Data Bases and ADO.NET

Relational Databases

- Most data handling today done with <u>relational databases</u>
  - Logical representations of data that allow relationships among data to be considered without concern for the physical structure of the data
  - Composed of tables (like spreadsheets)
  - Lots of proprietary formats
  - Some database sources:
    - Microsoft SQL Server
    - Access
    - Oracle
    - Sybase
  - Visual Studio .NET can handle data from multiple locations (servers) stored in different formats

ADO.NET

- Based on Microsoft's ActiveX Data Objects
- Big advantage:
  - Data stored and transferred in Extensible Markup Language (XML)
  - Allows access to database data in many formats
  - Two basic types of database connections:
    - SQLClient for SQL Server
    - OLEDB for all other database formats
      - Can be used to obtain/update data from sources such as Access, Oracle, Sybase, DB2, etc.

Database Terminology

- Each database file can hold multiple tables
- A <u>table</u>:
  - Each row represents data for one item
    - Called a <u>record</u>
  - Each column used to store a different data element
    - Elements represented in columns are called <u>fields</u>

```
           Last Name     First Name     Phone
           ---------------------------------------------
Records    Smith         John           777-1111
           Jones         Mary           777-2222
                             Fields
```

Database Terminology, continued

- <u>Primary Key Field</u>
  - Used to identify a record in a table
  - A field that contains unique data not duplicated in other records in the table
    - e.g., social security number for employees
- <u>Current Record</u>
  - Anytime a table is open, one record is considered to be the current record
    - As we move from record to record in a table the current record changes

Queries

- A <u>query</u> retrieves information from a database
- <u>SQL</u> (<u>Structured Query Language</u>) is the standard for expressing queries
  - We won't need to be experts in using it since Visual Studio .NET provides a "Query Builder" tool to construct SQL queries

XML Data

- Industry standard for storing and transferring data
  - Specs at: www.w3.org/XML
- Most database formats store data in binary
  - Cannot be accessed by other systems or pass through firewalls
- Data stored in XML is text
  - Identified by tags similar to HTML tags
    - Not predefined as in HTML
    - We can define our own XML tags to indicate their content
      - So very flexible for describing any kind of data
- Use of XML allows programs to communicate even though they are written in different languages and run on different hardware

Overview of XML

- Machine-Readable and Human-Readable Data
- Defines the Data Content and Structure
- Separates Structure from Presentation
- Allows Developer to Define his/her Own Tags and Attributes

```
<employee>
  <name>Jake</name>
  <salary>25000</salary>
  <region>Ohio</region>
</employee>
```

XML Schemas

- A schema describes fields, data types, and any constraints on the data
- Defines the structure of an XML document
- A schema is expressed in XML as well
- Use of schemas permits strong typing and data validation

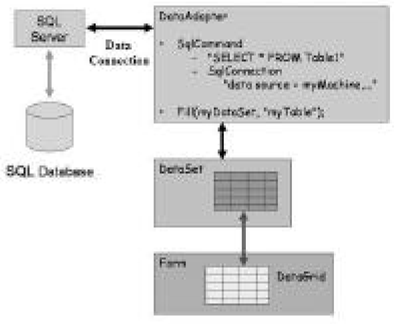
Using ADO.NET

- Data from a database can be displayed on a Windows Form or a Web Form
- Add controls to the form and bind the data to the controls
  - Controls can be what we've already seen:
    - label, text box, list box, combo box, etc.
  - Or special controls designed just for data:
    - DataGrid, DataList
- ADO.NET classes are in the System.Data namespace

ADO.NET Data Access

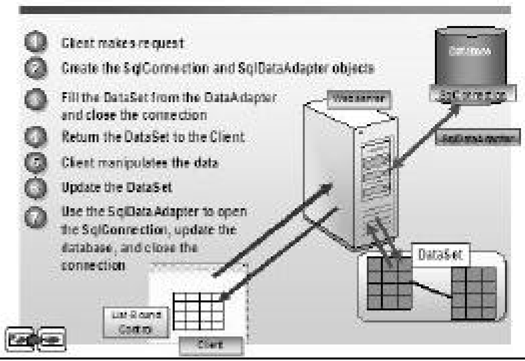
- 1. Set up a data connection
  - Establishes a link to the data source, which is a specific database file and/or server
- 2. Set up a data adapter
  - Handles retrieving and updating the data
  - Data adapter uses "Command" objects to retrieve/store records from the database and can be used to:
- 3. Create a dataset:
    - A temporary set of data stored in the computer's memory
    - ADO.NET datasets are disconnected
      - So data in memory does not keep an active connection to data source
      - Much better: Many more clients can connect and use the data serv er
    - Data adapters's Fill() method gets the data into the dataset
      - Uses SQL in a "Command" object to specify data to retrieve/update
- 4.. Add controls on the Windows Form or Web Form
  - Display the data from the dataset and allow user interaction
- 5. Write C# code to fill the dataset

