Table of contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. Before you start ................................................................................... 2
2. Getting started..................................................................................... 4
3. Creating the DB2 database project ..................................................... 7
4. Creating the DB2 C# procedures ...................................................... 9
5. Creating the DB2 VB procedure ....................................................... 22
6. Altering the body of CLR procedures .............................................. 29
7. Altering the signature of CLR procedures ....................................... 31
8. Summary and resources ................................................................... 35
9. Feedback........................................................................................... 36
Section 1. Before you start

About this tutorial

This tutorial demonstrates the development of DB2 Common Language Runtime (CLR) procedures that are coded in C# and Visual Basic. To show you the ease of building such procedures, this tutorial guides you through the steps required to develop a sample HR application for a small company using the DB2 "Stinger" Development Add-In for Visual Studio Technical Preview.

You will learn how to:

° Create DB2 class assembly projects in C# and Visual Basic.
° Develop DB2 CLR methods using ADO.NET and the DB2 managed provider.
° Create a DB2 database project and add a DB2 data connection.
° Install CLR assemblies using your database project.
° Catalog CLR procedures using your database project.
° Alter the body and signature of CLR procedures and their DB2 catalog information.

Should I take this tutorial?

You should take this tutorial if you want to:

° Develop and manage DB2 CLR procedures using C# and Visual Basic.
° Learn about the DB2 database project template in Visual Studio .NET
° Learn more about the advanced features of the DB2 "Stinger" Development Add-In for Visual Studio Technical Preview

Prerequisites

To complete the steps in this tutorial, you'll need:

° IBM DB2 V8.1.2 Application Development Client
° DB2 "Stinger" Technical Preview
° IBM DB2 V8.1 Server
° IBM DB2 SAMPLE database
° Microsoft Windows 2000, 2003, or XP with the Microsoft .NET framework V1.1
About the author

Abdul Al-Azzawe is the DB2 Tools architect for Visual Studio .NET. He joined IBM in 1990 and worked in the Toronto lab for ten years as part of the DB2 engine development team with special focus on the Windows platform. In March of 2000, Abdul joined the IBM Silicon valley lab to architect the release of the DB2 integrated SQL debugger, the DB2 Development Center, and the DB2 Development Add-Ins for Visual Studio 6.0. In May of 2002, Abdul joined the IBM San Francisco team to architect the integration with Visual Studio .NET, one of his proudest achievements in his career at IBM.

Abdul has written numerous technical articles and tutorials for the DB2 Developer Domain covering a variety of DB2 application development topics. Abdul appreciates hearing feedback from actual developers about DB2 tools, and welcomes any suggestions for ways to improve these products. Abdul can be reached at abdulh@us.ibm.com.

Trademarks

IBM, DB2, and DB2 Universal Database are registered trademarks of the IBM Corporation in the United States and/or other countries.

Microsoft, Windows, Visual Studio, and CLR are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.
Section 2. Getting started

Application overview

This tutorial takes you through the steps required to develop DB2 CLR procedures for a sample HR application, DB2CLRDemo. The application consists of an EMPLOYEE table, and a set of CLR procedures that access this table: GetEmployeeName, GetDepartmentEmployees, and GetManagers.

The EMPLOYEE table is part of the SAMPLE database that ships with the DB2 Universal Database product. It consists of a list of employee names, salaries, departments, jobs, and hiring dates.

The GetEmployeeName procedure is a C# method that returns as an output parameter the combined first name, middle initial, and last name for a specific employee given his employee number as an input parameter. This procedure demonstrates the process of developing a CLR procedure that uses input and output parameters.

The GetDepartmentEmployees procedure is a C# method that returns the list of all employees as a resultset for a specific department given its department number as an input parameter. This procedure demonstrates the process of developing a CLR procedure that returns a result set.

The GetManagers procedure is a Visual Basic method that returns the list of employees that are managers. The requirement for this procedure will evolve to return all female managers, and then to either return the female or male managers based on an input parameter. This procedure demonstrates the process of altering the body, then the body and signature of CLR procedures.

