Create and Use a Data Access Layer in a Web Application

By Susan L. Miertschin

Data Access Layer

Separate the data access logic from the presentation layer and from the business logic

Open a New Web Site

- Open Visual Studio
- File
  - New Web Site
    - Default.aspx
    - Web.config
    - App_Data folder
- Direct files to local location

Open a New Web Site

- Fix Default.aspx so that "A Simple Example of nTier Data Access" displays in the title bar of the browser at run time
- Fix the web.config file so that we can see error messages and so that Windows Authentication is turned off

Add a Database Reference

- Add a reference to the database in Visual Studio's Server Explorer
- The reference lets you use Visual Studio to:
  - add tables, stored procedures, views, etc.
  - view table data
  - Create queries by hand-coding or graphically via the Query Builder

Connect to the Database with Visual Studio's Server Explorer

- View
  - Server Explorer
- Create a new data connection (right click on Data Connections – Add)
- Data source type is Microsoft SQl Server
- Data provider is .NET Framework Data Provider for SQL Server
- Server name: as before in class

Connect to the Database with Visual Studio's Server Explorer

- Use SQL Server Authentication
- Use WebUser as the logon id for the Fallcreek database on the class SQL Server
  - Save the password in the connection string; it is WebUser (case sensitive)
- See screen shot on next slide

Add Connection Dialog Box

Server name changes from semester to semester.

Use SQL Server Authentication.

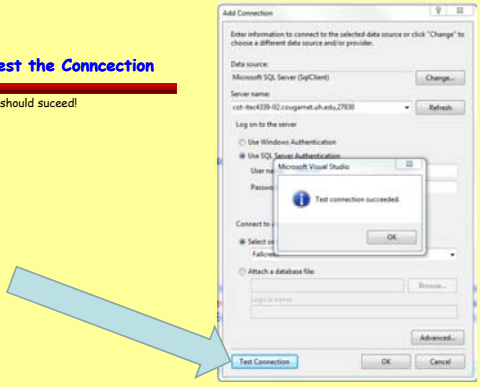
Use WebUser user name.

Password is WebUser.

Select Fallcreek Db to connect to.

Test the Conncection

It should suceed!



With the Data Connection in Visual Studio …

- You can see
  - Tables
  - Views
  - Stored procedures
  - User-defined functions
- Provided permissions have been granted

Data Access Layer

DAL

Alternative to a Data Access Layer

- Embed data-specific logic directly into the presentation layer
- Use the SqlDataSource control
- Ties the data access logic tightly to the presentation layer
- Not a recommended architecture for production – OK for development of a prototype

Data Access Layer – The Recommended Approach

- Coding layer separate from the presentation layer
- Typically implemented as a class library separate from the web application project

Data Access Layer Contents

- Code that is specific to the underlying data source
- Create a connection
- Issue SELECT, INSERT, UPDATE, DELETE commands
- Methods for accessing the underlying db data

Presentation Layer Interacts with DAL

- Presentation layer does not contain references to data access code
- Makes calls to methods and properties of the DAL for any and all data requests

Methods that Access Db

- Can return a DataSet or DataReader populated with data by a db query
- Ideally, the methods return strongly-typed objects

Strongly-Typed versus Loosely Typed Objects

Strongly Typed

- An object whose schema is rigidly defined at compile time
- Uses a typed DataSet
- Each column is implemented as a property

Loosely Typed

- An object whose schema is not known until runtime
- DataReader and DataSet
  - Their schema is defined by the columns returned by the database query used to populate them

Code to Access Info From a Loosely Typed Object versus a Strongly Typed Object

For Loosely Typed

```
DataTable.Rows(index,"ColumnName")
```

For Strongly Typed

```
DataTable.Rows(index).ColumnName
```

Creating Strongly Typed Objects

- Create business objects
  - Generally these are classes whose properties reflect the columns of the underlying db table (or view) that the business object represents
- FC example: a Patient class would have properties and methods that reflect the structure of the patient entity in the db
  - The patient business object might encompass more than one database entity (table)
- Use Typed DataSets
  - A class generated by Visual Studio based on the schema of a pre-existing db

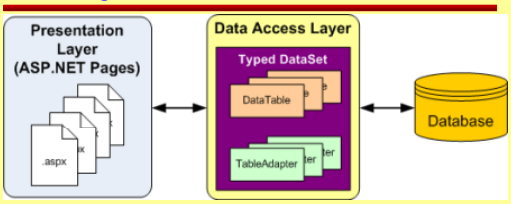
Typed DataSets

- A class generated by VS based on a db schema
- Members of the class are strongly-typed according to the schema

Typed DataSets

- Includes strongly typed DataTable classes
  - Do not have info about how to access data
  - Strongly typed objects used to pass data between application layers
- Includes TableAdapter Classes with
  - Info about how to access data
  - methods for populating datatable objects with values (data)
  - Methods for propagating modifications within the DataTables back to the db

Figure from Microsoft's Tutorial on Creating a DAL



http://www.asp.net/data-access/tutorials/creating-a-data-access-layer-vb

Add a Typed Data Set

- Right click on the project node in Solution Explorer
- Add New Item
- Select DataSet from the list of templates
- Name it staff_info.xsd
- Add it to the App_Code folder if Visual Studio suggests it

DataTable and TableAdapter

- Drag database items from Server Explorer onto the design interface staff_info.xsd
- Create a DataTable by dragging the staff_t table from the Server Explore onto the design interface