Connections, Data Adapters, Datasets



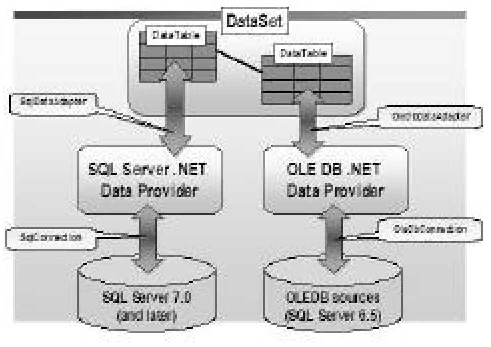
```
• SqlCommand
    – "SELECT * FROM Table1"
    – SqlConnection
        "data source = myMachine..."
• Fill(myDataSet, "myTable");
```

Accessing a Database through a Web Server


1. Client makes request
2. Create the SqlConnection and SqlDataAdapter objects
3. Fill the DataSet from the DataAdapter and close the connection
4. Return the DataSet to the Client
5. Client manipulates the data
6. Update the DataSet
7. Use the SqlDataAdapter to open the SqlConnection, update the database, and close the connection

Creating a Connection

- ADO.NET provides two types of Connection objects:
  - SqlConnection
    - Only for connecting to a Microsoft SQLServer database
  - OleDbConnection
    - For connecting to all other database systems
- Can use Visual Studio's "Server Explorer" to set up data connections
  - Start it with "View" | "Server Explorer"


DataSet
DataTable
DataTable
SqlDataAdapter
OleDbDataAdapter
SQL Server .NET Data Provider
OLE DB .NET Data Provider
SqlConnection
OleDbConnection
SQL Server 7.0 (and later)
OLEDB sources (SQL Server 6.5)

An Example Session: Using Visual Studio to Set Up an ADO.NET Application

- This example works with a small Access database
  - rnrbooks.mdb
- Contains two tables:
  - "Books" with the following fields:
    - ISBN, Title, Author, Publisher, SubjectCode, ShelfLocation, Fiction
  - "Subjects" with the following fields:
    - SubjectCode, Subject

Creating a New OleDbConnection (Access DB) in a Windows Forms App

- In the Visual Studio Tool Box click on "Data" and drag an OleDbConnection object to the form
- Select "Connection String" in the Properties window and click on "New Connection"
  - Could also right click on "Data Connections" in "Server Explorer" and Select "Add Connection"
- "Data Link Properties" Dialog Box appears
  - Select "Provider" tab & choose "Microsoft Jet 4.0 OLE DB Provider" and click "Next"
  - On the "Connection" tab | "1. Select or enter a database name", click "…" and navigate to the directory with the rnrbooks.mdb Access database file
  - Select the rnrbooks.mdb file and click on the "Open" button
  - Click on "Test Connection" and you should see a "Test connection succeeded" message box

Setting Up the Data Adapter

- In "Server Explorer", expand the new data connection and its "table" | "books" table
- select desired fields
  - (e.g., ISBN, Title, Author fields from rnrbooks.mdb)
    - <Ctrl>-click on each to do this
- Drag selected fields to the form
  - A new OleDataAdapter1 object will appear in the component tray

Generate Data Adapter's SQL

- Click somewhere else in the component tray to deselect the two data components
- Right-click on OleDbDataAdapter1 and select "Configure Data Adapter"
  - "Data Adapter Configuration Wizard" appears
    - Your connection should appear selected
  - Click "Next"
- You must use SQL statements for an Access database, so click "Next"
  - "Generate the SQL statements" page comes up
  - Note the default is to select all the chosen fields from "Books" Table
    - You can tailor the SQL query by using "Query Builder"
- This generates OleDbCommand objects w/ correct SQL
- Click "Finish"

Rename the Data Components

- Select OleDbConnection1
  - In Properties window change its name to conBooks
- Select OldebDataAdapter1
  - In Properties window change its name to daBooks

Generating a Dataset

- Select daBooks data adapter and Choose "Generate Dataset" from "Data" menu item
  - A "Generate Dataset" dialog box appears
- Name the new data set dsBooks & click OK
  - A new dsBooks1 dataset is added to the component tray
  - In Solution Explorer there's a new file called dsBooks.xsd
    - This contains the schema for the new dataset
      - It's a new class, and the new component in the tray is an instance of that class
        - » Look at Class View to see the new class
      - In code you write, refer to dsBooks1 (the dataset object), which is an instance of the dsBooks class
    - Double click on dsBooks.xsd to see the schema and the XML that has been generated for you by the Designer