Configuring your system

Since this is a technology preview, DB2’s DDL (Data Definition Language) support for creating CLR procedures was not completed in time, so we adopted a temporary work-around where CLR stored procedures are cataloged with the DB2 server as having language JAVA and parameter style JAVA.

In order for the DB2 engine to recognize these as CLR .NET procedures and not JAVA, you will need to define the system environment variable, JAVA_IS_DOTNET=1. This system environment variable must be set at the local or remote DB2 for Windows server where the Microsoft .NET framework V1.1 is installed.

Once set, re-boot your system so that the DB2 service will recognize this setting and treat all JAVA cataloged procedures as CLR .NET procedures.
disable all JAVA procedures support. To go back to supporting JAVA procedures, simply delete this system environment variable.

Registering the DB2 add-in

The DB2 "Stinger" Development Add-In for Visual Studio Technical Preview is automatically installed and registered when you install DB2; however, if you installed Visual Studio .NET after installing the DB2 application development client, then you will need to manually register the DB2 managed provider and the Add-Ins. This can be done using the following short-cut: Start => Programs => IBM DB2 => Set-up Tools => Register Visual Studio Add-Ins. This short-cut will attempt to register both the Visual Studio 6.0 and Visual Studio .NET Add-Ins.

To register the DB2 managed provider and development add-in from the command line, run db2nmpreg. To un-register them, run db2nmpreg -u.

Starting Visual Studio .NET


This should launch the IDE and automatically load the DB2 Development Add-In. To verify that the DB2 Development Add-In is loaded, you should see the IBM Explorer entry under the Visual Studio .NET View menu.

Creating the DB2 SAMPLE database

The steps in this tutorial were designed to work with a DB2 for Windows SAMPLE database. You must ensure that the sample database has been created on your DB2 for Windows server.

To create the sample database, run the following command on the DB2 server:

db2samp1

Note: The reason for requiring a local or remote DB2 for Windows server is that this tutorial describes the process for creating CLR (VB / C#) stored procedures that require the Microsoft .NET framework V1.1 in order for them to run.

Adding the DB2 data connection
To access DB2 server catalogs and new DB2 objects, you will first need to add the target DB2 server connection to the data connections folder of the IBM Explorer.

To add a local DB2 database connection:

1. Launch the IBM Explorer by clicking View => IBM Explorer.
2. Select the Data Connections folder of the IBM Explorer, and click Add Connection from the pop-up menu. This opens the Add Database Connection window. Note that by default, the value localhost is shown in the Server field. This allows you to access locally created or locally cataloged databases.
3. Select the database from the Database drop-down combo.
4. Specify the user name and password to use for the database. You can leave these fields blank if you want to use the Windows integrated security, which is your current log-on user name and password.
5. Optionally, specify the Default owner, which is your user ID, to be used as the schema in the wizards.
6. Click Test Connection to ensure that you have proper connectivity.
7. Click OK to add the connection.

To add a data connection for a remote SAMPLE database on a remote DB2 for Windows server, you need to know the remote server host name or IP address, port number, and database alias. Follows these steps:

1. Launch the IBM Explorer using View => IBM Explorer.
2. Select the Data Connections folder of the IBM Explorer, and click Add Connection from the pop-up menu. This opens the Add Database Connection window.
3. In the Server field, enter the remote server host name or IP address followed by a colon and the port number. For example, enter azzawe2.svl.ibm.com:446.
4. In the Database field, enter the name of the remote database. For example, RemoteDB.
5. Specify the user name and password to use for the remote database connection.
6. Optionally, specify the Default owner, which is your user id, to be used as the schema in the wizards.
7. Click Test Connection to ensure that you have proper connectivity.
8. Click OK to add the connection.

Once the data connection is created, the server catalog information will be retrieved in an asynchronous fashion to populate the Tables, Views, Procedures, and Functions folders.
Section 3. Creating the DB2 database project

Creating the DB2 database project

To catalog DB2 CLR procedures and deploy the CLR assemblies to the local or remote DB2 server, create a DB2 database project to automate this process for you.

