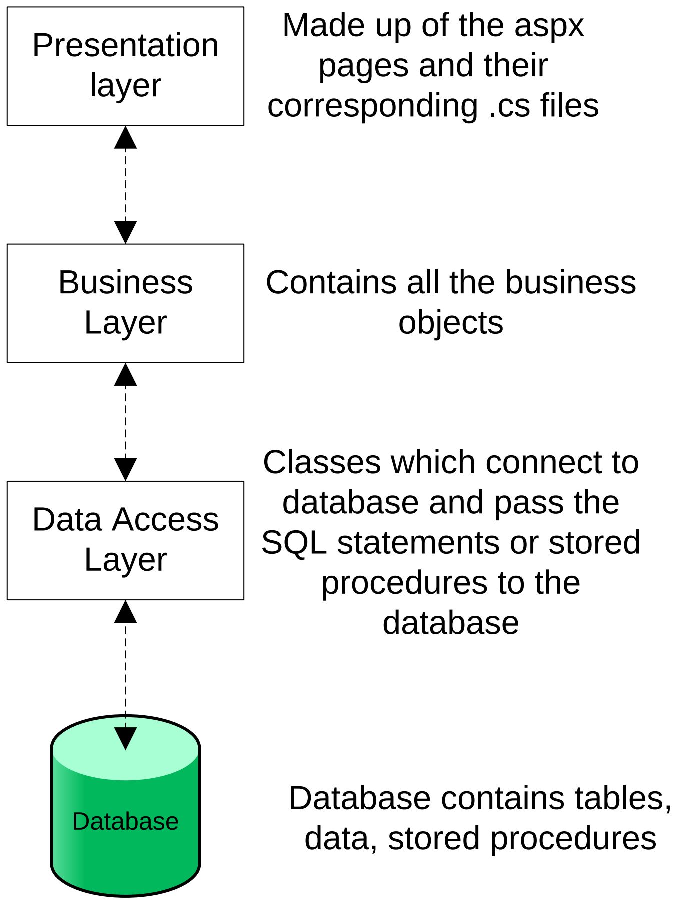
Data access layer part 2

Contents

Review of ADO.NET so far


Presentation layer
Made up of the aspx pages and their corresponding .cs files
Business Layer
Contains all the business objects
Data Access Layer
Classes which connect to database and pass the SQL statements or stored procedures to the database
Database
Database contains tables, data, stored procedures

ADO.NET is a Data access technology built into the .NET framework. ADO.NET provides a wide range of classes to work with databases. The System.data.sqlClient namespace contains the classes for connecting to Microsoft SQL server.

Data base connections

To be able to execute SQL statements you need to first open a connection to the database. This is achieved by the sqlConnection object.

Example

```
  SqlConnection objConnection = new SqlConnection(connectionString);
```

Connection strings

Contain the details needed to connect to database

- Name of database server
- Name of database
- Security access
- Connection timeout

It can (and should) be setup in the web.config file. As most of your application will use the same connection.

Web.config file:

```
<connectionStrings>
      <add name="ConnectionString" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\NORTHWND.MDF;Integrated
Security=True;User Instance=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

To read the connection string from your code you need to include the following code:

```
string strConn;
strConn = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;
```

TIP: The connection string can be generated for you automatically by using the SQLDataSource wizard.

Opening a connection

To connect to the database you need to use the connection object.

To open a connection you need to:

1. Set up connection string
2. Create connection object
3. Open connection using open() method
4. Every opened connection must eventually be closed by using the close() method

Example

```
//set up connection string
string strConnection;

strConnection =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;

//set up connection object
SqlConnection objConnection;
objConnection = new SqlConnection(strConnection);

//open connection
objConnection.Open();

//close connection
objConnection.Close();
```

Executing an SQL statement

The command object is used to execute SQL statements on the database. It needs to be configured so that it knows 2 things:

- The SQL statement (or stored procedure name) it needs to execute
- The database it needs to execute it at. The database is already supplied in the connection object.

Example:

```
//set up command object
```

```
 SqlCommand objCommand = new SqlCommand(strSQL, objConnection);
```

The above code sets up the command object so it is ready to excute but at this stage it has not yet executed.

The dataReader object

The dataReader object can be used to store data retrieved from database. It contains a Read() method used to open the dataReader for reading. The data reader is used for read only forward only data access.

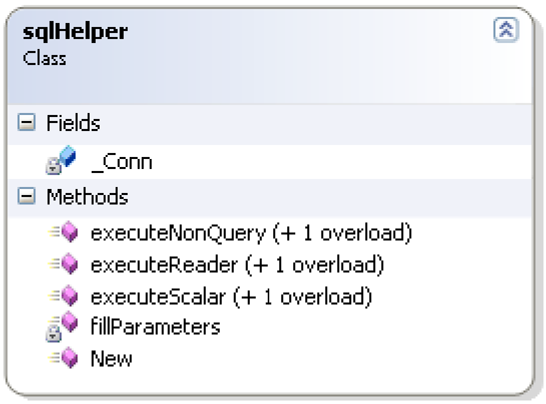
Example:

