IBM® DB2 Universal Database™

Development Center Tutorial for Video Online using Microsoft Visual Basic

Version 8

IBM® DB2 Universal Database™

Development Center Tutorial for Video Online using Microsoft Visual Basic

Version 8

Before using this information and the product it supports, be sure to read the general information under *Notices*.

Contents

About the tutorial

Welcome to the IBM® DB2 Universal Database™ Development Center tutorial for Video Online using Microsoft Visual Basic.

The Development Center tutorial guides you through basic Development Center tasks using the IBM DB2 Development Add-In within Microsoft Visual Basic. The tasks take you through steps to build a real application for a realistic scenario, a video-rental store. The IBM DB2 Development Add-In is used to exhibit the power and key functionality of the DB2 Development Center. The add-in also builds the IBM Video Online for e-business sample solution.

This tutorial describes the sample solution architecture. In addition, this tutorial explains how to use the DB2 Development Center Add-In for Microsoft Visual Basic to develop the data layer. More information about the entire sample solution is available at http://www.ibm.com/software/data/developer/samples/video.

This tutorial should take approximately two hours for you to complete.

Before you begin

Before you start the tutorial, you need some prerequisite knowledge of the concepts and tools that this tutorial uses.

Prerequisite knowledge:
- Fundamental knowledge of Active Data Objects (ADO)
- Fundamental knowledge of an extension of Component Object Model (COM+)
- Relational database concepts
- Fundamental knowledge of SQL
- Fundamental knowledge of Microsoft Visual Basic

Also, ensure that your computer has the minimum software requirements:
- IBM DB2 Universal Database for Windows, Version 8.1 or later
- IBM DB2 Universal Database Enterprise Server Edition
- IBM DB2 Application Development Client, Version 8.1 or later
- Microsoft Visual Basic, Version 6.0 or later
- Microsoft Data Access Components (MDAC), Version 2.6 or later

Save and unzip `videoonline.zip` to your C drive.

You can find `videoonline.zip` in the following locations:

- If the HTML documentation was installed with the DB2:
  `<db2 inst directory>\doc\htmlcd\<locale>\tutr\db2td\videoonline.zip`
- If the HTML documentation was installed with the HTML CD:
  `<cd drive>:\Program Files\sqllib\doc\htmlcd\<locale>\tutr\db2td`
  `\videoonline.zip`

where:

- `<locale>` is the locale (such as en_US)
- `<cd drive>` is the CD drive (such as C drive)

Make sure to expand all of the subfolders when you unzip the files. All the files that you need to complete this tutorial are included in `videoonline.zip`.

Product overview

IBM Video Online for e-business is an online video rental cyber store-front that can interact with IBM Video Central for e-business®. Video Central is the business-to-business data repository and generic Web service provider for registered, Web-based video rental applications. Video Online can either act as a stand-alone application, or it can work with Video Central to meet the increasing business needs of a typical video rental store.

Conventions used in this tutorial

This tutorial uses typographical conventions in the text to help you distinguish between the names of controls and text that you type. For example:

- Menu items are in boldface font:

  Click **Menu —> Menu choice.**
- Push buttons and selection choices on the application interface are in boldface font:

  Click **Button Name**.
- Text that you type is in example font:

  `This is the text that you type.`
- File or directory names are also in example font:

  `...\SQLLIB\spb\projects\file.mdb`

Introduction to the IBM Video Online for e-business application

In this lesson, you will learn about the Video Online sample solution business scenario and architecture design.

Business scenario

IBM Video Online for e-business is an online video rental cyber store-front (business to consumer e-commerce application). This type of Web application might be an independent business, or it might be an extension of an existing physical video rental store that wants to increase its revenue and customer base by establishing a Web presence. This solution takes advantage of a combination of IBM and Microsoft technologies including IBM DB2, the IBM DB2 Development Center, the IBM DB2 Development Add-In for Microsoft Visual Basic, eXtensible Markup Language (XML), eXtensible Stylesheet Language (XSL), Component Object Model (COM+), Active Data Objects (ADO), and Active Server Pages (ASP).

This e-business solution empowers customers to easily and quickly do any of the following tasks 24 hours a day from home or office:

- Register as a new member
- View and update their registration profile
- View rental history and current outstanding titles
- View and update the reserved titles list for automatic delivery when the title is available
- View and update the wished titles list for future notification when the title is available
- View a list of upcoming releases, new arrivals, and popular titles
- View a list of recommended titles based on business intelligence queries
- View detailed information about each rental title
- Search for rental titles
- Evaluate rental titles using reviews and five-star ratings

Customers no longer need to drive to the video store, search the aisles for the movie they want, wait in line to rent the video, or drive back to the video store to return the movie. When they are logged in, they can use a powerful search engine to find videos quickly based on a variety of search criteria, rent a video with a few clicks, and have the video delivered to their mailbox in days. Returning a video is also convenient. Customers simply put the video in the pre-addressed, stamped package that comes with the video and drop it in the mailbox. The site offers additional features including detailed biographies about actors, directors, and screen writers; critic and other member reviews; and a personal rental history.

