Data Bases and ADO.NET

Relational Databases
- Most data handling today done with relational databases
  - 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
  - Data stored and transferred in Extensible Markup Language (XML)
  - Allows simple access to database data in many formats
    - Easy-to-use classes represent tables, columns, rows inside relational databases
    - Introduces DataSet class representing a set of data from related tables encapsulated as a single unit preserving the integrity of the relationships between them
  - 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.
    - Many others supported

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

<table>
<thead>
<tr>
<th>Last Name</th>
<th>First Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>John</td>
<td>777-1111</td>
</tr>
<tr>
<td>Jones</td>
<td>Mary</td>
<td>777-2222</td>
</tr>
</tbody>
</table>

Database Terminology, continued
- **Primary Key Field**
  - 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

- **Current Record**
  - 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 query retrieves information from a database
- SQL (Structured Query Language) 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:
    • DataGridView
• 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 server
  • Data adapter’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

Connections, Data Adapters, Datasets
Accessing a Database through a Web Server

Creating a Connection

• ADO.NET provides several types of Connection objects
• Two important ones:
  – SqlConnection
    • Only for connecting to a Microsoft SQL Server database
  – OleDbConnection
    • For connecting to other database systems such as Access
• Can use Visual Studio’s “Server Explorer” to set up data connections
  – Start it with “View” | “Server Explorer”

An Example Session: Manual Coding to Read from a DB Table

• Windows App Example: DataReadingWithDataSet
  – Reads data from a small Access database: rnrbooks.mdb
  • Contains two tables:
    – “Books” with the following fields:
      » ISBN, Title, Author, Publisher, and other fields
    – “Subjects” with the following fields:
      » SubjectCode, Subject
  – Instantiates and opens an OleDbConnection to the DB
  – Creates an OleDbDataAdapter with an SQL SELECT command using the Connection
  – Instantiates/Fills a DataSet with data from one of the DB tables using the DataAdapter
  – Indexes through the rows of the table to get and display values of two fields of a table in a multiline text box

Changing the Contents of a Database

• Change actions:
  – Updating, Inserting, Deleting records
• All done in the same way:
  – Fill a DataSet with the data
    • As in previous example
  – Modify the data in the DataSet (Update, Insert, or Delete records)
    • Can use a CommandBuilder
  – After modifications, persist the DataSet changes back to the database
• See DataUpdate06 for an updating example

Adding a Row

• Again set up a Connection and a DataAdapter
• Create a CommandBuilder object
• Create/Fill a DataSet
• Create a new row with table’s NewRow() method
  rowData = thisDataSet.Tables("Books").NewRow();
• Give values to all its fields
  rowData["Title"] = New Title;
  rowData["Author"] = New Author;
• Add the row with the table’s Rows.Add() method
  thisDataSet.Tables("Books").Rows.Add(rowData);
  – Row will be added and Rows.Count property will be incremented
• Update the Adapter
  thisAdapter.Update(thisDataSet, "Books");
  – Only the changed fields are changed; the rest are left blank
Deleting a Row

- After setting up the Connection, DataAdapter, CommandBuilder, and DataSet:
  - Find the row to be deleted:
    - Determine the primary key before filling the data set:
      ```csharp
      thisAdapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
      thisAdapter.Fill(thisDataSet, “Books”);
      ```
    - Use the Find() method to find the row:
      ```csharp
      DataRow foundRow = thisDataSet.Tables[“Books”].Rows.Find(“222 - 444”);
      ```
    - Returns a DataRow if successful, null if not
  - Delete the row using the Delete() method:
    ```csharp
    foundRow.Delete();
    ```
  - Finally make change permanent with an Update:
    ```csharp
    thisAdapter.Update(thisDataSet, “Books”);
    ```

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
    ```csharp
    thisAdapter.SelectCommand.CommandText = “SELECT Title FROM Books”;
    ```
  - The data adapters’s Fill() member causes its SelectCommand to execute
  - The DataAdapter’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

Using Visual Studio Designer to Set Up Access to the Data Base

- The tasks of setting up the DataConnection, the DataAdapter (DataTable), and the DataSet are automated
- In addition VS facilitates simple navigation through Dbase tables with a Binding Navigator object
- Result is a DB application with a LOT of functionality without writing any code

Creating a Data Base Project with Visual Studio 2005

- Start a new VS Windows Application
  - Change Name and Text properties
- Setting up the Database Connection
  - Menu: “Tools” | “Connect to Database”
  - Brings up “Add Connection” Dialog Box
  - Choose Microsoft Access Database File
    - If you’ve installed SQL Server Express and the Northwind Dbase, you can choose “Microsoft SQL Server”
  - Brings up “Add Connection” Dialog Box
    - Browse to directory containing the dbase file (e.g., rnrbooks.mdb)
    - Leave User name and Password with default values
      ```csharp
      – Test the Connection and then click “OK”
      ```

Adding a Data Source to the App

- Go to Design View
  - Menu: “Data” | “Add New Data Source”
    - Brings up “Data Source Configuration Wizard”
    - Choose “Database” and click “Next”
    - In resulting “Choose your data connection” box the connection to rnrbooks.mdb should be there. Select it an press “Next”
    - Take default of saving connection string in the config file
    - In the resulting “Choose Your Database Objects” box, expand the tables and choose the fields you want to access
      - (e.g., ISBN, Titles, Author)
      - Click “Finish”
Using the Data Source in the App

• Menu: “Data” | “Show Data Sources”
  – Brings up a “Data Sources” Window
• Add Data-Bound Controls to the form
  – Expand the Books node in Data Sources
  – Drag each field node over to the form
    • Visual Studio will create data-bound text boxes with appropriate labels on the form
    • Also creates a Binding Navigator toolbar underneath the form’s title bar
      – Permits navigation through the rows of the database
    • Also in area below the form a DataSet, a BindingSource, and a TableAdapter object are created
      – TableAdapter is a single-table version of a DataAdapter
• Run the application – Lots of new toolbar functionality without writing any code!!

Adding a DataGridView Control to Form

• Displays all the records in the Database table in a spreadsheet-like format
• Very easy to use VS Designer to add the control:
  – Just drag the desired table from the Data Sources window
  – Resize resulting DataGridView control on the form
  – Run the program
    • DataGridView control is already connected to the database
      • If you click on any row in the grid the data in the other controls change to match the selected row
      • No code needs to be added – Visual Studio generated all the needed code

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
• Set DataBindings in form’s properties window
• Simple Data Binding
  – Connects one control to one data element
  • Use to display a field value in controls that display one item (e.g., listbox)
  – Do at design time using control’s properties window, or in code:
• Also, in a web app with a listbox, each time user makes a selection from the list, a postback occurs
  – After postback, the Web page redisplay and the Page_Load event occurs
    • Logic in Page_Load event handler must be modified or the 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
    • For list controls AutoPostBack property must be set to true for SelectedIndexChanged event handler to execute on the server

Some Code for Web Forms

```csharp
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 ConnectionString must specify where it is
• Easiest to put database file in the project’s bin directory and change the DataSource in the ConnectionString in the Form_Load event handler:

```csharp
private void Form1_Load(object sender, System.EventArgs e)
{
    conRnR.ConnectionString =
        "Provider=Microsoft.Jet.OLEDB.4.0;DataSource=rnrBookD.mdb";
    daTitles.Fill(dsTitles1);
}
```

• DataSource can be another machine/file