Chapter 8

Web Database Applications
Objectives

- Display and update database files in Web applications
- Access data with a *data source control*
- Display data using a grid, details, and/or a form view
- Select records from a list box by using a parameterized query
- Display data from related tables
- Determine when to use a data reader rather than a dataset
- Set up a multitier Web application
- Maintain state for data in a multipage application
Data Access in ASP.NET

• ASP.NET database controls differ from those used in Windows
  – Add a data source to the Web form and configure it for data access
  – ASP.NET does not provide support for dragging fields to the form for automatically creating data-bound controls

• By default, a Web page displays data from bound fields but does not pass changes back to the data source
Data Source Controls

• Handle all of the database access for a Web page
  – Select records to display, insert, delete, and edit records for database updates
  – Automatically opens a connection, executes commands and closes the connection

• The toolbox contains several types of data source controls in the *Data* section and controls for displaying data on a Web page
Adding Data to a Web Project

• Include the database in the project folder
• Create a new Web site project
  – Right-click the App_Data folder and select Add Existing Item from the context menu
  – Browse to find the database and add it to the project
  – Complete this step before configuring a data source
Adding a SqlDataSource Control

• Drag a SqlDataSource to a Web page
  – Control appears at design time but not at run time
  – Open the smart tag and select Configure Data Source
  – Set up the SQL SELECT statement using the Configure Data Source wizard
Making the Connection String Dynamic

• The connection string is automatically created and placed in the Web.config file with a hard-coded path to the directory.

• To make a project portable, delete the absolute path name and substitute |DataDirectory|
  – Indicates database is in the project’s App_Data folder.
Displaying Data in a GridView Control

• The smart tag on the GridView allows an auto format to be set as well the data source to be chosen
  – No code needs to be written
  – The Smart tag for the grid also offers options for pagination and sorting
Displaying Data in a DetailsView Control

• Displays data one record at a time
  – No code to write, add the data source and a detail control
  – Configure the data source and assign the data source to the detail data control
  – Customize the display of data by selecting *Edit fields* from the DetailsView’s smart tag
Displaying Data on a FormView Control - 1

- Displays one record at a time
  - More control of the field layout than with a DetailsView control
- Add and configure the data source and then add the **FormView control**
  - Links to additional records are created by selecting *Enable Pagination* on the smart tag
Displaying Data on a FormView Control - 2

- Select *Edit Templates* from the smart tag
- With *ItemTemplate* selected, type text directly in the template on the form
  - Modify headings
    - Change the order of fields, delete fields, change label text, change type of controls
  - When finished, select *End Template Editing*
Examining the Data Bindings

- Data bindings for Web applications are very different from the Windows version
- To view bindings, select the smart tag on the FormView control, select *Edit Templates*
  - Select *Edit DataBindings* from a control’s smart tag to bind to a data field
- Check the *Show all properties box* to see how many different properties can be used for binding
  - Use the Text property to display data
Displaying Data in a Drop-Down List

• The DropDownList control can be bound to data and used for selecting a record
  – Similar to the combo box selection in Windows
  – Must use a parameterized query to display the result of the selection
• The ViewState property of the DropDownList control maintains the contents of the list from one invocation to the next
  – If the application has multiple pages that can be navigated, steps must be taken to maintain the list contents
Creating a Parameterized Query –
Step-by-Step

• Set up the database access
• Add and configure the drop-down list
• Add a second data source control
• Add a FormView control for the store data
• Format the FormView data in a table
Adding Related Tables – Step-by-Step

• Make a copy of the project folder
• Add a third data source
• Add a GridView control to display the sales data
• Format the grid
Displaying Related Data on Multiple Pages - 1

- User can click the hyperlink field to view data from the related table
### Displaying Related Data on Multiple Pages - 2

<table>
<thead>
<tr>
<th>Property</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataNavigateUrlFields</td>
<td>Column of table for lookup value.</td>
<td>stor_id</td>
</tr>
<tr>
<td>DataNavigateUrlFormat-String</td>
<td>Navigation link that passes a query string to the second page.</td>
<td>Sales.aspx?stor_id={0}</td>
</tr>
<tr>
<td>Text</td>
<td>Text to display for the link.</td>
<td>Sales</td>
</tr>
</tbody>
</table>
Creating Multiple Pages – Step-by-Step

- Create the project
- Set up the data source
- Design the first page
- Set up the second page
- Test the project
Selecting Data in a GridView

To create a Select button in the rows of a GridView, select Enable Selection in the grid’s smart tag

- Buttons can be displayed as a link, button, or image
- Respond to a Select button click in the GridView’s SelectedIndexChanged event handler
- Refer to individual cells as GridView1.SelectedRow.Cells(index)
Data Readers versus DataSets

• A *DataReader* object provides a forward-only result set from a data source
  – Forward-only means that access is fast
  – Data reader is not suitable for doing file updates
  – The sorting capability is disabled

• A data reader provides the quickest method for loading a list box or retrieving other small amounts of information that are not subject to change while a Web page displays

• Often a single application can contain a data reader and a dataset

• The *DataSource Method* property of the SQLDataSource control allows a choice of whether the file access uses a dataset or a data reader
• The SqlDataSource control has the ability to cache data to improve application performance
• The EnableCaching property is not set by default
  – The value can be set to `true`
• A different cache is kept for each connection and set of parameters
• Set the FilterExpression property of the data source control when using the caching feature
  – Allows the filtering of data in the cache without submitting a new request to the database
Web applications do not automatically configure data binding for updates.

- Allowing updates requires a change in the configuration of the data source.

