Accessing EJBs from .NET using JBoss® and JNBridgePro™

Version 11.0

SPANNING JAVA & .NET

jnbridge.com

October 7, 2020

Accessing EJBs from .NET using JBoss and JNBridgePro

The following example shows how to access Enterprise Java Beans from C# using Visual Studio and the JNBridgePro Visual Studio proxy generation plug-in. The EJBs will be running on JBoss 4.3.0 or later. We assume that JNBridgePro 6.0 or later, and .NET framework 2.0 (3.0/3.5) or .NET 4.0 or later, have been installed on a Windows machine and that you are familiar with the use of Visual Studio 2005/2008/2010/2012/2013/2015/2017/2019 and JNBridgePro and have read the documentation. We also assume that JBoss 4.3.0 has been installed on a compatible machine, which may be the same machine as the .NET machine, or a different machine reachable on the network. In addition, the .NET machine must have a suitable JRE installed (if the .NET machine is also hosting JBoss, then the required JDK can be used).

If using VS 2010 or later, the directory, `..\JBoss43\JBoss43Example_Net40\`, contains a VS 2010 project already targeted for .NET 4.0. (The project can also be consumed by VS 2012, 2013, 2015, 2017, or 2019.) The directory, `..\JBoss43\JBoss43Example\`, contains a VS 2005 project targeting .NET 2.0. The VS 2005 project can, of course, be upgraded to later versions of Visual Studio by the user.

Deploying the Example EJB File

Install the example EJB file by copying `calculator.jar` from …`\demos\J2EE-Examples\JBoss43` to the JBoss `deploy` directory, e.g. `C:\Program Files\jboss-eap-4.3\jboss-as\server\production\deploy`. Please restart the server.

Building the Proxy DLL

Before opening Visual Studio, it may be wise to copy the example directory, `JBoss43`, to another location to avoid access privilege problems. Also be aware that some files may require setting read/write access.

Build the proxy assembly DLL as follows:

1. Start up Visual Studio and load the solution, `JBoss43\JBoss43Example\JBoss43Example.sln`. Select the menu item File→Add→New Project… and select project type JNBridge. Type in any project name. After the JNBridgePro project has been added to the solution, click on the object `DotNetToJavaProxies.jnb` in the Solution Explorer to open the proxy tool interface.
2. Select the menu item JNBridgePro→JNBridgePro Java Options… and verify that the box Start Java automatically has been checked, and that the Java configuration data is correct. Close the dialog box. This dialog may automatically open if any configuration data is incomplete.
3. Select the menu item JNBridgePro→Edit Classpath… and add the following files to the classpath:
   - **`calculator.jar`** found in the root directory for this example, `..\J2EE-Examples\JBoss43`. This is the same file deployed to the JBoss `deploy` directory (it contains the classes CalculatorHome and Calculator, used in the C# code).
   - **`jbossall-client.jar`** found in the JBoss. `client` directory (it contains the classes EJBHome, EJBObject, InitialContext and other EJB and JNDI related support classes). This jar file can be copied to a location on the .NET machine.

Close the Edit Class Path dialog box

4. Select the menu item JNBridgePro→Add Classes from Classpath… and add the following classes. For each class added, make sure that the box Include Supporting Classes is checked.
   - com.jnbridge.demo.calculator.CalculatorHome
   - com.jnbridge.demo.calculator.Calculator
   - javax.naming.Context
   - javax.naming.InitialContext
   - javax.naming.NamingException
   - javax.rmi.PortableRemoteObject
5. Click OK to add the classes and all supporting classes to the Environment pane. Check all the items in the environment (use the menu item JNBridgePro→Check All in Environment) and click on the Add button to move them all to the Exposed Proxies pane.
6. Select the menu item JNBridgePro→Build to build the proxy assembly.

Building and Running the Client Application

You've now completed the task of building an assembly containing proxies of the calculator EJB. The next step is to reference the proxy assembly and JNBridgePro components from the .NET console application, `JBoss43Example`, and configure the .NET-to-Java bridge for run-time execution.

1. Using the Add Reference… dialog in Visual Studio, add the project that created the proxy assembly as a reference. Also, add a reference to `JNBShare.dll`, found in the JNBridgePro install directory, e.g. `C:\Program Files\JNBridge\JNBridgePro v11.0\4.0-targeted\JNBShare.dll`.
2. Open the source file, `Program.cs`, and provide the values to connect to the JBoss server for the string variables `hostName`, `portNumber`, `login` and `passWord`. If the JBoss implementation does not need authentication credentials, then leave the credentials as empty strings.
3. Open the `App.config` XML file and uncomment the element `dotNetToJavaConfig`. Edit these attributes :
   - **`jvm`** points to the Java Virtual Machine, `jvm.dll`. The .NET machine (where you're building this example) must have a JRE installed. If this machine is also running JBoss, then the JDK client JVM can be used.
   - **`jnbcore`** points to `jnbcore.jar`, found in the `jnbcore` directory where JNBridgePro is installed.
   - **`bcel`** points to `bcel-5.1-jnbridge.jar`, found in the `jnbcore` directory where JNBridgePro is installed.
   - **`classpath`** points to the two jar files used to create the proxy assembly, `jbossall-client.jar` and `calculator.jar`.
4. Build the project. There should be no errors.
5. Run the .NET application. Output showing the results of calculations will be displayed.

Notes on this Example

- The default .NET-to-Java bridge configuration for this example is to use the shared memory transport. The `App.config` file can be modified to use tcp/binary. See the User's Guide for details.
- Instead of using the JNBridgePro Visual Studio plug-in, it is possible to use the JNBridgePro stand-alone proxy generation tool to create the proxy assembly. The resulting assembly can then be referenced from the .NET project in VS.
- While this example was tested against JBoss 4.3.0, it should work with any JBoss version above 4.0.5.
- It is strongly recommended that you use the same version of Java that is recommended for the particular version of JBoss that you are using.