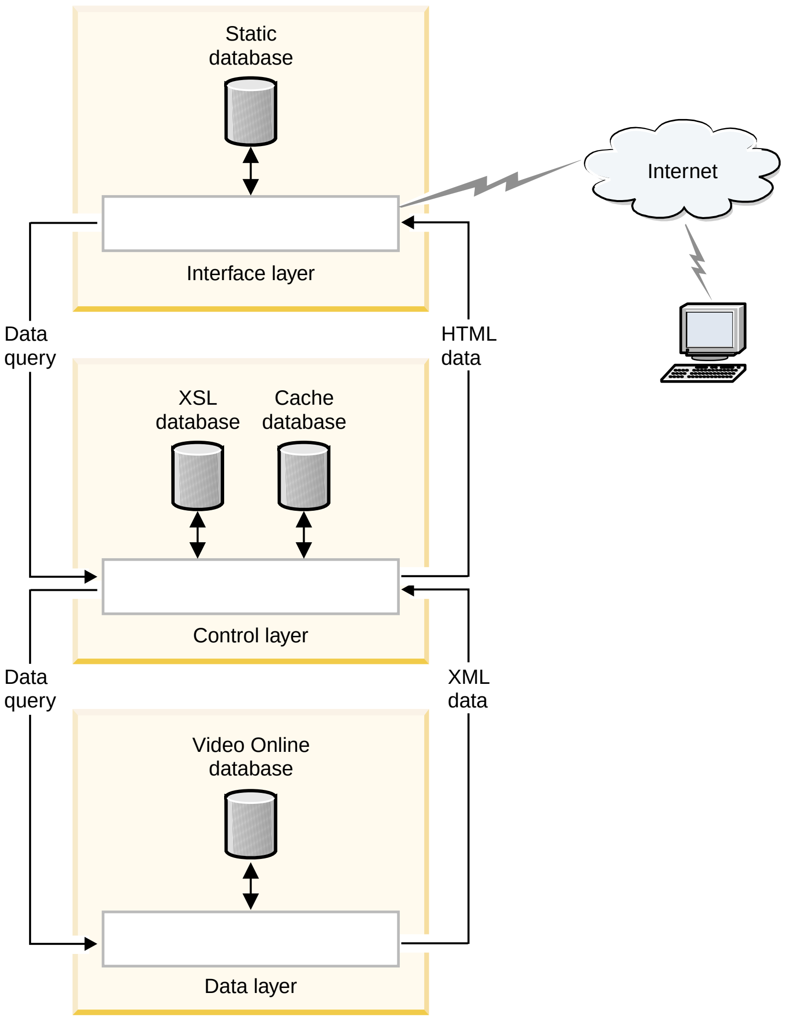
Multi-tier architecture

This IBM Video Online for e-business solution is built in a multi-tier fashion. There are three logical layers, each of which can be physically hosted on one application server or an application server farm. These layers are the data layer, the control layer, and the interface layer. The logical separation of these layers simplifies application development. Instead of creating one large application, you create modular reusable software components that can be easily maintained and extended to accommodate new requirements. This tutorial focuses on building only the data layer, but it is important to learn about the entire solution so you can put the tutorial tasks in context with the larger application.

At the base of the application is the data layer, also called the data-access layer, where the database resides. The data layer runs the data-access queries and updates, and it enforces data referential integrity and data consistency. The query results from this layer are forwarded to the control layer as XML result sets, which can be HTML fragments.

The control layer is the middle layer and is sometimes referred to in other applications as the business-logic layer. This layer enforces business rules and submits data requests from the interface layer to the data layer. When necessary, the control layer uses XSL to transform the data layer XML result sets into HTML fragments. These HTML fragments are combined to produce larger HTML segments that the interface layer can use.

The top interface layer is also referred to as the presentation layer in other applications. This layer provides the Web user interface in HTML by executing Active Server Pages (ASPs) that contain Visual Basic scripts. The interface layer does not read the data embedded in the HTML.



The visual presentation of the data is managed across two layers:

- The interface layer manages the general page layout, static HTML, and static images.
- The control layer performs the XML to HTML transformation, and the optional HTML caching.

The business rules of the solution are also managed across two layers:

- The control layer enforces business security rules including data access and functionality.
- The data layer uses database referential integrity, constraint checks, and triggers to ensure that business relational data rules are enforced.

Data layer

The data layer has several COM+ objects with one or more methods. Each method typically establishes a database connection, calls the necessary SQL or stored procedures, disconnects from the database connection, then returns the result set, if applicable, to the control layer in the form of an XML document. You will create a few of the COM+ objects in Lesson 5.

ADO facilitates communication between the database server and the data-layer objects and methods. For outbound XML data generated as a result of database query, the data layer invokes the XML stored procedure using an ADO command object. The XML stored procedure invokes the actual SQL stored procedure, converts the SQL result sets to an XML document, and returns the XML document as an output parameter. The XML document is then returned to the data layer as an ADO command output parameter. For inbound XML data required for database update, the data layer invokes the SQL stored procedure using an ADO command object passing in the XML document as an input parameter. The SQL stored procedure uses the XML table user-defined function (UDF) to convert the XML data into tabular SQL data. This tabular data is then used in the database update process. In Lesson 1, you will prepare the database, and in Lessons 2, 3, and 4, you will create UDFs as well as create and debug stored procedures.

Additionally, by using COM+ and ADO, you can use database connection pooling. The cost of individual connect and disconnect calls is minimized because the object needs to only hold or release a database connection from the pool of previously established database connections.

Control layer

The control layer, like the data layer, is made up of a number of COM+ objects. Each object has one or more methods that generally fall into the following two major business logic categories:

- Data query methods
- Data manipulation methods

Data query methods

These methods are responsible for returning one or more HTML data blocks to the interface layer. Data query methods first retrieve the appropriate data block from the optional HTML cache. If an entry is found, then the HTML data block is returned to the requester. If an entry is not found, then the method will perform the following actions:

1. Start the appropriate data layer method to retrieve the XML data document using the required input parameters.
2. Retrieve the required XSL document from the XSL cache.

3. Apply XSL transforms to the XML data document to convert the XML data to an HTML data block (fragment).
4. Asynchronously insert the HTML data block into the HTML optional cache.
5. Return the HTML data block to the requester.

The process remains the same regardless of the type of data block that is requested. The only difference is that some blocks are not cached because their content is too dynamic for caching to be feasible. Also, one method can handle multiple data blocks to reduce the number of round-trip calls between the various tiers. The methods responsible for returning the left, main, and right Web page elements to the interface layer are examples of methods that handle multiple data blocks.

Data manipulation methods

Data manipulation methods are responsible for updating the user account, and can add or remove wish-list or reserved-list entries, rate and review titles, or update account information. A data manipulation method performs the following actions:

1. Starts the appropriate data layer method to update the database.
2. If the update fails, returns a failure code.
3. If the update succeeds, asynchronously deletes all applicable cached HTML data block entries that are invalidated due to the update, and returns a success code.

There is an added level of complexity when a method handles more than one data element. For example, when multiple titles are added simultaneously to the wish list, the AddWishList method must handle all the titles. To accommodate the additional information, the method parameters can be represented in an XML document that describes the data elements.

Interface layer

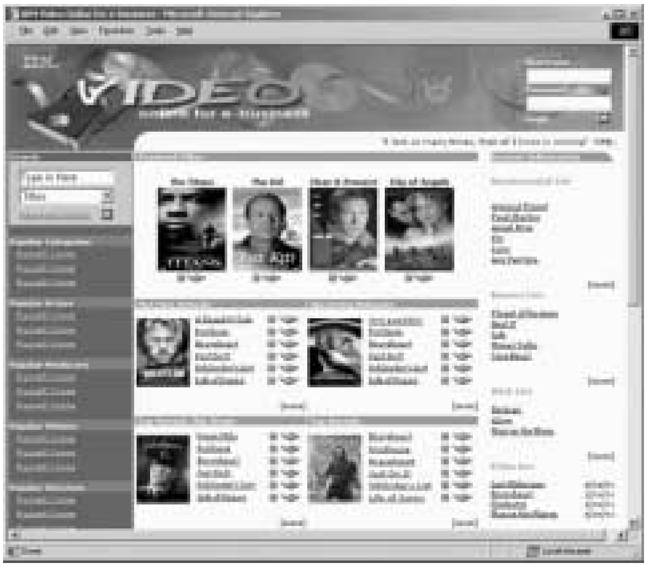
The interface layer uses several Active Server Pages to assemble the various pieces of the Video Online Web pages. The components of this layer are physically hosted by the Web application server, and typically reside behind an Internet firewall.

