, and the corporate infrastructure usually is configured with couple of platforms such as .NET, JAVA and PHP to fulfill the developer's requirements. So, it is good to interoperate or set up communication across these technologies in order to save human efforts and computer resources, because one algorithm developed on the .NET platform could be packed as API or DLL files and run across diverse platforms.

Essential

The aspirant should be knowledgeable in both Java and C# programming languages, as well as having deep understanding of JVM and CLR internal. This operation requires the subsequent software to be installed:

- Jni4net
- Jdk
- JVM
- Eclipse (optional)
- .net framework
- Vs studio 2010 (optional)

Interoperability

Interoperability enables communication, data exchange, or program execution among diverse systems in a way that requires the user to have little awareness of the underlying operations of those systems. The programmers can take advantage of the power of the Java platform by applying JNI, without having to abandon their investments in legacy code. Because the JNI is a core part of the JVM, programmers can address interoperability issues once, and expect their solution to work with all implementations of the Java platform.

Questions that often occur in the mind of developer include: why is interoperability so important? What kind of issues it is precisely addressing? To what extent is interoperability technology between Java and. NET beneficial? Here, the advantages outlined to justify interoperability are as following:

- **Migrations:** migration has to be well-planned and carefully executed when s system is updated or replaced, and often involves moving an application by a few parts at a time. This way of dividing a system for migration purposes often creates a demand for interoperability because some parts that have been migrated might need to communicate with others that have not.

- **Reusability:** most established companies have a number of legacy systems. By the term *legacy*, I mean technology that's not being keenly developed today. For example, a system located in the data center that's still in production but no longer offers a strategic advantage to the company is a legacy system. A plan to move these systems to a new platform might be a longer-term strategy. A solution that has the ability to interoperate with these systems has the potential to extend the life of these systems, and more importantly, the knowledge of the developers who work with them.

- **Adoption:** when organizations want to deploy a new technology such as the .NET Framework, it's rare that they simply replace an entire application or system. In many cases, a replacement is normally triggered by a pilot project. Such a pilot tends to be a short-term project with an aim to prove that the technology can work well with existing systems and applications. The ability for this pilot project to interoperate with existing production systems is imperative, and in many cases, can often determine its success.

Java Native Interface (JNI)

Sometimes, a situation arises where Java or .NET alone does not fit into the needs of your application. While we can write and execute applications entirely in Java and .NET framework independently, programmers use the JNI to write native methods to handle those situations when an application cannot be written in Java and .NET independently. The Java Native Interface framework is devised to enable Java code running in a Java Virtual Machine (JVM) to call, and to be called by, native applications and libraries written in other programming languages, for instance, .NET, C++ and Assembly. JNI can also modify an existing application which is developed and executed under CLR- to be accessible to Java applications. In simple terms, we can say that we can call a .NET application and library in Java and vice-versa by using JNI. Native code runs relatively faster than JVM code, so such implementation can assist at a great extent to solve complicated and time-critical operations.

Java, native languages, and .NET have their own data type infrastructure. During interoperability or communication of these diverse languages to each other, it is necessary to comply with a common data type scheme so that all participant languages can send messages to each other and follow data type semantics. The following table describes the primitive types in the Java programming language and the corresponding types in the JNI.

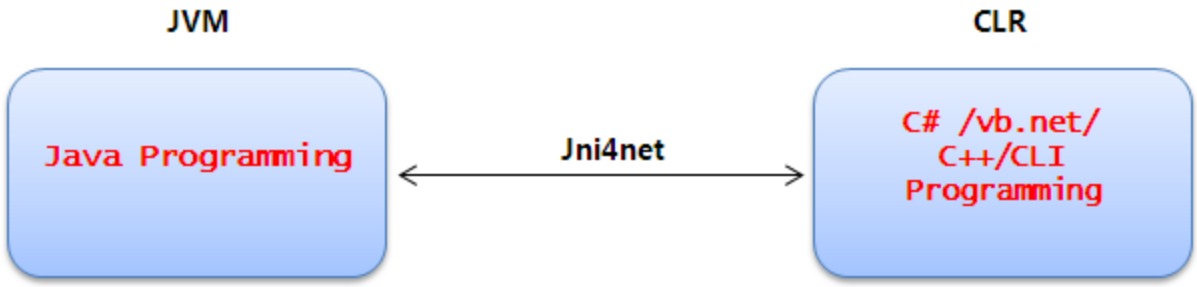
Native Fields	Java Fields	Descriptor
Jboolean	Boolean	Z
Jbyte	Byte	B
Jchar	Char	C

Jshort	Short	S
Jint	Int	I
Jlong	Long	J
Jfloat	Float	F
jdouble	Double	D

The Java platform is implemented on top of the host platform. So, it is necessary to allow Java applications to work closely with other languages written in native code so that developer can adopt the Java platform to build applications that were traditionally written in .NET and C++. The JNI is specially designed to interoperate or combine Java applications with CLR or native code. JNI typically offers two types of native code: native application and library. We can utilize the JNI to write native methods that allow Java applications to call functions implemented in native libraries. Java applications call native methods in the same way that they call methods implemented in the Java programming language.

Jni4net Framework

Sometimes circumstances demand that JAVA and .NET technology send messages or communicate with each other in order to save human effort or reduce programming glitches. It is not necessary that a company infrastructure is reliant particularly upon the J2EE or .NET framework. An algorithm written in Java could be consumed or manipulated in the .NET framework or vice-versa. Jni4net is a mechanism that allows a Java program to call a function in a C# program and a C# program to call a method in a Java program. The jni4net framework includes both .NET, FCL and JDK core classes in order to possibly reflect technology implementation across the boundaries.



So, jni4net could be conceived as API, which is frequently used in Java or .NET technology. Apart from that, this framework offers a couple of other functionalities such as garbage collection, automatic proxy generation and inter-process communication. The jni4net framework also addresses the interoperability mechanisms similarly supported by other languages. This is, however, not designed for a particular implementation of the Java virtual machine. Rather, it is a native interface that can be supported by every implementation of the Java virtual machine.