The DB2 project will contain the set of script files that defines your CLR procedures, and the list of assemblies that are required by these CLR procedures. When you build this project as part of the solution that defines one or more CLR procedures, create the assemblies first, deploy them to the server, and then catalog the procedures.

To create the DB2 database project:

1. Select **File => New => Project** from the Visual Studio .NET menu.
2. In the New Project window, select the IBM Projects folder and select the **DB2 Database Project** template.
3. Enter **DB2CLRDemo** for the project name and specify a location for the project. Click **OK**.

4. From the DB2 Data Connection window, select the remote or local SAMPLE data connection that was added earlier to IBM Explorer.
5. Click **OK**.

A new DB2CLRDemo solution and project will be created for you in the Visual Studio .NET Solution Explorer.

Use this project in the sections that follow to catalog the list of C# and VB procedures.
Section 4. Creating the DB2 C# procedures

Creating the DB2 C# class library

In this section of the tutorial, you will create a C# class library that defines two methods that will be cataloged as DB2 CLR procedures.

With the installation of the DB2 application development client, the Visual C# Projects folder is extended to include a new project template, DB2 Class Library. This new template is based on the standard C# Class Library template with the addition of an automatic project reference to the IBM.Data.DB2 managed provider assembly, and an example of a simple CLR method.

To create the DB2 Class Library C# project:

1. Select the DB2CLRDemo solution and click Add => New Project.
2. In the Add New Project window, expand the Visual C# Projects folder and choose the DB2 Class Library template.
3. Enter DB2CSharpDemo for a project name and specify a project location.
4. Click OK.
5. Save All.

The DB2CSharpDemo C# class library project is now added to your DB2CLRDemo solution. Note that the class library project has an automatic reference to the DB2 managed provider assembly, IBM.Data.DB2.
When you create a new class library project using the DB2 Class Library project template, a `DB2Class1.cs` file is also created containing a default CLR method that has the DB2 required signature for a CLR procedure, namely a public class with a public static void method.
Setting the projects dependencies

Since we now have multiple projects in our solution, it is important to setup the build order of these projects to ensure correct application build and deployment. The CLR methods and assemblies must be built first before they can be deployed and cataloged on the DB2 server.

To set the project dependencies and build order:

1. Right-click the DB2CLRDemo project and select Project Dependencies.
2. In the Project Dependencies window, select the DB2CLRDemo project from the Projects drop-down list and check DB2CSharpDemo in the Depends on list.
3. Click **OK**.

This ensures that the DB2CSharpDemo is built before the DB2CLRDemo project.

---

### Creating the GetEmployeeName C# procedure

Create the first simple C# procedure, `GetEmployeeName`. This procedure takes an employee number as input and returns a formatted employee name as output. Proper exception handling is used to accommodate any potential errors.

To create the `GetEmployeeName` procedure:

1. Expand the **DB2CSharpDemo** project.
2. Double click the **DB2Class1.cs** project. This will allow you to edit the code for this class.
3. Modify the default `Procedure1` method as follows:

```csharp
public static void GetEmployeeName(
```
String inEmpID,  
out String outEmpName)
{
    DB2Command myCommand = DB2Context.GetCommand();
    outEmpName = "No Data!";
    try
    {
        myCommand.CommandText =  
            "SELECT FIRSTNAME, MIDINIT, LASTNAME FROM EMPLOYEE" + 
            "WHERE EMPNO = " + inEmpID + ";";
        DB2DataReader reader = myCommand.ExecuteReader();
        if (reader.Read())
        {
            outEmpName = reader.GetString(0) + " 
                reader.GetString(1) + ". 
                reader.GetString(2);"
        }
        reader.Close();
    }
    catch (DB2Exception db2e)
    {
        outEmpName = "DB2Exception: " + db2e.Message;
    }
    catch (Exception syse)
    {
        outEmpName = "Exception: " + syse.ToString();
    }
}

4. Save All.
5. Build the solution.

If all goes well, your build should complete successfully.