• Click the Advanced button to specify update statements.
Updating a Database - 2

- Multiple users may update a database at the same time
  - Choose optimistic concurrency from the Advanced SQL Generation Options dialog box
Updating With a Data Source Control

• Adding basic updating features does not require coding

• Set the data source to the appropriate SQL update statements
  – The smart tag for the GridView, DetailsView, and FormView controls have extra check boxes that enable editing and deleting
Updating in a GridView

• Select *Enable Editing* and *Enable Deleting* on the smart tag
  – Adds a new column to the grid with *Edit* and *Delete* link buttons
  – User cannot make any changes until the *Edit* link button is clicked
    • Two link buttons for that row change to *Update* and *Cancel*
  – If the user clicks the *Delete* button, current record is deleted
    • Changes are immediately saved back to original data source
Allow Selection of Grid Rows

- On the grid’s smart tag check *Enable Selection*
  - Creates button labeled *Select*
  - Choose the option to *Edit Fields* and change many properties of the action buttons
    - Set the text of the buttons, column header text, and the *ButtonType*
Add a Drop-Down List to a Grid Cell

- Controls for each cell can be changed and a drop-down list can be provided for selection.
- Lists can be filled with fields from the grid’s data source or another data source.
Updating in a DetailsView or a FormView

- Users can choose to update in a GridView, a DetailsView, or a FormView
Updating Using a Drop-Down List for Navigation

• Write code to keep the list box contents up-to-date
  – If a record is added, modified, or deleted, rebind the drop-down list
    • Keeps contents in sync with the database

• For each event that could change the contents of the list, call the `DataBind` method of the drop-down list
Exception Handling

- Exceptions cannot be handled when performing updates without writing code
- Exceptions can occur when users add or edit data
  - Database constraints, such as a duplicate primary key and referential integrity violations
  - Bad input data and null fields
Catching Constraint Violations

- Code should be written in the event handlers for the SqlDataSource
  - The control has a set of events that take place right before (“ing”) and right after (“ed”) the action is taken
  - Place the code in the completed event handlers
- SqlDataSourceStatusEventArgs has an Exception object with a Message property and an ExceptionHandled property
  - Check for the existence of an exception
  - Display the error message
  - Set the ExceptionHandled to true so that the program can continue after handling the error
Validating Input Data

- Types of input validation performed depend on the requirements of the data fields
- To catch bad input before the update action use the “ing” events
- In the event handler arguments, the data values can be accessed
- When a value cannot be corrected in code, display a message and hold the screen for the user to make corrections
Multiple Tiers

• If it makes sense to separate a Windows application into multiple tiers, it makes twice as much sense for a Web application.

• In a Web application include the data access components in a separate data access component – Preferably a WCF Service component.
Using an ObjectDataSource

• Designed to connect to any type of object
• For example, can be used to connect data from a WCF Service to a data control on a Web page
Creating a Service for Database Updating – Step-by-Step

- Create an empty solution
- Add a WCF Service
- Rename the WCF Service
- Create the Dataset
- Write contracts and methods in the Service
- Set up the project to consume the data service
- Add controls to the default Web page
- Test the project
Maintaining the Selection of a Drop-Down List

- “Remember” which selection a user made from a drop-down list
  - Save the SelectedIndex property in a session variable

' Save the SelectedIndex in a session variable.
Session("CardType") = CreditCardDropDownList.SelectedIndex

- Restore the selection

' Restore the SelectedIndex from the session variable.
CreditCardDropDownList.SelectedIndex = _
    CType(Session("CardType"), Integer)
Maintaining the List in a Dynamic List Box

- To maintain state for a list box that has items added at run time, save the Items collection in a session variable:

  `Save the Items collection in a session variable. Session(“Titles”) = TitlesListBox.Items`

- To restore the Items collection, assign the session variable to the collection variable:
  
  – Loop through the collection, and add each item to the list
At times there is a need to write data from controls that are not bound to a database

- For example, online travel reservations show only selected value, not the entire reservations file

- Only the information that is needed must be transferred from the text boxes or other controls to a file
Data source controls include helpful feature to set up database queries for unbound controls

- For each of the SQL queries (SELECT, INSERT, UPDATE, and DELETE), display a Command and Parameter Editor dialog

- Click the build button for the InsertQuery property
Using Unbound Controls with a SqlDataSource

- *INSERT* command at the top of the window includes a parameter for each field of data
- Each parameter can connect to a parameter source
  - To connect to a control, select *Control* from the *Parameter source* drop-down list
    - Select the control from the *ControlID* drop-down list
Using Unbound Controls with an ObjectDataSource

- Set up the *Parameters collection* for each of the SQL commands to be used
Adding a Table from the IDE

- Add a new table from the Server Explorer window
  - Similar to process of adding a new stored procedure
  - Select the *Tables* node for the database, right-click, and select *Add New Table*
Viewing Data

• When testing an application, view the inserted records from the IDE
  – Select the database from the Server Explorer
  – Or–
  – Double-click on the database icon in the Solution Explorer
• Select the table and choose Show Table Data
Creating Custom Error Pages

• When working with a database, it is important to control what the user is allowed to see

• Protect the program by using a custom error page
  – Displayed instead of the default error message that may provide too much information

• Create the page and then set the Web.config file to indicate the page to use
  – Multiple pages can be set up and specify the page to use for specific error conditions
Using Validation Controls

• Use the validation controls to validate individual controls
  – Not available for GridView, DetailsView, or FormView

• May need to turn the validation on and off during program execution
  – Set the Enabled property to true to turn on and false to turn off
  – After setting the property to true, use the Page.Validate method to force a validation