Adding a DataGrid to the Form

- Set the form's <u>Name</u> property to BookForm and its <u>Text</u> property to Book Data
- Widen the form and drag a large DataGrid control from the tool box
- Change DataGrid's <u>Name</u> property to booksDataGrid
- Set its <u>DataSource</u> property to dsBooks1.Books
  - Grid now has column headings for the selected fields in the data set
  - The data has been "bound" to the control

Coding the Application

- Double-click on the form (not on the grid) to add a BookForm_Load event handler
- Type the following code into BookForm_Load event handler:

```
daBooks.Fill(dsBooks1);
```

  - Causes the data adapter's " SelectCommand" to be sent to fill the dataset with items specified in the query
  - [If this were a Web app, you would need to bind the DataGrid control with the following code: booksDataGrid.DataBind();]
- Change the Run parameter in Main()

```
Application.Run(new BookForm());
```

- Run the application
  - Grid should fill with data from the database file
  - Note that you can change the widths of the fields
  - And that you can sort according to any of them by clicking on the field name

Binding Data

- Connecting a control or control property to one or more data elements
- Set <u>DataBindings</u> in form's properties window
- <u>Simple Data Binding</u>
  - Connects one control to one data element
    - Use to display a field value in controls that display one item
      - e.g., text box or label controls
  - Do at design time using control's property window, or in code, e.g.

```
textBox1.DataBindings.Add("Text", dsBooks1, "Books.Author");
```

- <u>Complex Data Binding</u>
  - Connects a control to more than one data element
    - Used with DataGrid, DataList, ListBox, ComboBox controls

```
dataGrid1.SetDataBinding(dataSet1, "Books");
```

- After binding, we can navigate through the data in the data set and the data that appears in the bound controls will be updated automatically as we move from record to record

Executing SQL Commands

- After a DataAdapter populates a data set, the data adapter has the following commands:
    - DeleteCommand, InsertCommand, SelectCommand, UpdateCommand…

  These are OleDbCommand objects that specify how the data adapter deletes, inserts, selects, and updates data in the database
  - Set their CommandText property to the SQL to be executed in a query
  - The data adapters's Fill() member causes its SelectCommand to execute
  - The data adapter's InsertCommand has an ExecuteNonQuery() member function that will execute the action described in the SQL of the InsertCommand's CommandText
  - It works the same way for the DataAdapter's UpdateCommand and DeleteCommand

Example of User-entered SQL Queries

- User types a database query into a text box, result is displayed in a datagrid control
- Creating the application with Visual Studio:
  - If Connection already exists:
    - Drag an OleDbDataAdapter from the Data Toolbox
      - In resulting Wizard select the Connection, Keep the default "Use SQL Statement", click "query builder", select the "Books" table from "Add" menu and "Close", Mark "All" columns
    - Drag a DataSet from the Data Toolbox
      - Choose "Untyped DataSet"
    - Drag a Textbox for the user query, a "Submit Query" button, and a DataGrid to hold the result

Coding the Application

- Submit Button Click handler:

```
oleDbDataAdapter1.SelectCommand.CommandText = textBox1.Text;
dataSet1.Clear();
oleDbDataAdapter1.Fill(dataSet1, "Books");
dataGrid1.SetDatabinding(dataSet1, "Books");
```

- This code should really be put inside try/catch blocks

An Example of Viewing Selected Records & Adding New Records to a Data Base

- Works with the same Books database
  - User enters a Title
    - "Find" button causes the Title, ISBN, and Author of that book to be displayed in textboxes
  - User changes ISBN and/or Author
    - "Update" button changes the data for that Title in the database
  - User enters a new record (Title, ISBN, Author)
    - "Add" button inserts the new record into the database

Creating & Coding the Application

- Create connection, DataAdapter, and DataSet as in the previous example
- Create the controls with the VS Designer
- In Form1 constructor, open the Data connection:

```
oleDbConnection1.Open();
```

Button Click Handlers

- "Find" button:

```
dataSet1.Clear();
oleDbDataAdapter1.SelectCommand.CommandText = "SELECT * FROM
    Books WHERE Title = '" + textBoxTitle .Text + "'";
oleDbDataAdapter1.Fill(dataSet1);
Display(dataSet1);   // Display() helper function puts new data into textboxes
```