The interface layer presents the application interface, manages the overall Web page layout, and provides navigation between the Web pages.

Web page layout

The Web pages are a combination of static and dynamic HTML data blocks. The static blocks include content such as basic page layout, images, and fixed text and controls. ASP server-side include statements are used to implement these static blocks. Dynamic HTML blocks are blocks of HTML that contain embedded data that is dynamically created on demand. These dynamic blocks are retrieved from the control layer using the COM+ method calls. The basic Web page layout includes:

- The top area that contains the site logo and banner, member welcome section, and site top navigational controls
- The left area that contains search controls and any navigational links that are related to the content shown in the center area
- The center area that contains the information that is requested by the member, such as title lists and expanded title or cast and crew information
- The right area that contains quick member account information including the customer's recommended title list, reserved titles queue, wish list titles, outstanding titles, and an activity log that tracks the member's actions while the customer is visiting the site
- The bottom area that contains application information and legal text



Session information and security

Video Online Web site customers can view much of the site content regardless of whether they are logged in. Customers who sign up for membership are assigned a unique user ID and password combination. ASP session management in the interface layer stores the member login identifier, LoginID, in the session object. Customers must enable cookies on their Web browsers.

After the LoginID is saved in the session, a customer can retrieve and update specific account data. Additional security mechanisms such as encryption and origin-point validation can be used to secure regular account updates and sensitive account updates, such as changing credit card or address information.

You can use a secure sockets layer (SSL), to encrypt sensitive data. With origin-point validation, you can ensure that the client computer is the same as the computer that issued the login request. Both methods can be used to prevent client cookie sniffing, HTML data sniffing, and identity spoofing.

Encryption and validation create additional response-time overhead.

Static Web content

Frequently accessed static data is replicated from the main Video Online database and stored on the Web server as plain text or image files for fast retrieval. Video title images are examples of static data that is kept locally on the Web server.

Summary

The Video Online architecture is based on a three-tier logical layering model. Balancing work between the three layers, the application takes advantage of the latest technology to meet business needs. The data layer is the only layer with direct access and knowledge of the Video Online database. Information requests and updates are first passed through the interface layer to the control layer, where the appropriate data-layer method-call requests are determined and sent to the data layer. Next, the data layer performs the requested query or update, and returns information to the control layer in the form of XML. Then, the control layer transforms the XML into HTML as required, combines multiple HTML fragments into larger sections, and returns the HTML sections to the interface layer.

Lesson 1: Enable the database to retrieve XML record sets

The objective of this lesson is to configure the sample Video Online database and prepare the database environment for application development in future lessons. Batch commands were created for this tutorial to simplify some of these tasks. In this lesson, you will enable the database for XML.

In order to retrieve XML record sets, you need to create a user-defined function (UDF). The required UDF in the database, which will enable you to retrieve XML-formatted record sets, will be created using the `vosetupxml.bat` command.

To enable the database:

1. Open the DB2 command window.
2. From the `videoonline\db` directory, enter `vosetupxml.bat`.

Checkpoint

You created the UDF in the database. You can use the DB2 command window to verify that the UDF was created and works properly.

To verify that the UDFs were created properly:

1. Open the DB2 command window.
2. From the `videoonline\db` directory, enter `vocheckxml.bat`.

After you run vocheckxml.bat, you should see the following result in your DB2 command window: `The SQL Command completed successfully`.

If the UDFs were not created properly, run `vosetupxml.bat` again and look for errors.

Lesson 2: Creating the Visual Basic project and launching the IBM DB2 Development Add-In

The objective of this lesson is to create a Visual Basic project and then launch the IBM DB2 Development Add-In. You will use the add-in to create UDFs and stored procedures within the DB2 database. Next, the add-in generates ADO code for Connection and Command objects, so that the Visual Basic application can interact with the database. This interaction takes place through the use of COM+ objects.

Step 1: Registering the IBM DB2 Development Add-In

The process for installing or registering the IBM DB2 Development Add–In depends on whether you installed Microsoft Visual Basic before you installed DB2. After the add-in is successfully registered, you can begin using it in your Visual Basic development environments.

To register the IBM DB2 Development Add–In:

- If Visual Basic was installed before DB2, the DB2 installation automatically registers the IBM DB2 Development Add–In.
- If Visual Basic was installed after DB2, enter the following command on a command line:

  ```
  db2vscmd register
  ```

Step 2: Setting up the Visual Basic project

Before you can create UDFs, you need to create a new ActiveX project and activate the IBM DB2 Development Add–In. After the add-in is activated, many features of the Development Center are available in the Microsoft Visual Basic environment.

To set up a new Visual Basic project:

1. Create a new Visual Basic project:
   a. Start Microsoft Visual Basic. The New Project window opens.
   b. Click **ActiveX DLL**.
   c. Click **Open**. In the Project Explorer window, you can see Project1 and the Class1 class module.
   d. In the Project Explorer window, click **Project1 (Project1)**.

   e. In the Properties window, rename Project1 to `IVOEBDL` and press Enter. You will see the name change in the Properties window, Project Explorer window, and Code window.
2. Add the Wishlist.cls class file to the project:
   a. In the Project Explorer window, right-click **IVOEBDL**, and click **Add —> Add File**. The Add File window opens.
   b. Select the `videoonline\VB\Wishlist.cls` file.
   c. Click **Open**.
3. Remove Class1:
   a. In the Project Explorer window, right-click **Class1 (Class1)**, and click **Project —> Remove Class1**. The Microsoft Visual Basic window opens.
   b. Click **No**.
4. Save your new project:
   a. Click **File —> Save Project As**.
   b. Go to the `videoonline` directory.
   c. Click **Save**. If the Source Safe Control alert window opens, click **No**.

Step 3: Preparing the OLE DB provider

Next, you need to configure the OLE DB provider in Microsoft Visual Basic. The OLE DB provider is the link between the database and the application.