Code analysis:

DB2 stored procedure, regardless of language, are all invoked within a given client connection context. CLR procedures make use of the DB2Context object to create commands that are pre-initialized with this connection.

After creating the command object, command text was defined to query the relevant employee columns for a given employee using the client supplied employee number passed as input parameter to the stored procedure.

A DB2 data reader is used to execute the command. The GetString(index) method is used on the data reader to lookup the values for the various columns that resulted from executing the command text query. Since it was not our intention to return the result set back to the client, the reader is closed.

Creating the GetDepartmentEmployees C# procedure

Create the GetDepartmentEmployees C# procedure that returns the list of
employees in a given department. The employees list will be returned as a result set from the procedure.

Although you can add another method to the same `DB2Class1.cs` from the previous step, instead, create a new class simply for the purpose of demonstrating how this can be accomplished.

To create the `GetDepartmentEmployees` C# procedure:

1. Right-click the DB2CSharpDemo project and select `Add => Add New Item`.
2. In the Add New Item window, select the `DB2 Procedure Class` project template and name it `CSharpSP.cs`.
3. Click OK. This opens the new `CSharpSP.cs` in the editor.
4. Modify `Procedure1` as follows:

   ```csharp
   public static void GetDepartmentEmployees(
       String inWorkDept)
   {
       // Create new command object from connection context
       DB2Command myCommand = DB2Context.GetCommand();

       // Return all selected department employees as a result set
       myCommand.CommandText = "SELECT * FROM EMPLOYEE " + "WHERE WORKDEPT = " + inWorkDept + ";";
       DB2DataReader rs = myCommand.ExecuteReader();
   }
   ```
5. Save All.
6. Build the solution.

If all goes well, your build should complete successfully.

Code analysis:

DB2 stored procedure, regardless of language, are all invoked within a given client connection context. CLR procedures make use of the DB2Context object to create commands that are pre-initialized with this connection.

After creating the command object, command text was defined to query all the employee columns for those employees belonging to the client supplied department number which is passed to the procedure as an input parameter.

A DB2 data reader is used to execute the command. Note that the close method on the data reader was not invoked. The DB2 engine interprets this as a dynamic result set that is to be returned to the client or the caller of the procedure.

Cataloging the DB2 C# procedures

To catalog the two DB2 C# CLR procedures:

1. Right-click the DB2CLRDemo project and select Add => Add New Item.
2. In the Add New Item window, select the DB2 CLR Procedure template and enter CSharpSPs.db2sp for the Name of the new item.
3. Click **Open**. This launches the DB2 CLR Procedure Wizard.

4. Click **Next** to go to the CLR Method Selection page. Note that the two methods, `GetDepartmentEmployees` and `GetEmployeeName` are automatically recognized and selected.
5. Click **Next** to go to the Specifications page.
6. Select and expand the `GetDepartmentEmployees` method. Enter your user ID for Schema, 1 for Number of result set(s), `GETDEPTEMP` for Specific name, and `Retrieve list of department employees` for Comment.

![Specifications](image1)

7. Select the `inWorkDept` parameter. Modify the default type of `VARCHAR(256)` to `CHARACTER(3)`.

![Specifications](image2)

8. Select and expand the `GetEmployeeName` method. Specify your user ID for
Schema, GETEMPNAME for Specific name, and Retrieve specific employee name for Comment.

9. Select the inEmpID parameter and change the type from VARCHAR(254) to CHARACTER(6).
10. Select the outEmpName parameter and leave the type of VARCHAR(254).
11. Click Finish.
12. Save All.

**Action analysis:**

The DB2 CLR Procedure Wizard is used to automatically detect the list of CLR projects in your solution. Once a project is chosen, it automatically detects the list of classes and methods that qualify as valid DB2 CLR procedure, that is, those that match the signature requirement for a stored procedure.