```
//set up date reader to hold data
SqlDataReader objDataReader;

//execute SQL and store in data reader
 objDataReader = objCommand.ExecuteReader();
```

Data access layer

We concluded our last look at ADO.NET with the creation of a simple data access class:


sqlHelper
Class
Fields
_Conn
Methods
executeNonQuery (+ 1 overload)
executeReader (+ 1 overload)
executeScalar (+ 1 overload)
fillParameters
New

The sqlHelper class contains a private variable called _Conn which holds the connection string.

It contains a constructor (New) which reads the connection string from the web.config file and stores it in the _Conn variable.

It also contains 3 public methods:

- executeNonQuery - executes an SQL insert, update or delete statememt (not a query)
- executeScalar - executes an SQL statement which returns a single value
- executeReader - executes an SQL statement which returns a table of data (many rows and/or columns)

All of these methods have been overloaded to take in an extra parameter. The extra parameter contains the parameters that need to be passed to the SQL statement. For example if we are setting up an SQL statement to retrieve all products belonging to a particular category then we need to pass the category ID to the executeReader method.

To populate the parameters passed in we have used a private method called fillParameters.

Activity 1. Use data access layer

sqlHelper.cs1 & dataAccessReviewed.aspx

Create the App_Code folder in your current ASP.NET application. Right click on the folder and select “Add existing item” navigate to where you last created this data access layer and include it in this current project.

Have a quick look through the code to familiarise yourself with it.

Create a web page and display the customer ID and contact name again. Use your data access layer.

Part 2:

There are a lot records lets add paging. Go to the source view for your page and add the allowPaging property and set it to true:

```
<asp:GridView ID="gvCust" runat="server" AllowPaging="true">
</asp:GridView>
```

Try and view the page you should get the following error:

“The data source does not support server-side data paging.”

This is because we used a data reader to store the data. A data reader is used for read only forward only data access. It cannot be used for paging as once the data has been read you cannot go backwards or forwards through the set of records. For the same reason you can't use it for sorting.

So if we want to have paging or sorting in our application we can't use the data reader and therefore we can't use this data access layer.

Dataset

There is another object called a dataset. This object stores a subset of the database.

It can hold several tables, stored as dataTables, at a time

Can be used for displaying, updating, inserting, deleting and navigation of data

How are datasets created

Datasets are created using:

```
DataSet objDataSet = new DataSet();
```

Once created they can be populated using the dataAdapter

Data Adapter

Contains 4 command objects

- Select command
- Update command
- Insert command
- Delete command

Uses the fill() method to fill the dataset with data

```
//set up data adapter
SqlDataAdapter objDataAdapter = new SqlDataAdapter(strSQL, objConnection);
//fill dataset
objDataAdapter.Fill(objDataSet);
```

Data binding

Can bind to all the controls a data reader can.

Activity 2. Display customers using a dataset

custUsingDataset.aspx

Create a web page with a gridview on it and name it gvCust.

Add the following code to the code behind file:

```
protected void Page_Load(object sender, EventArgs e)
{
        //set up connection string
        string strConnection;

        strConnection =
ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;

        //set up connection object
        SqlConnection objConnection;
        objConnection = new SqlConnection(strConnection);

        //open connection
        objConnection.Open();

        //set up SQL to be executed
        string strSQL;
        strSQL = "select customerID, contactName from customers";

        //set up data adapter
        SqlDataAdapter objDataAdapter = new SqlDataAdapter(strSQL, objConnection);

        //create a dataset
        DataSet objDataSet = new DataSet();
        //fill dataset
        objDataAdapter.Fill(objDataSet);

        //bind to gridview
        gvCust.DataSource = objDataSet;
        gvCust.DataBind();

}
```

Now try and add paging to the grid view. You will need to set the allowPaging property to true for the gridview and also add the following code in the code behind. To create this method select the lighting bolt for the gridview and double click on the pageIndexChanging event.