Microsoft OLE DB is a set of OLE/COM interfaces that provide applications with uniform access to data stored in diverse information sources. There are OLE DB consumers and OLE DB providers. An OLE DB consumer is any system or application that uses OLE DB interfaces. An OLE DB provider is a component that exposes OLE DB interfaces. Using ADO to access data from an OLE DB provider greatly simplifies application development because ADO hides the inherit complexities of the OLE DB provider interfaces.

With the IBM OLE DB Provider, DB2 can act as an OLE DB provider. This support gives OLE DB-based applications the ability to extract or query DB2 data using the native OLE interface. Additionally, OLE DB consumers can access data on a DB2 Universal Database server.

To configure the OLE DB provider in Microsoft Visual Basic:

1. Click **Project —> References**. The References – IVOEBDL.vbp window opens.
2. Select **Microsoft ActiveX Data Objects 2.6 Library**. If you have a newer version (such as Microsoft ActiveX Data Objects 2.7 Library) installed, select it. If you need to download MDAC 2.6 or later, you can install it from http://microsoft.com/data/default.htm.

3. Click **OK**.

See Additional Information for more information about the IBM OLE DB provider.

Step 4: Starting the IBM DB2 Development Add-In

To start the IBM DB2 Development Add–In:

Click **Add-Ins —> IBM DB2 Development Add–In.**

It might take a few seconds for the DB2 Development View to open. When the DB2 Development View opens, you will see the IVOEBDL project.

Now you are ready to start creating stored procedures and UDFs that are needed for your Video Online application using the IBM DB2 Development Add–In.

Note: It is advisable that you do not maximize the Visual Basic window because some of the dialogs and wizards that are launched by the DB2 Development Add-In may open behind the Visual Basic window.

Step 5: Adding and testing a database connection

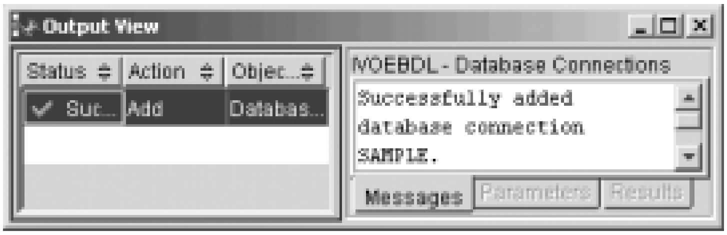
Use the Add Database Connection wizard to add a database connection to your project.

To add a connection using the Add Database Connection wizard:

1. In the DB2 Development View, right-click **IVOEBDL**, and click **Add Connection**. The Add Database Connection wizard opens.
2. On the Connection page, specify the database alias that you want to use for your connection and your user ID and password:
   a. In the **Alias** field, specify **SAMPLE**.
   b. Select the **Use your current user ID and password** check box.
   c. Click **Next**.
3. On the Filter page, specify whether you want to filter the objects in your connection:
   a. Select the **Filter using specified criteria** check box.
   b. In the **Schema** field, select **Equal to the names**, and type `VO` in the text field.
   c. Click **Next**.
4. On the Options page, specify the SQL schema for the database connection:
   a. In the **SQL schema or SQL ID** field, type `VO`.

b. Click **Finish**. The Sample database is added to the DB2 Development View.

Messages about the database connection are displayed on the Messages page of the Output View.



```
Successfully added
database connection
SAMPLE.
```

Now you are ready to start creating UDFs and stored procedures that are needed for your Video Online application.

Step 6: Creating and populating the database

This tutorial provides scripts to perform all the tasks involved in creating and populating the Video Online sample database. These scripts are included in videoonline.zip.

The vosetupdb.bat script connects to the database and creates tables. Then the script loads the sample data into each table. The Video Online database is created and populated by the script.

To create the complete database:

1. Open the DB2 command window.
2. From the `videoonline\db` directory, enter `vosetupdb.bat`.

It can take a few seconds for the scripts to complete all of the steps. You can refer to the files in `videoonline\db\data\msg` for messages indicating whether the contents of the tutorial were successfully created and populated.

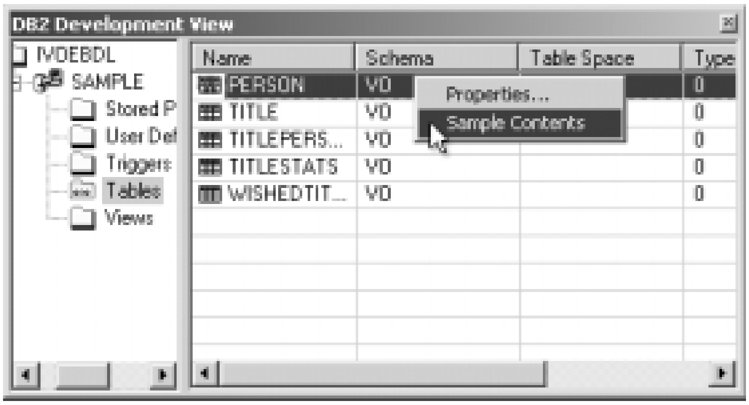
Checkpoint

You created the `VO` schema and associated tables and indexes in the Sample database. You can use the DB2 Development Add-In to verify that the schema and tables were properly created and that the data was loaded.

To verify that the tables were created and populated:

1. In the DB2 Development View, expand the **Sample** database.

2. Right-click **Tables**, and click **Refresh**. You will see five tables listed in the DB2 Development View.
3. Right-click on any of the tables, and click **Sample Contents**.



Name	Schema	Table Space	Type
PERSON	VO		0
TITLE	VO		0
TITLEPERS...	VO		0
TITLESTATS	VO		0
WISHEDTIT...	VO		0

If the tables were not created and populated correctly, rerun `videoonline\db\vosetupdb.bat` and look for errors.

In the DB2 Development View, right-click **IVOEBDL**, and click **Save**.

Lesson 3: Creating the UDFs

The objective of this lesson is to create, build, and run two UDFs. These functions perform necessary functionality for the titles feature of the video online sample solution and can be built upon for additional features. You will build each of the functions using the IBM DB2 Development Add-In within Microsoft Visual Basic. The add-in uses the Development Center wizards and windows to create UDFs. In each case, the SQL statement is provided so that you can focus on the tasks instead of the application logic.

Step 1: Creating the TitleAvailability UDF

Now that your project is set up and the connection is added, you are ready to create the necessary UDFs. UDFs are extensions or additions to the existing built-in functions of the SQL language. A UDF can be a scaler, which returns a single value each time is is called, a column function, which is a passed set of like values and returns a single value for the set, a row function, which returns the row, or a table function, which returns a table.

A UDF can be a column function only when it is sourced on an existing column function.