DataTable versus TableAdapter

- DataTables are strongly typed instantiations based on the database design
- DataTables do not contain information about how to access the data from the underlying database table
- Use a TableAdapter class to retrieve data to populate the DataTable at runtime

DataTable versus TableAdapter

- The TableAdapter class serves as the DAL
- Create methods for the TableAdapter class
- One way is to use the configuration wizard to do so

staff_t Table Adapater and DataTable

- Right click on staff_tTableAdapter and select Configure to open the configuration wizard
- The default SELECT query retrieves all the fields from the table
- Your application does not have to display all the fields retrieved – but what is the point of retrieving data the app does not need to display

staff_t Table Adapater and DataTable

- Accept the default query and click Next
- Two methods are generated with the wizard (could be renamed at this step)
  - Fill
  - GetData
- Select the option to Create methods to send updates (GenerateDBDirectMethods option)

Fill and GetData Methods

- Fill
  - Requires a DataTable object as input
  - Populates the runtime DataTable object with data (structures have to match)
- GetData creates a DataTable object and populates it with data and returns the object

staff_t Table Adapater and DataTable

- Click Next to see a list of tasks the Wizard completed
- Click Finish

Staff Information

- Examine the data available in the staff_t table by looking at the field names in the staff_t DataTable representation in the designer
- Is the information sufficient if you want to create a staff contact directory without addresses, but with phone numbers and email addresses?

Staff Information

- What other tables are needed to get the information we want?
  - Drag all the tables needed from the Server Explore onto the designer window
  - Notice that existing relationships are modeled and represented in the display as well

Generic TableAdapter from Toolbox

- Drag a TableAdapter from the Toolbox to the designer window
- Configure it:
  - To use the existing Fallcreek connection
  - Next
  - To Use SQL statements
  - Next

Generic TableAdapter from Toolbox

- Write a SLQ statement to retrieve staff name in the format first+MI+last together with email addresses
- Or – Use the QueryBuilder to build the query using a somewhat graphical interface to do so

SQL Query for Staff eMail List

```
SELECT  S.first_name_vc +
SUBSTRING(S.middle_name_vc, 1, 1) +
S.last_name_vc  AS [Staff Member Name],
SC.contact_info_vc  AS [Fallcreek eMail]
FROM  staff_t AS S
LEFT OUTER JOIN staff_contacts_t  AS SC
ON S.staff_id = SC.staff_in
WHERE  (SC.contact_type_in = 2)
```

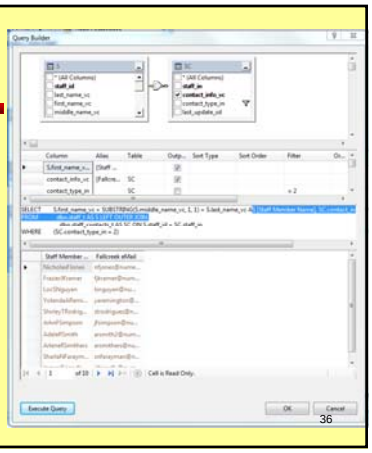
Execute the Query from Query Builder to test

Execute the Query in Query Builder to Test

10 rows returned.

Click OK.

Then Finish.



Examine Queries in TableAdapter Properties

- Look at staff_tTableAdapter properties
  - ConnectionString
  - InsertCommand
    - CommandText
- Look at staff_contacts_t1 (just created) properties
  - ConnectionString
  - InsertCommand
    - CommandText

Next step is to …

Build the application so that intellisense "knows" about the class

And then …

USE THE DAL

Go to Default.aspx

- Drag a Gridview data control from the toolbox onto the page
- Open the codebehind window for the page and select Page Events and the Load event to generate the Page_Load subroutine structure

Write Code Lines

Description	Code
Instantiate a runtime TableAdapter for the eMail list	Dim eMailListAdapter as New *(use intellisense here)*
Set the GridView's DataSource property to the result of the GetData method of the TableAdapter	GridView1.DataSource = eMailListAdapter.GetData
Bind the Gridview to the data	Gridview1.Databind()

40

Now it's time to …

SAVE ALL – BUILD ALL – TEST RUN

41

Staff Information

- A complicated query can be saved on the SQL Server as a view

```
CREATE VIEW
   name_of_view
AS
SELECT …
```

42

SQL Query for Staff Information

```
SELECT S.last_name_vc AS [Last Name], S.first_name_vc AS
  [First Name], S.middle_name_vc AS [Middle Name],
  S.office_location_ch AS [Office], SP.phone_number_ch AS [Phone],
  PT.description_vc as [Phone Type], SC.contact_info_vc AS
  [Other Contact], CT.type_description_vc AS
  [Other Type], S.driv_license_no_ch AS [DL]
FROM  dbo.staff_t S
FULL JOIN staff_phones_t SP ON S.staff_id = SP.staff_in
LEFT OUTER JOIN phone_type_t PT ON SP.phone_type_in =
PT.phone_type_id FULL JOIN staff_contacts_t SC
ON SC.staff_in = S.staff_id LEFT OUTER JOIN contact_type_t CT
ON SC.contact_type_in = CT.contact_type_id
```

Describe the result set in English

Create DAL Objects that Use a View

- vw_staffinfo
- You can SELECT everything from the view
- Use the objects you create by retrieving data into a GridView control on a page

Create and Use a Data Access Layer in a Web Application

By Susan L. Miertschin