The wizard automatically determines some default mappings; however, it is strongly recommended that you explicitly map all of your procedures and their parameters. For procedures, map their names and number of result sets (current limit is a maximum of one result set). For parameters, map their names and appropriate data types.

The DB2 project script file is generated based on your user input. All CLR procedures selected in this wizard will be defined in one script file. Additionally, the assembly hosting these CLR methods is also registered with the DB2 database project for the purpose of automating their install and deployment to the DB2 server.

---

**Setting-up the assembly install**

Now that we have defined the two CLR methods, you also need to ensure that the assembly containing the CLR procedures is deployed to the local or remote DB2 server.

The DB2 database project now supports management of assemblies that are automatically deployed to the server at project build time as required. This means that only when the assemblies are modified (or the first time they are built) they are deployed to the server.

To setup the assembly install:

1. Right-click the DB2CLRDemo project and select Assemblies. This opens the CLR Assemblies window.
2. For the DB2CSharpDemo.dll file name, click the [...] to properly locate the assembly. Assuming that you want to deploy the Debug project configuration DLL, select the bin\Debug\DB2CSharpDemo.dll file.
3. Click **OK** to close the CLR Assemblies window.
4. Build solution.
5. Save All.

You should see, among other things, the following in your build output pane:

```
Installation of CLR assembly 'C:\test\DB2CSharpDemo\bin\Debug\DB2CSharpDemo.dll' succeeded.
```
Action analysis:

Visual Studio .NET supports multiple builds configurations. This means that when you build your CLR assemblies, their final build location may be different depending on which configuration you build with. You must explicitly tell the DB2 database project what the actual file location for the CLR assembly DLL. The default location will most likely be incorrect.

The process for choosing the assembly was described. When the solution is built, the CLR assembly will be automatically deployed and installed on the server before the actual CLR procedures are cataloged. Additionally, the DB2 database project performs a timestamp check on the assembly file. If it detects that the assembly is newer than what was deployed last, it redeployes the updated assembly and re-catalogs the affected CLR procedures to ensure that they point to the updated assembly.

Testing the DB2 C# procedures

To test the two C# procedures:

1. From the IBM Explorer, expand the SAMPLE data connection folder.
2. Right-click the Procedures folder and select Refresh. You should see the GETDEPARTMENTEMPLOYEES and GETEMPLOYEEENAME stored procedures.

To test the GETEMPLOYEEENAME procedure:

1. Right-click the GETEMPLOYEEENAME procedure and select Run Procedure.
2. In the Parameter Values window, uncheck Null and specify a value of 000010.
3. Click OK.
4. You should see the GETEMPLOYEEENAME - DB2 Data Results output data grid. Expand the + sign.
5. Expand the OUTPUT entry and you should see CHRISTINE I. HAAS.

You can re-run the procedure test with other input values (e.g. 000020, 000100), including incorrect values (e.g. 123456).

To test the GETDEPARTMENTEMPLOYEES procedure:

1. Right-click the GETDEPARTMENTEMPLOYEES procedure and select Run Procedure.
2. In the Parameter Values window, uncheck Null, and specify a value of D11.
3. Click **OK**.
4. You should see the GETDEPARTMENTEMPLOYEES - DB2 Data Results output data grid. Expand the + sign.
5. Expand the RESULTSET entry and you should see the list of employees in D11.

You can re-run the procedure test with other input values (e.g. D21, E11), including incorrect values (e.g. X00).
Section 5. Creating the DB2 VB procedure

Creating the DB2 VB class library

In this section of the tutorial, we will create a VB class library that defines a method that will be cataloged as a DB2 CLR procedure.

With the installation of the DB2 application development client, the Visual Basic Projects folder is extended to include a new project template, DB2 Class Library. This new template is based on the standard VB Class Library template with the addition of an automatic project reference to the IBM.Data.DB2 managed provider assembly, and an example of a simple CLR method.