In this step you will create a scalar UDF called TitleAvailability. This UDF determines if a movie title is in stock or out of stock and retrieves the appropriate icon when given a title identification number. Stored procedures that return a customer's wish list will use this UDF to return the availability of the titles.

To open the Create SQL User-Defined Function wizard:

1. In the DB2 Development View, right-click **User-Defined Functions**, and select **New**. The New Object window opens.
2. Click **User-Defined Function** in the left pane.
3. Click **SQL** in the right pane.
4. Click **OK**. The Create SQL User-Defined Function wizard opens.

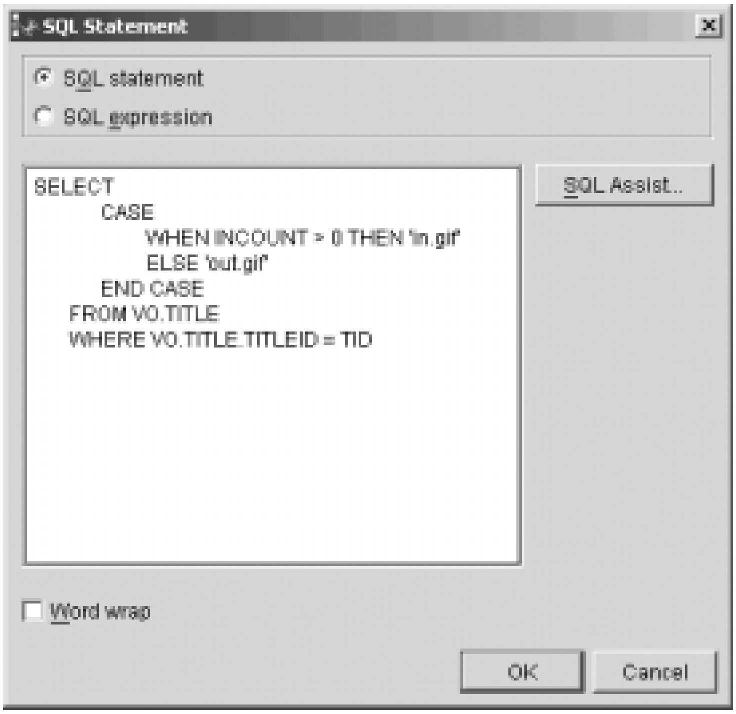
To create the UDF using the wizard:

1. On the Name page, specify a name for the UDF:
   a. In the **Name** field, type `VO.TitleAvailability`.
   b. Click **Next**.
2. On the Definition page, specify the settings to define the UDF:

a. In the **Statement** option row, click ... . The SQL Statement window opens.

b. To create the SQL statement and set the parameters for the UDF, replace the default SQL statement by typing the following SQL statement into the text box:

```
SELECT
      CASE
          WHEN INCOUNT > 0 THEN 'in.gif'
          ELSE 'out.gif'
      END CASE
    FROM VO.TITLE
    WHERE VO.TITLE.TITLEID = TID
```



[✓ SQL statement] SQL expression

```
SELECT
      CASE
          WHEN INCOUNT > 0 THEN 'in.gif'
          ELSE 'out.gif'
      END CASE
    FROM VO.TITLE
    WHERE VO.TITLE.TITLEID = TID
```

This SQL statement uses TID as an input parameter, and it checks to see if there are any copies of the title in stock. If there is at least one copy of the movie, the in-stock image is returned. If there are no copies available, the out-of-stock image is returned.

c. Click **OK** to close the SQL Statement window. You should see the revised SQL statement in the Definition page.

d. In the **Output Type** field, specify **Scalar** as the value.

e. Click **Next**.

3. On the Return Data Type page, specify the data type that you want the UDF to return:
   a. In the **SQL type** field, specify **VARCHAR**.
   b. In the **Length** field, type `10`.
   c. Click **Next**.
4. On the Parameters page, specify parameters for the UDF:
   a. Click **Add**. The Add Parameter window opens.
   b. In the **Name** field, type `TID`.
   c. In the **SQL Type** field, specify **Integer**.
   d. Click **OK**. You will see the TID parameter in the Parameters page.
   e. Click **Finish**.

You can see your new UDF in the **User-Defined Functions** folder in the DB2 Development View. Messages about the build are displayed on the Messages page of the Output View. You will run and test this UDF in the checkpoint procedure at the end of the lesson.

Step 2: Creating the XWishedTitles UDF

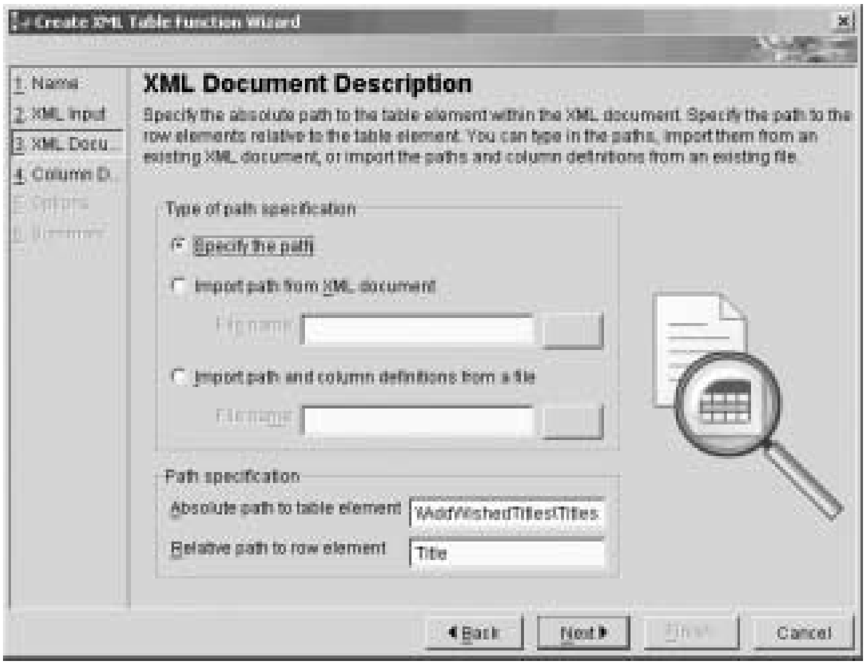
In this step, you will create a table UDF called XWishedTitles. Based on an XML VARCHAR, this UDF calls other DB2 functions to retrieve a list of IDs.

To open the Create XML Table Function wizard:

1. In the DB2 Development View, right-click **User-Defined Functions**, and click **New**. The New Object window opens.
2. Click **User-Defined Function** in the left pane.
3. Click **XML** in the right pane.
4. Click **OK**. The Create XML Table Function wizard opens.

