Update Data Asynchronously using SqlCommand Object

Source	http://www.dotnetspider.com/kb/Article3003.aspx

Introduction

SqlCommand Object provides means of executing SQL statements and transactions asynchronously. The methods BeginExecuteNonQuery and EndExecuteNonQuery in the command object initiates and terminates the asynchronous execution of the Transact-SQL statement or stored procedure respectively. Given a callback procedure and state information of an object, you can provide more flexible UI interactions while performing lengthy background database operations.

The BeginExecuteNonQuery Method takes the following syntax in C#.

SqlCommand.BeginExecuteNonQuery Method (AsyncCallback, Object)

The C# syntax of the EndExecuteNonQuery Method is:

public int EndExecuteNonQuery (IAsyncResult asyncResult)

A call to BeginExecuteNonQuery to execute a SQL statement must have a matching call to EndExecuteNonQuery in order to complete the operation. If the process of executing the command has not yet finished, EndExecuteNonQuery blocks until the operation is complete.

You can verify that the command has completed its operation by using the IAsyncResult instance returned by the BeginExecuteNonQuery method. The AsyncState property of the IAsyncResult contains the information about the asynchronous operation. You may also check the completion status of an asynchronous operation using the IsCompleted property.

The following example shows you how to perform an asynchronous data upate on a table amed 'advt'.

```
         private void Savebutton_Click(object sender, EventArgs e)
        {

            SqlConnection conn = new SqlConnection("Data
Source=localhost;Integrated Security=SSPI;Initial Catalog=Bala;Asynchronous
Processing=true");
            SqlCommand comm;
            // To emulate a long-running update for all rows in a table
            conn.Open();
            comm = new SqlCommand("update advt set width=20 where advtid >
1030", conn);
            AsyncCallback  acb = new AsyncCallback(doUpdate);
            comm.BeginExecuteNonQuery(acb, comm);
// Display the Status in a label when you initiate the update process.
            label1.Text = "Data update in progress. Please wait!";
```

```
            doSomeWork();
        }
```

You cannot interact with the main user interface, in this case, form and its controls from a different thread that is started by BeginExecuteNonQuery method. Hence the callback procedure is all but guaranteed to be running from a different thread than the form. To display the result status in the label, you need to invoke a delegate method passing the required value, the no of rows affected by the update process.

```
         delegate void CrossThread(int nos);
        void doUpdate(IAsyncResult c1)
        {
            SqlCommand comm1 = (SqlCommand)c1.AsyncState;
            int rows = comm.EndExecuteNonQuery(c1);
            CrossThread ct = new CrossThread(FinishWork);
            this.Invoke(ct, rows });

         }
```

This method sets the status of the aysnchronous operation in the text property of the label.

```
         void FinishWork(int rows)
        {
            label1.Text = rows.ToString() + " rows update completed!";
        }
```

And, finally the method doSomeWork performs some process during the asynchronous update operation.

```
         void doSomeWork()
        {
          // some process
        }
```

Summary

System.Data.SqlClient.SqlCommand class has asynchronous methods to perform ExecuteReader and ExecuteXmlReader operations also.

~~~ End of Article ~~~