To create the DB2 database project:

1. Right-click the DB2CLRDemo solution and select Add => New Project.
2. In the Add New Project window, expand the Visual Basic Projects folder and choose the DB2 Class Library template.
3. Name the project DB2VBDemo and specify a location.
4. Click OK.
5. Save All.

The DB2VBDemo Visual Basic class library project is now added to your DB2CLRDemo solution. When you create a new class library project using the DB2 Class Library project template, a DB2Class1.vb file is also created and it
contains a default CLR method that has the DB2 required signature for a CLR procedure, namely a public class with a public shared sub method.

Setting the projects dependencies

Since we now have multiple projects in our solution, it is important to setup the build order of these projects to ensure correct application build and deployment. The CLR methods and assemblies must be built first before they can be deployed and cataloged on the DB2 server.

To set the project dependencies and build order, follow these steps:

1. Right-click the DB2CLRDemo project and select **Project Dependencies**.
2. In the Project Dependencies window, select **DB2CLRDemo** from the Projects drop-down list. Check both the **DB2CSharpDemo**, and **DB2VBDemo** entries in the **Depends on** list.
3. Click **OK**.

This will ensure that **DB2CSharpDemo** and **DB2VBDemo** will be built before the
Creating the GetManagers VB procedure

In this step, you will create the GetManagers VB procedure that returns the list of managers from the EMPLOYEE table. The managers list will be returned as a result set from the procedure.

To create the GetManagers procedure:

1. Expand the DB2VBDemo project.
2. Double click the `DB2Class1.vb` entry to open the code in the editor.
3. Modify `Procedure1` as follows:

   ```vba
   Public Shared Sub GetManagers()
       Dim myCommand As DB2Command
       myCommand = DB2Context.GetCommand()
       myCommand.CommandText = "SELECT * FROM EMPLOYEE WHERE JOB='MANAGER'"
       Dim myReader As DB2DataReader
       myReader = myCommand.ExecuteReader()
   End Sub
   ```
4. Save All.
5. Build solution.

**Code analysis:**

Developing Visual Basic procedures is no different than developing C# procedures. The DB2Context object is used to create new DB2 commands that are pre-initialized to point to the context connection for the stored procedure. The DB2 data adapter object is left open to allow for the result set to be returned to the client application.

Cataloging the DB2 VB procedures

Now that we have built the CLR method and assembly, we need to create the script file that will catalog the CLR procedure on the DB2 server.

To catalog the DB2 VB CLR procedure:

1. Right-click the Procedures folder of the DB2CLRDemo project and select `Add => Add New Item`. This opens the Add New Item window.
2. Select `DB2 CLR Procedure Wizard`, and enter `vBSPs.db2sp` for Name.
3. Click **Open**. This launches the DB2 CLR Procedure Wizard.

4. Click **Next**. In the CLR Project Selection step, choose the DB2VBDemo project from the projects drop-down list.

5. Click **Next**. In the CLR Method Selection step, you will see **GetManagers** pre-selected in the Available methods list.
6. Click **Next**. In the Specifications page, select **GetManagers**. Enter your user ID for Schema, 1 for Number of result set(s), **GETMGRS** for Specific name, and **Retrieve list of managers** for Comment.

7. Since this procedure has no parameters, nothing else is required. Click **Finish**.

8. Save All.

You should now see the VBSPs.db2sp project item added under your Procedures project folder.

**Action analysis:**

The DB2 CLR Procedure wizard treats Visual Basic and C# procedures pretty much the same. The wizard showed an initial step where you had to pick up the CLR project. This step is only shown if there are more than one CLR project in your solution.

---

**Setting-up the assembly install**

Now that you have defined the new CLR method, you need to ensure that the assembly containing the CLR procedure is deployed to the local or remote DB2 server.

To setup the assembly install:

1. Right-click the **DB2CLRDemo** project and select **Assemblies**. This opens
the CLR Assemblies window.

2. From the File name drop down list, select **DB2VBDemo.dll** and click the [...] button to locate **bin\DB2VBDemo.dll**.

