Connecting your ASP.NET website to your database

A website is much more useful when it can support business operations and save data from the transactions. The data can be later compiled and displayed in reports and dashboard visualizations. The website is feeding data to the databases keeping them and the reports/dashboards connected to them up to date. A database is also more useful when it is connected to program users via the website. The website is useful in that it provides access, the code inside the website also controls what website users can do to the database. The web site protects the database by controlling what data is accepted. DBA’s do not give public access to their databases, so a website is used to protect the integrity of the information of the database, by implementing error-checking.

So how does the website connect to the database? This document attempts to explain this. We are using Visual Studio.NET’s ASP.NET functionality to create and host webpages and the website. Visual Studio also includes functionality to enable the webpage and database to interract. That set of functionality is called ADO.NET, or Active Data Objects. You are enouraged to read the provided ADO.NET glossary to better understand this document. Each of the concept below leverage the ADO.NET functionality inside Visual Studio.NET in effect connecting the ASP.NET webpages to the SQL Server database.

In this course we do not make exciting looking webpages, that is another course. Rather we work on web-database interactivity making a transaction processing system that performs data management and provide the datasource for PowerBI reports, charts and maps.

***Connecting webpage to database:*** Each webpage needs to include a SQLconnectionstring. This string of values tell the webpage which cloud-based database (DBMS) server to connect to, which database account to access, and the userID and password that is used to connect to the database account. Each data management operation discussed below runs over the specified database connection.

***Loading lists and tables at webpage start-up***: In your webpage you want to display data from the database when the program starts up. You want to see lists pre-filled with items (radiobutton lists, dropdown lists), and you want to see gridviews that display tables of items such as product lists. You would use the DataAdapter.Fill() command to retrieve data into datatables (which are arrays). Each dataAdapter has a SQL SELECT statement that specifies which database table to retrieve values from. The dataAdapter also must specify which database connection to use.

***Retrieving filtered lists of data:*** You can retrieve filtered lists of data depending on list selections made by the program user. If you make the SQL SELECT statement inside the dataAdapter have a WHERE statement that uses a parameter (@p1), then the SELECT statement can retrieve a filtered list of values into a datatable (array). For example you can retrieve a list of sales for one customer selected. The filtered list can be retreived because the SELECT statement use a variable which is called a parameter. You have to pass the filter value from the webpage into the SQL SELECT statement, then run the Data.Adapter.Fill() operation (which is run over the specified connection)

***Retrieving single values from a database***: Sometimes you ned to retrieve one value from a database whether it is a calculated value or just raw value. The easiest way is to use a SQLCommand with a SELECT statement that specifies which colunm from which row to retrieve. You use the SQLCommand’s .executescalar function to run the SQL SELECT statement and you assign the results to a variable.

***Saving new rows of transaction data***: to simplify the saving new rows of data to a database table we use the same approach as in earlier modules. We create a new row in an array, load its columns with values from the webpage controls, add the row to the array then use a dataAdapter.update() to save the new row of data in the array back to the database. This works because the array has the same column schema as the database table, which is made possible by running a dataAdapter.fillschema(array) command at page\_init to tell each array that will be used for saving rows of data, its schema (its column names, their datatypes, default values, and foreign key restrictions).

***Updating data in the database:*** Two ways are provided for updating data in the SQL Server database. In each case values are taken from the webpage and copied back to the database. To understand this section you must understand that just as in previous modules when we used two types of arrays 1) arrays to store rows of transaction data and 2) arrays that store categorical information (called dimensional tables) and summary running totals. The example used was individual payroll records of labor worked for a contruction company being stored in a transaction table of payroll records and then the summary table of payroll totals by labor category being updated.

a) Updating values using code behind the scenes. After saving a row of transaction data, values in other tables need to be updated (for example after a sale the inventory of each product needs to be decreased). We write SQL UPDATE SET code to update each table that needs to be updated. This UPDATE SET code is placed inside a SQLcommand and is run without any human interaction

b) Manual update of a row of data – it is common to have to manually update rows of data in dimension tables, for example updating the mailing address of a customer. Here you would use a paramtereized dataAdapter.Fill() to bring values from the database into textboxes on the screen. You need a second transaction to push the data back to the database, we can easily do this with a dataAdapter.Update() operation.

As a final concept we connect PowerBI reports and dashboards to the SQL database. So we use the ASP.NET website to load transaction data into the SQL database, and to perform updates and deletes. We display filtered lists, transaction records, and summary data in the website for transaction convenience. The data is securely stored in the SQL database. We use PowerBI to connect to the database. The reports, charts and maps made in PowerBI use the data stored in the database. The PowerBI charts can be stored in the cloud, distributed in a .pbix file, or embedded in a ASP.NET webpage. So the course is made ‘full-circle’ due to the ASP.NET – SQL Server interraction made possible by the ADO.NET objects discussed here. Not pretty but epic functionality, that can be learned and leveraged to solve business problems.

Your professor has carefully created learning materials that leverage ASP.NET webpages, VB.NET/ADO.NET code, SQL Server databases, and PowerBI. That’s an nice packaging of technology to provide useful functionality.