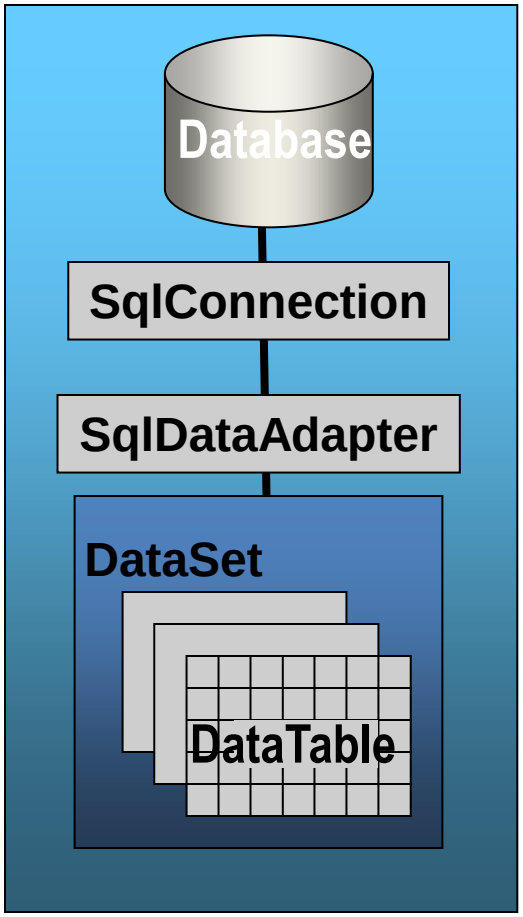
Introduction to Programming II

Lecture 10 – ADO II – DataSet, GridView And Stored Procedure

Semester II 2013

OVERVIEW



- **SqlConnection –** connects to the DB
- **SqlDataAdapter** - executes sql commands between the DS and DB
- **DataSet** - a cached DB in memory

DataSet class

- The ***DataSet*** class of .NET presents a set of data separated and distinct from any data stores.
- You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains - *non-datasource-specific* entity.

DataSet class (cont’)

- DataSet works with all models of data storage:
  - flat, relational, and hierarchical (since it does not have any 'knowledge' of the source of its data).
- To use DataSets the System.Data namespace should be imported

DataSet class (cont')

- An in-memory cache of data from a data source.
- Common way to represent and manipulate data
  - Universal data container
  - *Not just for use with databases*.
- Designed to be disconnected from the data source
  - Connect, execute query, disconnect.
- Can use XML
  - To read and write XML structured data
  - To read and write XMLSchema.

DataAdapter

- A ***DataAdapter*** is the object that connects to the database or file to fill the DataSet.
- Therefore it is a provider specific:
  - SqlDataAdapter,OleDbDataAdapter.
- **DataAdapter** object works as a bridge between the DataSet and the source of data.
- Used to ***fill*** DataSets and ***update*** the database.

DataSet Filling and Updating

- The cycle:
  - DataAdapter.Fill(DataSet)
  - Let user modify data in DataSet
  - DataAdapter.Update(DataSet)
- When you update, by calling an Update method , the DataAdapter takes (only) modified rows from the DataSet one by one:
  - If added, calls configured INSERT command
  - If modified, calls configured UPDATE command
  - If deleted, calls configured DELETE command

Fill

```
DataSet ds = new DataSet();
SqlDataAdapter adptr = new SqlDataAdapter(command, con);
adptr.Fill(ds,"t1"); //opens and closes the DB!
```

- This will fill up the DataSet with the requested query and will call it t1

The DataAdapter commands

- You can use DataAdapter constructor with a command and connection object.
- Or you can use DataAdapter commands.(to change the DB not the DS, but only on existing data)
- Contains four command objects
  - SelectCommand
  - InsertCommand
  - UpdateCommand
  - DeleteCommand
- SqlCommandBuilder(DataAdapter) – will build this command automatically. Will need a PK in the table

Update

```
SqlDataAdapter adptr = new SqlDataAdapter(cmd, con);
SqlCommandBuilder comb = new SqlCommandBuilder(adptr);
adptr.Update(t); //opens and closes the DB!
```

- This will update the relevant rows in the DB according to changes made on the DataSet\DataTable

Working with DB – summery

- To work with DB the following classes of ASP.NET may be used:
  - DataReader
    - Connected, non-cached, forward read-only items
  - DataSet
    - Disconnected, cached, scrollable data
  - DataAdapter
    - Logic for populating the DataSet and propagating changes back to the datasource

Parameters

- An insert query will look like this:

sql="Insert into Authors (au_id,au_lname,au_fname,phone, address,city,state,zip,contract) Values ***(@Au_id, @Au_lname, @Au_fname, @Phone, @Address, @City, @State, @Zip, @Contract) "***

- And parameters will be added to the command:

```
cmd.Parameters.Add(New SQLParameter("@Au_lname",
    frmLname.text)) ;

cmd.Parameters.Add(New SQLParameter("@Au_fname",
    frmFname.text)) ;

cmd.Parameters.Add(New SQLParameter("@Address",
    frmAddress.text)) ;
```

DataGridView

- The ***DataGridView*** control is the most complex and powerful of the controls used to display and maintain sets of data.
- It has near a 100 properties, methods and events.
- The DataGridView displays a tabular data (data organized into rows and columns) from the data source or renders a table containing the SQL data.