3. Click **OK** to close the CLR Assemblies window.

4. Build solution.

5. Save All.

You should see, among other things, the following in your build output pane:
Installation of CLR assembly 'C:\test\DB2VBDemo\bin\DB2VBDemo.dll' succeeded.
Generated EXTERNAL name is 'clr\DB2VBDemo.dll\56d9e64b-e38b-44f9-a292-9e8372ab9fe1.dll'.
Performing EXTERNAL name replacement for script file 'C:\test\DB2CLRDemo\Procedures\VBSPs.db2sp'...
EXTERNAL name replacement for script file 'C:\test\DB2CLRDemo\Procedures\VBSPs.db2sp' completed successfully.

Action analysis:
Setting up the assembly install for Visual Basic CLR procedures is no different than what was detailed earlier for C# procedures. You still need to choose the appropriate path for the assembly which might be configuration specific.

Testing the DB2 VB procedures

To test the VB GetManagers procedure:

1. From the IBM Explorer, expand the SAMPLE data connection folder.
2. Right click on the Procedures folder and click Refresh to see the GETMANAGERS procedure.
3. Right-click the GETMANAGERS procedure and click Run Procedure.
4. You should see the GETMANAGERS - DB2 Data Results output data grid. Expand the + sign.
5. Expand the RESULTSET entry to see the list of employees that are managers.
Section 6. Altering the body of CLR procedures

Altering the body of GetManagers procedure

As every programmer knows, there is no such thing as a write once work always project. Requirements are bound to change and bugs are bound to be found. CLR procedures are no different.

In this step, you'll see a new requirement for the GetManagers CLR procedure to return the list of female managers only. This change will only impact the body of the procedure as oppose to its signature, so the catalog information for the procedure is still the same.

To alter the implementation of the GetManagers procedure, do the following:

1. Expand the DB2VBDemo project and double click on the DB2Class1.vb class module.
2. Using the editor, alter the body of the GetManagers procedure as follows:

   ```vbnet
   Dim myCommand As DB2Command
   myCommand = DB2Context.GetCommand()
   myCommand.CommandText = _
   "SELECT * FROM EMPLOYEE WHERE JOB='MANAGER'" & _
   " AND SEX = 'F'"
   Dim myReader As DB2DataReader
   myReader = myCommand.ExecuteReader()
   ```

3. Save All.
4. Build Solution.

When the solution is re-built, you will notice that the assembly is re-deployed to the server and that the CLR procedure is re-cataloged.

Action analysis:

When you modify the body of the CLR procedure in such a way that it requires no alteration to the signature definition of the procedure, then you do not need to perform any additional actions apart from building of the solution. A solution build is required to allow the CLR assembly to be re-built, and to allow the DB2 database project to re-deploy the updated assembly to the server.

Note that the database script files are also re-compiled so that the CLR procedures are updated to point to the new version of the assembly.

Testing the altered DB2 VB procedures

To test the VB GetManagers procedure:
1. From the IBM Explorer, expand the SAMPLE data connection folder.
2. Right click the Procedures folder and select **Refresh**. You should see the GETMANAGERS procedure.
3. Right-click on the GETMANAGERS procedure and select **Run Procedure**.
4. You should see the GETMANAGERS - DB2 Data Results output data grid. Expand the + sign.
5. Expand the **RESULTSET** entry and you should see the list of employees that are female managers (SEX = F).
Section 7. Altering the signature of CLR procedures

Altering the signature of GetManagers procedure

Unlike the previous section where we altered the implementation only of the procedure, this section will introduce a change in requirement that will impact the signature of the stored procedure.

You are now asked to alter the GetManagers CLR procedure so that it would take in as input the sex type for the managers, namely male or female. This will cause the procedure to return the list of male or female managers in the EMPLOYEE table.

