



SPANNING JAVA & .NET

Accessing EJBs from .NET using IBM WebSphere and JNBridgePro™

Version 6.0

JNBridge, LLC
www.jnbridge.com

COPYRIGHT © 2002–2011 JNBridge, LLC. All rights reserved.

JNBridge is a registered trademark and JNBridgePro and the JNBridge logo are trademarks of JNBridge, LLC.

Java is a registered trademark of Oracle and/or its affiliates. Microsoft, Windows, and Visual Studio are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries. IBM and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States and other countries.

All other marks are the property of their respective owners.

May 2, 2011

Accessing EJBs from .NET using WebSphere 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 IBM WebSphere 6 or later. We assume that JNBridgePro 4.0 or later, and .NET framework 2.0 or later, have been installed on a Windows machine and that you are familiar with the use of Visual Studio 2005/2008 and JNBridgePro and have read the documentation. We also assume that WebSphere 6, including the examples, 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 WebSphere 6, then the IBM JRE can also be used with modifications to `Program.cs` and `App.config` (see below).

If using VS 2010, the directory, `..\WebSphere6\WebSphere6Example_Net40\`, contains a VS 2010 project already targeted for .NET 4.0. The directory, `..\WebSphere6\WebSphere6Example\`, contains a VS 2005 project targeting .NET 2.0. The VS 2005 project can, of course, be upgraded to VS 2008 by the user.

Deploying the Example EJB

This example uses a managed calculator EJB already installed on the WebSphere 6 in the Samples Gallery. The source directories for this example bean can be found, on Windows, at `[WS6InstallDir]\profiles\AppSrv01\samples\src\TechSamp\BasicCalculatorEJB\ejbModule\com\ibm\websphere\samples\technologysamples\ejb\stateless\basiccalculator.ejb`.

The Java archive file for the managed bean is `BasicCalculatorEJB.jar` and can be found in the root directory for this example. Please make sure that the WebSphere 6 Samples Gallery is installed and the calculator bean is deployed. Please use the WebSphere Samples Gallery manager application to deploy the EJB.

Building the Proxy DLL

Before opening Visual Studio, it may be wise to copy the example directory, `WebSphere6`, 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, `WebSphere6\WebSphere6Example\WebSphere6Example.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:

- **BasicCalculatorEJB.jar** found in the root directory for this example, ..\J2EE-Examples\WebSphere6. This archive contains the BasicCalculator and BasicCalculatorHome classes used in the C# code.
- **j2ee.jar**, **sibc.jms.jar**, **sibc.jndi.jar** and **sibc.orb.jar** found in the root directory for this example. These archives constitute the WebSphere stand-alone thin-client (it contains the classes EJBHome, EJBObject, InitialContext and other EJB and JNDI related support classes).

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.ibm.websphere.samples.technologysamples.ejb.stateless.basiccalculatorejb.
BasicCalculator
 - com.ibm.websphere.samples.technologysamples.ejb.stateless.basiccalculatorejb.
BasicCalculatorHome
 - javax.naming.Context
 - javax.naming.InitialContext
 - javax.naming.NamingException
 - javax.rmi.PortableRemoteObject
 - java.rmi.RemoteException
 - javax.ejb.CreateException
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, WebSphereExample, 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 JNShare.dll, found in the JNBridgePro install directory, e.g. C:\Program Files\JNBridge\JNBridgePro v6.0\2.0-targeted\JNShare.dll. If using VS 2010 and targeting .NET 4.0, use C:\Program Files\JNBridge\JNBridgePro v6.0\4.0-targeted\JNShare.dll.
2. Open the source file, Program.cs, and provide the values to connect to the WebSphere server for the string variables `hostName`, `portNumber`, `login` and `password`. If the WebSphere implementation does not need authentication credentials, then leave the credentials as empty strings.

3. If you are planning on using the IBM private JRE, please comment out this line in `Program.cs`:

```
h.put((java.lang.JavaString)"com.ibm.CORBA.ORBInit"  
    ,(java.lang.JavaString)"com.ibm.ws.sib.client.ORB");
```

This line must be present if a Sun JRE is being used.

4. 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 WebSphere, then the IBM private JRE can be used, see item 3, above, and comment, below.
 - **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 jar files used to create the proxy assembly:
`BasicCalculatorEJB.jar`, `j2ee.jar`, `sibc.jms.jar`, `sibc.jndi.jar` and `sibc.orb.jar`. If the IBM private JRE is being used, do not include `sibc.orb.jar`. If a Sun JRE is being used, you must include `sibc.orb.jar`.
5. Build the project. There should be no errors.
6. 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` or `HTTP/SOAP`. 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.
- If using a Sun JRE, it is recommended that you use Java 5.