To create the UDF using the wizard:

1. On the Name page, specify the name of the table UDF.
   a. In the **Name** field, type `VO.XWishedTitles` where `VO` is the schema and `XWishedTitles` is the name of the XML UDF that you are creating.
   b. Click **Next**.
   c. Click **Next**.
2. On the XML Document Description page, specify the absolute path to the table element within the XML document and the path to the row elements relative to the table element:
   a. In the **Absolute path to table element** field, type `\\AddWishedTitles\Titles`.
   b. In the **Relative path to row element**, type `Title`.



XML Document Description

Specify the absolute path to the table element within the XML document. Specify the path to the row elements relative to the table element. You can type in the paths, import them from an existing XML document, or import the paths and column definitions from an existing file.

Type of path specification
- (●) Specify the path
- ( ) Import path from XML document
  - File name:
- ( ) Import path and column definitions from a file
  - File name:

Path specification
- Absolute path to table element: \AddWishedTitles\Titles
- Relative path to row element: Title

c. Click **Next**.

3. On the **Column Definition** page, click **Add**. The Add Column Definition window opens.
4. In the **Relative path** field, type `TitleID`.
5. In the **SQL type** field, specify **INTEGER**.
6. Click **OK**.
7. Click **Finish**.

You can see your new UDF in the **User-Defined Functions** folder in the DB2 Development View. Messages about the build are displayed on the Messages page of the Output View. You will run and test this UDF in the checkpoint procedure at the end of the lesson.

Step 3: Importing the TitleRating UDF

In this step, you will import a scaler UDF called TitleRating. This UDF is used to convert a TitleRating float value to a five-star rating icon name.

To open the Import wizard:

1. In the DB2 Development View, right–click **User–Defined Functions**, and click **Import**. The Import User–Defined Function window opens.
2. Click **File System** in the left pane.

3. Click **Source File** in the right pane.
4. Click **OK**. The Import wizard opens.

To import the UDF using the wizard:

1. On the Source File page, click ... in the **Statement** option row so that you can select the file that you want to import. The Choose window opens.
2. Select the `videoonline\db\ddl\TitleRating.db2` file.
3. Click **Choose**.
4. In the Import wizard, click **Next**.
5. Click **Next**.
6. Click **Finish**.

You can see your new UDF in the **User-Defined Functions** folder in the DB2 Development View. Messages about the build are displayed on the Messages page of the Output View. You will run and test this UDF in the checkpoint procedure at the end of the lesson.

Checkpoint

In this lesson, you created and built three UDFs. You can check that the UDFs are correct by running each one.

To run each UDF:

1. To run the TitleAvailability UDF:
   a. In the DB2 Development View, right-click the name of the UDF, and click **Run**. The Specify Parameter Values window opens.
   b. In the **Value** field, type 1.
   c. Click **OK**.

   After you run the UDF, look at the Output View for results. Any messages indicating success or failure are displayed on the Results page. You should see `in.gif` as the result.
2. To run XWishedTitles:
   a. In the DB2 Development View, right-click the name of the UDF, and click **Run**. The Specify Parameter Values window opens.
   b. In the **Value** field, type:
   ```
   <AddWishedTitles><Titles><Title><TitleID>101</TitleID>
   </Title></Titles></AddWishedTitles>
   ```
   c. Click **OK**.

After you run the UDF, look at the Output View for results. Any messages indicating success or failure are displayed on the Messages page. You should see `101` as the result.

3. To run TitleRating:
   a. In the DB2 Development View, right-click the name of the UDF, and click **Run**. The Specify Parameter Values window opens.
   b. In the **Value** field, type `1`.
   c. Click **OK**.

   After you run the UDF, look at the Output View for results. Any messages indicating success or failure are displayed on the Messages page. You should see `10.gif` as the result.

In the DB2 Development View, right-click **IVOEBDL**, and click **Save**.

Lesson 4: Creating and importing the wish list stored procedures

The objective of this lesson is to create one stored procedure and import three stored procedures used to implement the wish-list functionality. You will create and import stored procedures that query and update records in the database.

Step 1: Creating the XWishedTitlesAdd stored procedure

The XWishedTitlesAdd stored procedure uses an SQL insert statement to add one or more titles to a customer's wish list. You will create the XWishedTitlesAdd stored procedure using a wizard. The XWishedTitles table UDF is used to extract the title IDs from the input XML document.

To open the Create SQL Stored Procedure wizard:

1. In the DB2 Development View, right-click **Stored Procedures**, and click **New**. The New Object window opens.
2. Click **Stored Procedure** in the left pane.
3. Click **SQL** in the right pane.
4. Click **OK**. The Create SQL Stored Procedure wizard opens.

To create the stored procedure using the wizard:

1. On the Name page, specify a name for the stored procedure:
   a. In the **Name** field, type `VO.XWishedTitlesAdd` where `VO` is the schema and `XWishedTitlesAdd` is the name of the stored procedure that you are creating.
   b. Click **Next**.
2. On the Definition page, specify the settings to define the stored procedure:
   a. In the **Result set** field, specify **None**.
   b. Click **Next**.
3. On the Parameters page, specify parameters for the stored procedure.
   a. Click **Add**. The Add Parameter window opens.
   b. Click **In** to set the Parameter mode as input.
   c. In the **Name** field type `MID`.
   d. In the **SQL type** field, specify **Integer**.
   e. Click **Apply**. The `MID` parameter is shown in the Parameters page.
4. Add the TITLES input parameter.

   a. In the **Name** field, type TITLES.
   b. In the **SQL type** field, specify **VARCHAR**.
   c. In the **Length** field, type 4000.
   d. Click **OK**. The TITLES parameter is shown in the Parameters page.
   e. Click **Next**.
5. On the Options page, specify options for creating and building the stored procedure:
   a. Clear the **Build** check box because you will be modifying the stored procedure using the editor.
   b. Click **Finish**.