To alter the signature and body of the GetManagers procedure, do the following:

1. Expand the DB2VBDemo project and double click on the DB2Class1.vb class module.
2. Using the editor, modify the signature and body of the GetManagers procedure as follows:

```vbnet
Public Shared Sub GetManagers(ByVal inSex As String)
    Dim myCommand As DB2Command
    myCommand = DB2Context.GetCommand()
    myCommand.CommandText = "SELECT * FROM EMPLOYEE WHERE JOB='MANAGER'" & 
        " AND SEX = '" & inSex &uestas & "'
    Dim myReader As DB2DataReader
    myReader = myCommand.ExecuteReader()
End Sub
```
3. Save All.
4. Build Solution.

Although the assembly is re-built and re-deployed to the server, we can not run the GetManagers procedure as the signature has been altered and the DB2 catalogs needs to be updated accordingly.

Recataloging the DB2 VB procedure

In this step, we will update the catalog information for the GetManagers procedure to reflect the method signature change.

To re-sync the list of parameters of the GetManagers VB procedure:

1. Expand the Procedures folder of the DB2CLRDemo project.
2. Double click on the VBSPs.db2sp project item. This will open the script
3. Scroll down and right click **CLR Procedure GETMANAGERS** and select **DB2 CLR Procedure Wizard**.

4. In the DB2 CLR Procedure Wizard, click **Next** to go to the Specifications page.

5. Expand the GetManagers procedure.

6. Select the new inSex method parameter and re-map its type from **VARCAHR(254)** to **CHARACTER(1)**.
7. Click **Finish**.
8. Save All.

You will note that the VBSPs.db2sp script will be re-compiled. The DB2 catalog definition of the GetManagers CLR procedure will now be updated to reflect the addition of the new parameter.

**Action analysis:**

When you modify the body and or signature of the CLR procedure in such a way that it requires an alteration to the signature definition of the procedure, then you must re-invoke the DB2 CLR Procedure wizard to update the mapping information for the altered CLR method.

A solution build is required to allow the CLR assembly to be re-built, and to allow the DB2 database project to re-deploy the updated assembly to the server as well as re-compile all of the project script files so that the CLR procedure definitions are updated to reference the new assembly.

---

**Testing the altered DB2 VB procedure**

To test the GetManagers VB procedures:

1. From the IBM Explorer, expand the SAMPLE data connection folder.
2. Right click on the Procedures folder and click **Refresh**. You should see the GETMANAGERS procedure.
3. Right-click the GETMANAGERS procedure and click Run Procedure.
4. In the Parameters Values window, uncheck the Null option and specify a value of either F or M.
5. Click OK.
6. You should see the GETMANAGERS - DB2 Data Results output data grid. Expand the + sign.
7. Expand the RESULTSET entry and you should see the list of employees that meet your parameter criteria (i.e. male or female).
Section 8. Summary and resources

Summary

Although the usage of CLR methods as database procedures is a new technology, it promises to be a revolutionary step forward bringing the power of the .NET framework to DB2. This tutorial demonstrated the ease of developing and altering C# and Visual Basic procedures using the DB2 Development Add-In Tech-Preview release.

Using the easy to follow instructions for building the sample DB2CLRDemo HR application, you have learned:

° How to create DB2 class assembly projects in C# and Visual Basic.
° How to develop DB2 CLR methods using ADO.NET and the DB2 managed provider.
° How to create a DB2 database project and how to add a DB2 data connection.
° How to install CLR assemblies using your database project.
° How to catalog CLR procedures using your database project.
° How to alter the body and signature of CLR procedures and their DB2 catalog information.

All of the tasks required to build this application were done from within the IDE itself, demonstrating the tight integration of the DB2 application development tooling in this environment.

Resources

The following IBM resources should prove useful for you:

° DB2 Development Add-Ins Technical Preview - An Overview
° DB2 Application Development Technical Preview
° DB2 Developer Domain for additional technical articles and tutorials (http://www.ibm.com/software/data/developer)

The following Microsoft resources should also prove useful for you:

° Microsoft Visual Studio .NET Resources (http://msdn.microsoft.com/vstudio/)
° Microsoft .NET Framework Community (http://www.gotdotnet.com/)
Section 9. Feedback

Your feedback

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)