Read data asynchronously as XML using SqlCommand Object

Source	http://www.dotnetspider.com/kb/Article3005.aspx

Introduction

System.Data.SqlClient.SqlCommand object has two reader methods namely ExecuteReader and ExecuteXmlReader. While ExecuteReader is mainly used for retreiving sets of rows of results returned from queries and storing in a SqlDataReader object, ExecuteXmlReader creates a System.Xml.XmlReader. The results returned from your query is parsed into Xml format, and, an XmlReader object is attached when the first row is retrieved.

A typical ExecuteXmlReader query can be formatted as

SELECT * FROM tablename FOR XML AUTO, XMLDATA

This type of query only works with Microsoft SQL Server 2000 or later.

Asynchronous Reading

```
The BeginExecuteXmlReader method initiates asynchronous execution of a SQL
statement or stored procedure that returns rows as XML, so that other tasks
can run concurrently while the statement is executing. You must call the
EndExecuteXmlReader method to finish the operation and retrieve the requested
XML data.

private void button3_Click(object sender, EventArgs e)
{

        SqlConnection conn = new SqlConnection("Data
Source=localhost;Integrated Security=SSPI;Initial Catalog=Bala;Asynchronous
Processing=true");
        conn.Open();
        comm = new SqlCommand("select * from advt FOR XML AUTO, XMLDATA",
conn);
        AsyncCallback acb = new AsyncCallback(doSomeWork);
        comm.BeginExecuteXmlReader(acb, comm);
        label1.Text = "Data Retreival in progress. Please wait!";
}
```

Unless until you call EndExecuteXmlReader, the results from the query are blocked. You can verify that the command has completed its operation by using the IAsyncResult instance

returned by the BeginExecuteXmlReader method. IAsyncResult represents the status of the operation at any point of time so that you can check if the operation is completed or not. Once you get the results in the XmlReader, you can read the elements one by one until you reach the end of it.

Also, notice that because the process(doSomeWork) happens in a separate thread, you cannot gain access to main thread. So, at the end of the process, you need to invoke a delegate method to access the textbox in the main UI thread to display the completion status.

```
delegate void CrossThread(string s);
void doSomeWork(IAsyncResult c1)
{
    SqlCommand comm1 = (SqlCommand)c1.AsyncState;
    System.Xml.XmlReader xr = comm.EndExecuteXmlReader(c1);
    xr.Read();
    string data = "";
    while (!xr.EOF)
    {
        data += xr.ReadOuterXml() + "\r";
        xr.Read();
    }
    CrossThread ct = new CrossThread(FinishWork);
    this.Invoke(ct, data);
    xr.Close();

 }
void FinishWork(string data)
{
    textBox1.Text = data;
    label1.Text =  "Rows retreived from table displayed as XML";
}
```

Summary

Asynchronous operations on databases enables you to provide multi-tasking capabilities in your application.

***~~~ End of Article ~~~***