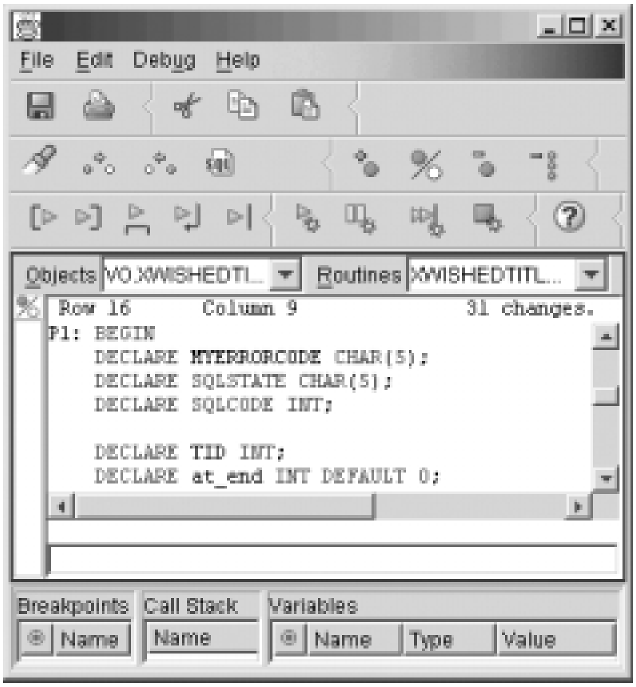
You can see your new stored procedure in the **Stored Procedure** folder in the DB2 Development View.

To modify the stored procedure using the editor:

1. In the DB2 Development View, right-click **XWishedTitlesAdd**, and click **Edit Source**. The editor opens.
2. In the editor window, replace the generated code of the stored procedure body by copy and pasting or typing the following SQL statement:

   Replace only the code from P1:Begin to End P1.

```
P1: BEGIN
    DECLARE MYERRORCODE CHAR(5);
    DECLARE SQLSTATE CHAR(5);
    DECLARE SQLCODE INT;

    DECLARE TID INT;
    DECLARE at_end INT DEFAULT 0;

    DECLARE cursor1 CURSOR FOR
      SELECT TitleID
      FROM   TABLE (VO.XWishedTitles(Titles)) AS T;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET MYERRORCODE = SQLSTATE;
    DECLARE CONTINUE HANDLER FOR not found SET at_end = 1;

    OPEN cursor1;

    FETCH cursor1 INTO TID;
    WHILE (at_end = 0) DO
      INSERT
        INTO   VO.WISHEDTITLES ( CUSTOMERID, TITLEID, DATEADDED )
        VALUES ( MID, TID, CURRENT DATE );

      FETCH cursor1 INTO TID;
    END WHILE;
    CLOSE cursor1;
END P1
```



```
Row 16      Column 9           31 changes.
P1: BEGIN
    DECLARE MYERRORCODE CHAR(5);
    DECLARE SQLSTATE CHAR(5);
    DECLARE SQLCODE INT;

    DECLARE TID INT;
    DECLARE at_end INT DEFAULT 0;
```

3. Click **File —> Save Object**.
4. Close the editor.
5. In the DB2 Development View, right-click **XWishedTitlesAdd**, and click **Build**.

You can see the new stored procedure in the **Stored Procedures** folder in the DB2 Development View. Messages about the build are displayed on the Messages page of the Output View. You will run and test the stored procedures at the end of the lesson.

Step 2: Importing three wish-list stored procedures

Wish-list stored procedures use an SQL query to return detailed information about all of the titles in a member's wish list. The results of this query are displayed in the main panel of the Web interface. You will import three wish-list stored procedures (GetWishedTitles, GetNonWishedTitles, and XGetWishedTitles) using a wizard.

To open the Import wizard:

1. In the DB2 Development View, right-click **Stored Procedure**, and click **Import**. The Import Stored Procedure window opens.
2. Click **File System** in the left pane.

3. Click **Source File** in the right pane.
4. Click **OK**. The Import wizard opens.

To import the stored procedure using the wizard:

1. On the Source File page, click in the **Statement** option row so that you can select the file that you want to import. The Choose window opens.
2. Select the `videoonline\db\ddl\GetWishedTitles.db2` file.
3. Click **Choose**.
4. In the Import wizard, click **Next**.
5. Click **Next**.
6. Click **Finish**.

Repeat the process to import two more stored procedures:
- `GetNonWishedTitles.db2`
- `XGetWishedTitles.db2`

You can see the new stored procedure in the **Stored Procedures** folder in the DB2 Development View. Messages about the build are displayed on the Messages page of the Output View. You will run and test the stored procedures at the end of the lesson.

Checkpoint

To verify that your stored procedures work correctly, you can run each stored procedure and check its messages and results.

Follow these steps for the XWishedTitlesAdd stored procedure:
1. In the DB2 Development View, right-click the stored procedure, and click **Run**. The Specify Parameter Values window opens.
2. In the **Value** field for `MID`, type `2`.
3. In the **Value** field for `Titles`, type
```
<AddWishedTitles><Titles><Title><TitleID>5</TitleID>
</Title></Titles></AddWishedTitles>
```
4. Click **OK**.

Follow these steps for the GetWishedTitles, GetNonWishedTitle, and XGetWishedTitles stored procedures:
1. In the DB2 Development View, right-click the stored procedure, and click **Run**.
2. In the **Value** field for `MID`, type `1`.

3. Click **OK**.

The Output View opens, and you will see the Messages page indicating that the stored procedures ran successfully. Click the **Results** page of the Output View to see the stored procedures results.

In the DB2 Development View, right-click **IVOEBDL**, and click **Save**.

Lesson 5: Adding the database utility module

The objective of this lesson is to create a new database utility (DBUtil) module in your Visual Basic project. The IBM DB2 Development Add-In automates much of this process for you. This module supplies database connections to other class modules in the application.

Step 1: Adding a DBUtil module

First, you must add an empty module to your Visual Basic project. To add a new module:

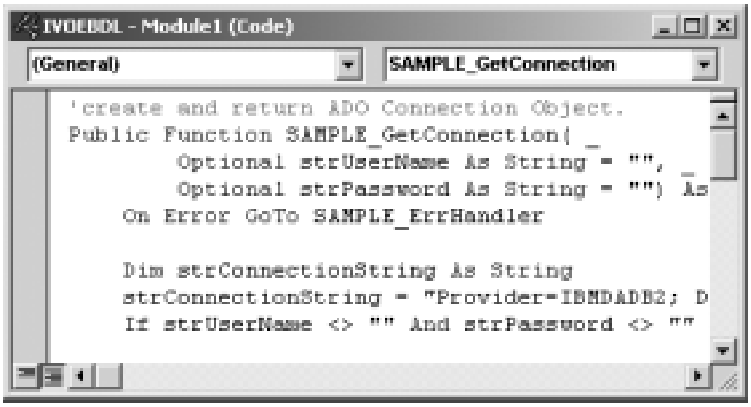
1. In the Project Explorer window, right-click **IVOEBDL**, and click **Add —> Module**. The Add Module notebook opens.
2. Click **Open.** The IVOEBDL – Module1 (Code) window opens. A Modules folder is added to the Project Explorer window.
3. In the Properties window, rename Module1 to `DBUtil` and press Enter. You will see the name of the module change in the Properties window, the Project Explorer window, and in the Code window.