- "Add" Button:

```
oleDbDataAdapter1.InsertCommand.CommandText = "INSERT INTO
    Books (" +
"ISBN, Title, Author" +
") VALUES ('" +
textBoxISBN.Text + "', '" +
textBoxTitle.Text + "', '" +
textBoxAuthor.Text + "')";
oleDbDataAdapter1.InsertCommand.ExecuteNonQuery();
```

- “Update” Button:

```
oleDbDataAdapter1.UpdateCommand.CommandText =
"UPDATE Books SET Author = '" +
textBoxAuthor.Text + "'," +
" Title = '" + textBoxTitle.Text + "'" +
" where ISBN = " +         "'" +
textBoxISBN.Text + "'";
```

Display(…) Helper Function

```
private void Display(DataSet dataSet)
        {
                DataTable dataTable = dataSet.Tables[0];
                if (dataTable.Rows.Count != 0)
                {
                        textBoxISBN.Text =(string)dataTable.Rows[0][0];
                        textBoxTitle.Text =(string)dataTable.Rows[0][1];
                        textBoxAuthor.Text =(string)dataTable.Rows[0][2];
                }
        }
```

- Note the DataTable object retrieved from the DataSet
  - Tables[0] is the first table in the dataset (there’s always at least one)
  - Contains the table resulting from the SQL query
  - Rows property give us access to all records retrieved
    - It’s like a 2-D array – [Row #] [Column #]
    - Count property gives us the number of rows in the table
- The code puts the data into the three text boxes

Navigating through Datasets Using Bound Controls

- Simple-bound controls display data from a single record at a time
- It’s common to add navigation controls like buttons to move from one record to another
- Use a Binding Context

BindingContext

- BindingContext is a property of a form
- Responsible for moving through the records in a dataset table and ensuring that all of the controls on the form display data from the same record and table
- Holds a collection of BindingManagerBase objects
  - There’s one BindingManagerBase object for each table
  - A BindingManagerBase object is retrieved from a form’s BindingContext by specifying the dataset object and the table in that object
    - Then we can manage the synchronization of data from that table with the control’s forms by getting/setting the BindingManagerBase object’s properties such as:
      - Position: gets or sets the current record
      - Count: returns the number of records

Using BindingContext and BindingManagerBase

- Use a class-level BindingManagerBase variable and assign a dataset table from the form’s BindingContext collection in the Form_Load event handler (after filling the dataset)

```
BindingManagerBase bmBooks;
private void Form1_Load(object sender, System.EventArgs e)
{
    daBooks.fill(dsBooks1);   // Fill the dataset
    bmBooks = this.BindingContext[dsBooks1, “Books”];
            // the Books table in the dataset dsBooks1
}
```

- Now we can use the Position property of bmBooks to navigate through the dataset
  - All bound controls display element at current Position

Example Program: DataBooks

- Works with the rnrbooks.mdb Access database
- Form has “Author”, “ISBN”, and “Title” Label controls
- “>” button moves one record forward
- “<” button moves one record backward
- “First” button displays first record
- “Last” button displays last record

Detailed Steps for Creating DataBooks Application

- New Windows Application
- Server Explorer:
  - Should still show the connection for rnrbooks
  - Expand the "Table" | "Books" nodes and drag ISBN, Title, and Author fields over to the form
    - Creates the connection and data adapter components
    - Name them conBooks and daBooks
- Select the Data adapter component (daBooks) and generate the dataset, calling it dsBooks
  - (Menu: "Data" | "Generate Dataset")
  - A dsBooks1 dataset will be generated

- Drag over three labels
  - Name them: authorLabel, IsbnLabel, titleLabel
  - Set BorderStyle to Fixed3D
- Drag over Buttons: ">" (nextButton), "<" (previousButton), "First" (firstButton), "Last" (lastButton)
- Expand DataBindings property of each control
  - For each Select "Text", drop down the list, and expand dsBooks1
    - For each, select the following for the Text property:
      - authorLabel: Author
      - IsbnLabel: ISBN
      - titleLabel: Title
- Declare the binding manager (bmBooks) and write code for the form1_Load event handler

---

- Add click event handlers for each button
- Add code to handlers:
  - nextButton_Click()

    bmBooks.Position ++;
  - previousButton_Click()

    bmBooks.Position --;
  - firstButton_Click()

    bmBooks.Position = 0;
  - lastButton_Click()

    bmBooks.Position = bmBooks.Count – 1;

Displaying the Record Position and Number of Records

- Add another label control to the DataBooks app that will display the current record number & the total number of records
- Since we want this label control to be updated for any of the buttons clicked, write a single DisplayRecPosn() helper function to do the work
  - Use bmBooks.Count property to get the number of records
    - Or alternatively the Tables["table-name"].Rows.Count property of the dataset
  - Set the Text property of the new label to a string containing bmBooks.Position + 1 (converted to a string)
- Call DisplayRecPosn() in each button click handler

