SYSTEMS DESIGN / CAPSTONE PROJECT
MIS 413

User Guide 4.3

Accessing Stored Procedures via Code (not Wizards)

There are times when the developer needs additional control of data transfers between the stored procedures and objects (i.e. GridView). Learning to access stored procedures via manually coding the connection and command lines will assist you.

Example A: Demonstrates the use of code to INSERT a new record into a valid table

This example assumes you have built a form containing text boxes that contain the new values and a submit button. The following code is normally placed in the _submit_Click subroutine.

1. Build and TEST your INSERT stored procedure

2. Build your INSERT web form

3. Once you have error checked your input, the following code will:
    a. Build a connection string to the database
    b. Define the name of the INSERT Stored Procedure
    c. Declare all the input parameters (the @'s) and give them a value from the text boxes or drop down boxes etc.
    d. Open the database
    e. Execute the Stored Procedure
    f. Notify the user if successful
    g. Or handle errors

```
//note below the [mis413...] should be the name of your connection string, open your
webconfig file and find the connection string name
        string dbConn =
System.Configuration.ConfigurationManager.ConnectionStrings["mis413ConnectionString"].
ConnectionString;

        //build a connection to the database
        SqlConnection conn = new SqlConnection(dbConn);

        //use the above connection to execute a particular stored procedure [substitue
your personInsert stored procedure name below]
                using (SqlCommand cmd = new SqlCommand("[PersonInsert]", conn))
        {
            cmd.CommandType = CommandType.StoredProcedure;

        //build the parameters (input items) that the stored procedures needs
            cmd.Parameters.AddWithValue("@firstname", this._firstName.Text);
            cmd.Parameters.AddWithValue("@lastname", this._lastName.Text);
            cmd.Parameters.AddWithValue("email", this._email.Text);
            cmd.Parameters.AddWithValue("roleID", this._role.SelectedValue);
        // add more parameters as needed by your stored procedure
        // open the database and actually run the stored procedure, also catch any
errors and display them in your _message label
            try
            {
                conn.Open();
                cmd.ExecuteNonQuery();
                this._message.Text = "Person was inserted";

            }
                //if there are any errors with the store procedure, display them in
the message label
            catch (SqlException ex)
            {
                this._message.Text = "Error on inserting new person " + ex.Message;
            }
        }
```

Hints: Other commands commonly employed to Execute Stored Procedures

There are three typical CMD.Execute….. commands, each has a special purpose

executeNonQuery	Generally used for Deletes, Updates, Inserts if NO Values are expected to be returned
executeReader	Generally used for Select statements, where you desire to return one to many rows of data
executeScaler	Used by Select Statements, where you desire only one value (i.e. the new Primary Key value generated) returned

Finally, the use of **datasets** to return values from Stored Procedures is very common and is discussed in another User Guide.

Example B: A Sample Login Subroutine, using a DataReader Object to retrieve data about the user.

This example assumes you have text boxes that request the user's email and user's password (via textboxes) found on the standard ASP.Net Account/Login Page. The following code is normally placed in the LogIn subroutine.

1. Build and TEST a select Stored Procedure that returns the ID, FN, LN of the user. Special: this Stored Procedure has a WHERE clause that will only return the values if the Email and Pwd input parameters match those in the database.

2. Set up the page to enable manual coding for SQL Items (imports the proper DLL's) Above the "Partial Class" command (after your comments before the partial class)

```
using System.Data.SqlClient;
using System.Data;
using System.Web.Security
```

3. Build the code to open the db and check the email and password entered as follows:

```
//build a link the the name/pwd/user for your particular database
```

```
//note below the [mis413...] should be the name of your connection string, open your
//webconfig file and find the connection string name
        string dbConn =
System.Configuration.ConfigurationManager.ConnectionStrings["mis413ConnectionString"].
ConnectionString;

        //build a connection to the database
        SqlConnection conn = new SqlConnection(dbConn);

        //use the above connection to execute a particular stored procedure
//substitute your specific stored procedure name here]
        using (SqlCommand cmd = new SqlCommand("[tnjPersonLogin]", conn))
        {
            cmd.CommandType = CommandType.StoredProcedure;
        //build the parameters (input items) that the stored procedures needs
            cmd.Parameters.AddWithValue("@email", this._email.Text);
            cmd.Parameters.AddWithValue("@pwd", this._password.Text);
             // open the database and actually run the stored procedure, also catch any
      //errors and display them in your _message label
            try
            {
                conn.Open();
                SqlDataReader dtrReader = cmd.ExecuteReader();
                // see if any data was returned from the stored procedure, if no data
//than not a valid user or password
                if (dtrReader.HasRows)
                {
                    // read the first record
                    dtrReader.Read();

        //specialized code goes here, what should you do if the login is valid?
        //for example do you want to set a cookie
      // if you would like to check if all of this works you might want to test to here
//by inserting the following code and the deleting once you confirm it works
                      _message.Text = "Email/Password Good.";

                }
                else
                {
                    //no data was returned from the stored procedure, update a message
//label
                      _message.Text = "Invalid email or password.";
                }

            }
                //if there are any errors with the store procedure, display them in
//the FailureText Literal
            catch (SqlException ex)
            {
                      _message.Text = "Error on sign in procedure " + ex.Message;
            }
        }
```