Data Source

- To specify the data source for the DataGridiew you have to perform the following operations:

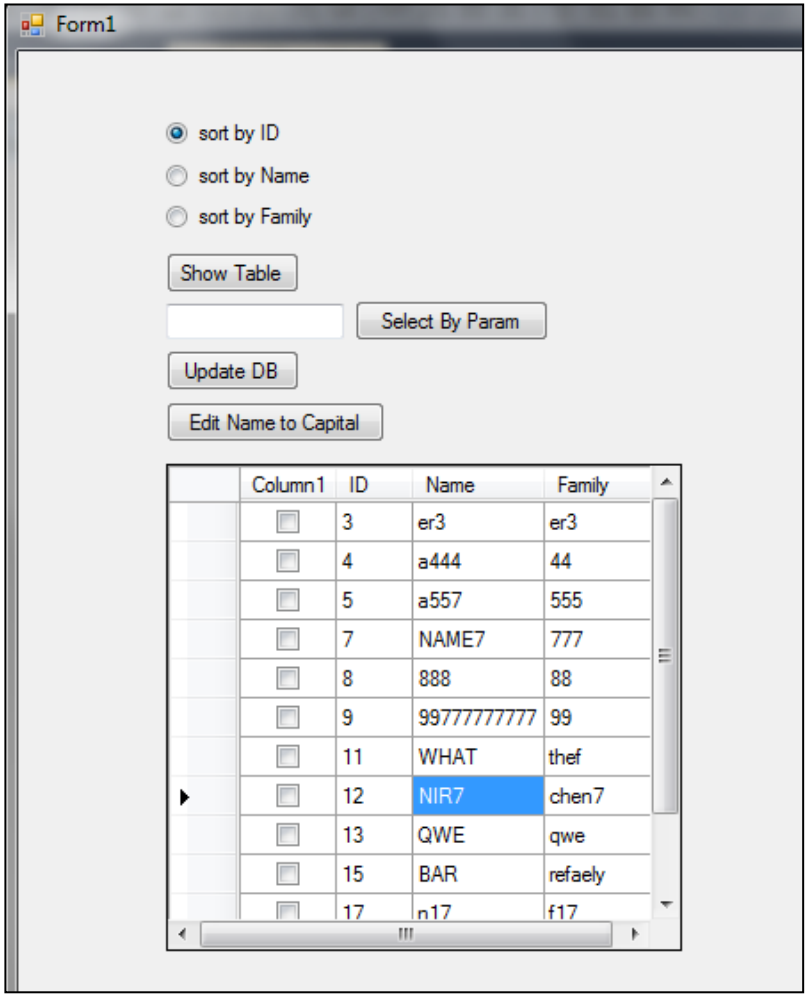
```
MyDataGrid.DataSource=ds.Tables["Authors"];
```

- Or an alternative syntax is to specify both a DataSource and a DataMember from the DataSet to be used:

```
MyDataGrid.DataSource=ds;
MyDataGrid.DataMember="Authors";
```

Example

- Build the example DS and DG together



Stored Procedure

- CREATE PROCEDURE ProcName @ParameterName ParameterType [= default] [OUTPUT],… AS

  SQL Statement

- @@ERROR – the error thrown in the sql -server

Stored Procedure - example

- CREATE PROCEDURE SearchUser @MyID int , @FamilyName varchar(20) output AS

  Select @FamilyName= Family

  From tbUser

  where ID=@MyID

  return @@ERROR

  GO

Stored Procedure – example2 returns a whole table

```
CREATE PROCEDURE [dbo].[SearchUserTable] @MyID
int AS

        Select *

        From tbUser

        where ID=@MyID
```

Calling Stored Procedure

- myCon.Open()
- SqlCommand MySPCommand = new SqlCommand("SearchUser", myCon)
- MySPCommand.CommandType = CommandType.StoredProcedure
- SqlParameter parName = new SqlParameter("@MyPar1", SqlDbType, [Size])
- parName.Value = Value (for input parameter)
- parName.Direction = ParameterDirection.Input/output/ReturnValue

Calling Stored Procedure cont’

```
MySPCommand.Parameters.Add(parName)
MySPCommand.ExecuteNonQuery()
myCon.Close()
parName.Value…
```

Example

- This example is for Stored Procedure and also for working with DB without separate DB-Layer