Using a List Box or Combo Box

- Any of the toolbox "list" controls can be automatically filled with values from a dataset
  - Set control's DataSource and DisplayMember properties
    - DataSource connects to the specified dataset
      - e.g., dsBooks1 (the dataset)
    - DisplayMember connects to a specified field for the data to be displayed in the control
      - e.g., Books.Title (table and field name in database)
  - Must use the data adapter's Fill() method

    daBooks.Fill(dsBooks1);

Using Mutiple Data Adapters

- Needed when you are using more than one dataset
- Or you're using more than one table in a dataset
- A single connection object can supply the connections for multiple data adapters
- In many cases the information for the second dataset is not known until run time
  - e.g., one dataset for a list of titles
  - Another for a selected record
  - User selects a title from the list
  - ISBN and author displayed in two text boxes are those of the book whose title was selected by the user

Creating a Parameterized Query

- If the dataset is to include only certain records:
  - Modify the SQL SELECT statement in the command used by the data adapter
    - i.e., put a WHERE clause in the SQL, for example:
      SELECT Title, Author, ISBN FROM Books WHERE Title = “The Stand”
      or SELECT Title, Author, ISBN FROM Books WHERE Title = ?
    - This can be coded manually
  - Easier to use Visual Studio’s Query Builder
    - Select the data adapter
    - Choose “Data” | “Configure Data Adapter” from menu
      - Opens the wizard
      - On fourth page of wizard choose “Query Builder”
      - Click to choose desired fields
      - Type “=?” in “Criteria” column of the field you want (Title)
        - » A parameter
        - » The actual value will be provided at run time
    - Query Builder creates the correct SQL command

Displaying Data for a Selected Item

- The value for a parameterized query is provided at run time
  - Could come from user input into a text box or could be an item selected from a list box or combo box
- Specify parameter value using the <u>Parameters</u> collection of the data adapter’s <u>SelectCommand Value</u> property
  - For example if Title is to come from a titlesTextBox:
    daBooks.SelectCommand.Parameters[“Title”].Value = titlesTextBos.Text;
  - This must be done before filling the data adapter

Example Program: DataBooksCombo

- Form has a combo box to display all the titles in the Book table of the rnrbooks.mdb Access database
- When user selects a title:
  - The ISBN and Author of the selected book will appear in two label controls
- Application has two data adapters
  - daTitles: all records in the Books table
  - daBooks: only the record that matches the Title selected by user
- Handler for the SelectedIndexChanged combo box event makes the selection and updates the text property of the label controls

Using ADO.NET with Web Forms

- Because of client/server/client round trips and stateless nature of web pages, all controls must be explicitly bound using control-ID.DataBind();
- Also, recall that in a web app similar to DataBooksCombo, each time user makes a selection from the dropdown list, a postback occurs
  - After postback, the Web page redisplays and the Page_Load event occurs
    - Logic in Page_Load event handler must be modified or dataset for the list elements will be recreated
    - Use the fact that a page’s IsPostBack property is set to false the first time a page displays and true every time after that
- Finally, for list controls the AutoPostBack property must be set to true for the SelectedIndexChanged event handler to execute on the server

Some Code for Web Forms

```
private void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
    {
       daTitles.Fill(dsTitles1);
       titlesDropDownList.DataBind();
    }
}

private void TitlesDropDownList_SelectedIndexChanged(object sender, System.EventArgs e)
{
    if (TitlesDropDownList.SelectedIndex != -1)
    {
       dsBooks1.Clear();
       daBooks.SelectCommand.Parameters[“Title”].Value =
           titlesDropDownList.SelectedItem.Text;
       da.Books.Fill(dsBooks1);
       IsbnLabel.DataBind()
       authorLabel.DataBind();
    }
}
```

Making ADO.NET Projects Portable

- When moving DB projects from one computer to another, connection information must be changed
- Database must be available on new computer
  - Or <u>ConnectionString</u> must specify where it is
- Easiest to put database file in the project’s bin directory and change the <u>DataSource</u> in the <u>ConnectionString</u> in the Form_Load event handler:

```
Private void Form1_LOAD(object sender, System.EventArgs e)
{
   conRnR.ConnectionString =
     “Provider=Microsoft.Jet.OLEDB.4.0;DataSource=rnrBookd.mdb”;
   daTitles.Fill(dsTitles1);
}
```

- <u>DataSource</u> can be another machine/file