Advanced Programming Language (630501)
Fall 2011/2012 – Lecture Notes # 23

Retrieving Real-Time Data

- **Connected Data Base Access (Main Steps)**
- **Retrieving Data Using a DataReader**
- **Closing the DataReader**
- **Connection Strings.**

Connected Data Base Access (Main Steps)

1) Import the required namespaces:
   - ***System.Data***
   - ***System.Data.SqlClient*** namespaces (for ***SQL Server***).
2) Declare a ***SqlConnection*** object call it ***Conn (for example)***.
3) Declare a ***SqlCommand*** object call it ***Comm***.
4) Declare a ***sqlDataReader*** object call it ***dr***.
5) Declare a ***string***, call it ***strCon***. Initialize the ***strCon*** string to the connection string for the database.
6) Declare a ***string***, call it ***strSQL***. Initialize the ***strSQL*** string to the ***SQL SELECT*** statement:
7) Instantiate the ***Conn*** using the ***strSQL*** string.
8) ***Open*** the connection.
9) Instantiate the ***Comm*** object using the ***strSQL*** string and the ***Conn*** connection.
10) Create the ***dr*** by calling the ***ExecuteReader()*** method of the ***Comm*** object (The data population commands will go here).
11) Create a ***while loop*** that evaluates the ***dr.Read()*** method to get the next row of data and access each row (Data Processing).
12) ***Close*** all the objects.

Retrieving Data Using a DataReader

- You can use the ADO.NET ***DataReader*** to retrieve a read-only, forward-only stream of data from a database. Results are returned as the query executes, and are stored in the network buffer on the client until you request them using the ***Read*** method of the DataReader.
- Using the ***DataReader*** can increase application ***performance***.

- The DataReader provides an ***unbuffered*** stream of data that allows procedural logic to efficiently process results from a data source sequentially. The DataReader is a ***good choice*** when retrieving large amounts of data.
- Each .NET data provider included with the .NET Framework has a DataReader object:
  - ***OLE DB*** includes an ***`OleDbDataReader`*** object.
  - ***SQL*** Server includes a ***`SqlDataReader`*** object.
  - ***ODBC*** includes an ***`OdbcDataReader`*** object.
  - ***Oracle*** includes an ***`OracleDataReader`*** object.
- Retrieving data using a DataReader involves creating an ***instance*** of the ***`Command`*** object and then ***creating a DataReader*** by calling ***`Command.ExecuteReader`*** to retrieve rows from a data source. The following example illustrates using a DataReader where reader represents a valid DataReader and command represents a valid Command object.

```
reader = command.ExecuteReader();
```

- You can ***access*** each column of the returned row by passing the ***name*** or ***ordinal reference*** of the column to the DataReader. However, for best performance, the DataReader provides a series of methods that allow you to access column values in their native data types (***`GetDateTime, GetDouble, GetGuid, GetInt32,`*** and so on).

Closing the DataReader

- You should always call the ***`Close`*** method when you have finished using the DataReader object.
- Note that while a ***DataReader*** is open, the ***Connection*** is in ***use*** exclusively by that DataReader. You cannot execute any commands for the Connection; including creating another DataReader, until the original DataReader is closed.

Connection Strings

- In order to connect to a data source, you need to ***build a connection string*** that will define the context of the connection. The parameters of the connection string will differ somewhat depending on the data provider.
- The ***recommended procedure*** for creating a connection to a data source is as follows:
  - Create a ***global string*** to hold the connection string, and use the ***Application object*** to store the string.
  - Create the ***connection as a local object***, and explicitly destroy the connection when it is no longer needed.

- The .NET data providers give us the three connection classes:
  - ***SqlConnection*** (for use with SQL Server 7.0 or higher),
  - ***OleDbConnection*** (for use with data sources through OLE-DB providers),
  - ***OdbcConnection*** (for use with legacy ODBC drivers).

Example 1

- Connection to an **SQL Server 2000** data provider with the following parameters:

```
Server name is Hjalmar
Database name is Marvin
Security is set to Mixed Mode
Username is "sa"
Password is "42"
Timeout is 1 minute (60 seconds)
```

- The following code will make the connection.

```
// The following code segment defines the connection string
// We will use SQL Server .NET Data Provider
string strCn;
strCn = "User ID=sa;Password=42;Initial Catalog=Marvin;";
strCn = strCn +  "Data Source=Hjalmar;Connection TimeOut=60;";
// now we define the connection object and open the connection
SqlConnection sqlCn = new SqlConnection(strCn);
sqlCn.Open();
```

Closing the Connection

- After we are finished with a connection, we should close it down by calling the ***Close()*** method of the connection object, like this:

```
sqlCn.Close();
```

Example 2

- The next example will connect to the Access 2000 database ***c:\data\Marvin.mdb***
- Here's the code:

```
// The following code segment defines the connection string
// We will use an OLE DB .NET Data Provider
string strCn;
strCn = "Provider=Microsoft.Access;Initial Catalog=c:\\data\\Marvin.mdb;";
OleDbConnection oleCn = new OleDbConnection(strCn);
oleCn.Open();
```