```
protected void gvCust_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
        //set up the page index
        gvCust.PageIndex = e.NewPageIndex;

        //bind gridview
        gvCust.DataBind();
}
```

Adding the dataset to the data access layer

So now we should change our data access layer to include the dataset. Why bother with the data reader at all since the dataset does what the data reader does plus more? The data reader is a lot more efficient than the dataset so when we just want to display data without paging or sorting we should use the data reader.

Activity 3.   Adding the dataset

sqlHelperV2.cs

Add the following 2 functions to your data access class

```
public DataSet runSQLDS(string strSQL)
{
        //create connection object get connection string from private
        //variable _Conn
        SqlConnection objConnection = new SqlConnection(_Conn);

        //create data adapter
        SqlDataAdapter objAdapter = new SqlDataAdapter(strSQL, objConnection);

        //create dataset
        DataSet objDataSet = new DataSet();

        //populate dataset with result of executing the sql
        objAdapter.Fill(objDataSet);

        //return dataset
        return objDataSet;
}

public DataSet runSQLDS(string strSQL, SqlParameter[] parameters)
{
        //create connection object get connection string from private
        //variable _Conn
        SqlConnection objConnection = new SqlConnection(_Conn);

        //create data adapter
        SqlDataAdapter objAdapter = new SqlDataAdapter(strSQL, objConnection);

        //fill parameters
        fillParameters(objAdapter.SelectCommand, parameters);

        //create dataset
        DataSet objDataSet = new DataSet();

        //populate dataset with result of executing the sql
        objAdapter.Fill(objDataSet);
        //return dataset
        return objDataSet;
}
```

Activity 4.   Test the methods

testDataSet.aspx

Whenever code is added it MUST be tested. Create a web page with a grid view on it named gvProducts. Place the following code in the code behind:

```
protected void Page_Load(object sender, EventArgs e)
{
        sqlHelper objDAL = new sqlHelper();
        string strSQL;

        strSQL = "select productName, unitPrice from products";

        gvProducts.DataSource = objDAL.runSQLDS(strSQL);
        gvProducts.DataBind();
}
```

The above code tests the runSQLDS method with no parameters.

testDataAdapterWithParams.aspx

Now create a web page that will test runSQLDS with parameters. Add a dropdown list and name it DDLCategory, add a button and a gridview. In the code behind include the following code:

```
using System.Data;
using System.Data.SqlClient;

public partial class testDataAdapterWithParams : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        //populate drop down list only the first time the page is loaded
        if (Page.IsPostBack == false)
        {
            //create SQL object
            sqlHelper objSqlHelper = new sqlHelper();

            //create SQL statement that needs to be passed
            string strSQL;
            strSQL = "select categoryID, categoryName from categories";

            //populate drop down list
            ddlCategory.DataSource = objSqlHelper.executeSQL(strSQL);
            ddlCategory.DataTextField = "categoryName";
            ddlCategory.DataValueField = "categoryID";
            ddlCategory.DataBind();
        }

    }
    protected void btnView_Click(object sender, EventArgs e)
    {
        //create SQL object
        sqlHelper objSqlHelper = new sqlHelper();

        //create SQL statement that needs to be passed
        string strSQL;
        strSQL = "select productName, unitPrice from products where categoryID = 
@categoryID";

        //set up parameters
        SqlParameter[] objParams;
        objParams = new SqlParameter[1];
        objParams[0] = new SqlParameter("@categoryID", DbType.Int16);
        objParams[0].Value = int.Parse(ddlCategory.SelectedValue);

        //populate gridview
```

```
            gvProducts.DataSource = objSqlHelper.runSQLDS(strSQL, objParams);
            gvProducts.DataBind();
        }
    }
```

What about stored procedures

Our data access layer only handles SQL statements it does not handle stored procedures. To change your data access layer to handle stored procedures you need to change the commandType property to storedProcedure

⬂ syntax

```
objCommand.CommandType = CommandType.StoredProcedure
```

⬂ Example

```
public SqlDataReader executeSP(string strSQL)
{
   //create connection object get connection string from private
   //variable _Conn
   SqlConnection objConnection = new SqlConnection(_Conn);

   //create command object
   //get SQL to execute from the strSql parameter
   //passed as input to this function
   SqlCommand objCommand = new SqlCommand(strSQL, objConnection);
   objCommand.CommandType = CommandType.StoredProcedure;

   //open database connection
   objConnection.Open();

   //execute SQL and return dataReader
   return objCommand.ExecuteReader(CommandBehavior.CloseConnection);
}

public DataSet runSpDs(string strSQL, SqlParameter[] parameters)
{
   //create connection object get connection string from private
   //variable _Conn
   SqlConnection objConnection = new SqlConnection(_Conn);

   //create data adapter
   SqlDataAdapter objAdapter = new SqlDataAdapter(strSQL, objConnection);

   objAdapter.SelectCommand.CommandType = CommandType.StoredProcedure;

   //fill parameters
   fillParameters(objAdapter.SelectCommand, parameters);

   //create dataset
   DataSet objDataSet = new DataSet();

   //populate dataset with result of executing the sql
   objAdapter.Fill(objDataSet);

   //return dataset
   return objDataSet;

}
```