Step 2: Adding the ADO connection code

Now that you have a module, you can add the code that will create an ADO connection object so you can connect to your sample database. This code must be added to the module and each stored procedure.

To use the IBM DB2 Development Add-In to create the ADO connection code:

1. In the Project Explorer window, double-click **DBUtil**.
2. In the DB2 Development View, right-click **Sample**, and click **Add ADO Connection Code**. The `Sample_Get Connection` function is inserted into the module.



```
'create and return ADO Connection Object.
Public Function SAMPLE_GetConnection( _
        Optional strUserName As String = "", _
        Optional strPassword As String = "") As
    On Error GoTo SAMPLE_ErrHandler

    Dim strConnectionString As String
    strConnectionString = "Provider=IBMDADB2; D
    If strUserName <> "" And strPassword <> ""
```

3. In the Project Explorer window, right-click **DBUtil**, and click **Save DBUtil**. The Save File As window opens.
4. Click **Save**.

Step 3: Adding the ADO command code

Now that you have a module, you can add the code that will create and invoke an ADO command object.

To use the IBM DB2 Development Add-In to create the ADO command code:
1. Put your cursor at the end of the generated code.
2. In the DB2 Development View, right-click the stored procedure **XWishedTitlesAdd**, and click **Add ADO command code**. Command code is added at the end of the DBUtil module.
3. In the Project Explorer window, right-click **DBUtil**, and click **Save DBUtil**. The Save File As window opens.
4. Click **Save**.

Repeat steps 1 through 4 for the following stored procedures:
- GetNonWishedTitles
- XGetWishedTitles

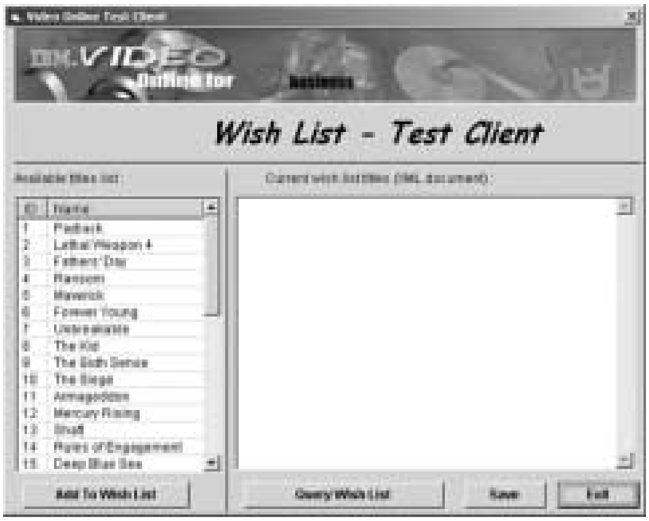
You have successfully created and saved the ADO connection module.

Checkpoint

Now that you created and saved the ADO connection module, you need to add the test client project and set the test client form as the project startup object.

To add the test client project:

1. Click **File —> Add Project**. The Add Project window opens.
2. Click the **Existing** tab.
3. Select the `videoonline\VB\Client.vbp` file.
4. Click **Open**. You will see the client added to your Properties window and Project Explorer window.
5. In the Project Explorer window, right-click **Client (Client.vbp)**, and click **Set as Start Up**.
6. Click **File —> Save Project Group As**. The Save File As window opens.
7. Click **Save**.
8. Click **Start** ▶. The test client application will run.



ID	Name
1	Payback
2	Lethal Weapon 4
3	Fathers' Day
4	Ransom
5	Maverick
6	Forever Young
7	Unbreakable
8	The Kid
9	The Sixth Sense
10	The Siege
11	Armageddon
12	Mercury Rising
13	Shaft
14	Rules of Engagement
15	Deep Blue Sea

You can experiment with the application by selecting and adding titles to the wish list. You can then query the wish list to validate that the titles have been added.

To add a title or titles to your wish list:

1. In the **Available titles list**, select the title or titles that you want to add to your current wish list. You can select more than one title by holding down the Ctrl key.
2. Click **Add To Wish List**. An XML message window will open showing the titles that are being added.
3. Click **OK**.

To view your current wish list titles:

1. Click **Query Wish List**. You can see the added titles in the **Current wish list titles (XML document)** list.
2. Click **Save**. Your wish list is saved to `videoonline\wishlist.xml`.

Summary

This tutorial taught you how to use the IBM DB2 Development Center Development Add-In for Microsoft Visual Basic to build a key component of the Video Online application. The Development Center is used to build the layer that directly interacts with the database, called the data layer. To complete this layer component, you first learned about the three-tiered Video Online architecture and design, then created and populated the database. Next, you prepared the database and environment, created the Visual Basic project and started the DB2 Development Center Development Add-In. Finally, you built UDFs and stored procedures, and added the database utility module.

Additional information

The following resources can provide more information about various topics related to this tutorial:

- IBM Video Sample Solutions Family Web site. This Web site has all of the IBM video sample solution resources including sample code, tutorials, and articles. The Video Online solution and the Video Central solution are included in the Web site.

  http://www.ibm.com/software/data/developer/samples/video
- IBM DB2 Development Center Web site.

  http://www.ibm.com/software/data/db2/udb/dc
- IBM DB2 Developer Domain. This Web site has technical information on the DB2 and Data Management development platform.

  http://www.ibm.com/software/data/developer
- Writing Applications Using the IBM OLE DB Provider for DB2. This paper includes detailed information about the supported applications, interfaces, properties, data services, and OLE DB services.

  ftp://ftp.software.ibm.com/ps/products/db2/info/vr7/pdf/letter/db2age70.pdf
- Microsoft Universal Data Access Web site - OLE DB Section. This Web site includes detailed information about the OLE DB technology, including a link to product information and a white paper, as well as recent OLE DB news headlines.

  http://www.microsoft.com/data/oledb
- Microsoft Component Object Model (COM) technologies Web site. This Web site offers white papers, articles, presentations, and links about COM-based technologies such as Distributed COM (DCOM), COM+, MSMQ, Microsoft® Transaction Server (MTS), ActiveX® Controls, and more.

  http://www.microsoft.com/com/default.asp
- World Wide Web Consortium Extensible Markup Language Web site. This Web site offers multiple resources including papers, articles, developer discussions, and more about XML.

  http://www.w3.org/XML/
- O'Reilly xml.com – XML from the inside out Web site. This Web site offers the latest news, resources, and technical articles about XML related technologies.

  http://www.xml.com/

Notices

Trademarks

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-237-5511 for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Printed in U.S.A.