⇒ TimeSpan t = d - d1;
Response.Write(t.TotalDays.ToString());

considers time as well because both d & d1 are DateTime Obj.

t.Days.ToString() returns an integer indicating days only.

⇒ DateTime d = DateTime.UtcNow;
Calender1.TodaysDate = d;
DateTime d1 = DateTime.Now;
TimeSpan t = d - d1;
Response.Write(t.Hours.ToString() + ":" + t.Minutes.ToString() + ":" + t.Seconds.ToString())

⇒ e.Day.Date < d1 && e.Day.Date > d2.Day.Date.AddDays(1)

Static DateTime d1; ✓

↑ else does not remember when page comes back to server.

7/8/19 Database Connectivity
Project name → right click → Add new item → SQL Server Database

Extension = .mdf
Put in folder: App_Data

Right click on table → Add new table.

⇒ Table is reserved keyword in .NET

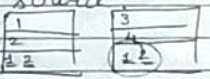
Generate Script option creates a text file having the SQL query which can be executed on SQL server.
Click 'Update Database' to create the table (executing the query) within IDE itself.

⇒ Toolbox → Data → GridView control
SqlDataSource
↳ click on '>' arrow & select 'configure data source'

By default, it has the database created locally (in App_Data) and the connection string automatically appears.

When the connection is stored or added to web config file, all the web pages in the particular project can access it. Any changes can be configured in web config file only.

⇒ Click '>' in GridView control & select the appropriate data source
Enable Paging :



Enable Sorting : sorts all data based on the column name

user clicks at runtime.

⇒ Advanced SQL Generation Options
- Generate update, delete query
- use optimized concurrency (for reading of foreign key & primary key constraints)

(AutoGenerate Edit Button - Lets user edit details of the database at runtime)

⇒ Query (edit or delete) is not available in the source (in view page source) as .NET is server side tech. & data is sent back to server each time

DetailsViewControl - Shows one record at a time.

Col1 value
Col2 value
Col3 value
1 2
↑ enable paging

Properties: AutoGenerate Edit
" Delete
" Insert
DataSourceID

→

Grid	Detail
Select record	Display only selected record.
DataSource1	DataSource2
Done to prevent data inconsistency	Queries GridView & then fetches the corresponding record

Here, once connection is established, no operation is performed on the actual database (when we try to sort the data in GridView, it fetches the data from database and runs. any changes are updated)

⇒ Query Builder to give customized SQL commands

FormView control - Shows one record at a time but, in the look & feel of a form.

⇒ GridView control does not provide insert functionality.

13/8/19 Programmatically performing database connectivity

```
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
```

protected void Button1_Click (object sender,
EventArgs e)

{ try {

SqlConnection con = new SqlConnection();
con.ConnectionString = " ";
↳ Replace " " with " " & add @ i.e @"

SqlCommand cmd = new SqlCommand();
cmd.CommandText = " <query>";

cmd.CommandType = CommandType.Text;
cmd.Connection = con;

SqlDataAdapter da = new SqlDataAdapter();
da.Fill
da.SelectCommand = cmd;
con.Open();
da.Fill(dt); // returns no. of rows & columns
con.Close();
DataTable dt = new DataTable();

GridView1.DataSource = dt;
GridView1.DataBind();

} catch (Exception e1)
{ }

}

⇒ DataSet is a collection of DataTables (useful while performing joins).

DataSet ds = new DataSet();
:
con.Open();
da.Fill(ds[0]);
con.Close();
gridview1.DataSource = ds[0];
gridview1.DataBind();

if not mentioned then overwritten

if a single table is taken ds[0] by default

Access Tables:
ds.Tables[<index>];

If Fill is called twice, it should create 2 different tables with separate index

⇒ using System.Configuration;
:
con ConnectionString = ConfigurationManager.ConnectionStrings["FacultyInfoStr"].ToString();

name of the data source

ds index [0,1,...] based on no of add element

advantage: Any changes in the connection string are made in web config file only which is reflected everywhere.

19/8/19 Ideally, when data is present in a table and the column name is attempted to change, it is not allowed. Hence, drop & recreate table

```
SqlConnection con;
SqlCommand cmd;
SqlDataAdapter da;
Button1_Click {

    con = new SqlConnection();
    con.ConnectionString = "_____";

    cmd = new SqlCommand();
    cmd.CommandText = "insert into StudentInfo
        (Id, Name, Department, Password) values (
    cmd.Connection = con;          '"+id+"', '"+name+"',
                                   '"+department+"',
                                   '"+TextBox4.Text+"')

    con.Open();
int i=cmd.ExecuteNonQuery(); con.Close();
    Response.Write("No. of rows affected "+i);
}
```

⇒
```
cmd.CommandText = "update StudentInfo
                   set Name='"+name+"',
                   Department='"+department+"',
                   Password='"+TextBox4.Text+"'
                   where id='"+id+"'";
```

⇒
```
cmd.CommandText = "delete from StudentInfo
                   where id='"+id+"'";
```

⇒
```
cmd.CommandText = "Select * from StudentInfo";
cmd.Connection = con;

da = new SqlDataAdapter();
da.SelectCommand = cmd;
```

DataTable dt = new DataTable();
con.Open();
da.Fill(dt);
con.Close();

DropDownList1.DataSource = dt;
DropDownList1.DataTextField = dt.Columns[1].ToString(); (Name)
DropDownList1.DataValueField = dt.Columns[0].ToString();
DropDownList1.DataBind();

⇒ If 'Select Name, Department from StudentInfo' is written the Name becomes the 0th column (resultant table) & Department becomes the 1st column

→ dt.Rows.Count returns the no. of records which have been retrieved by firing the Select query.

→ dt.Rows[0].ToString() does not work & gives ~~error that the~~ databinding error (DataRowView property not present)

Login Page

cmd.CommandText = "Select * from StudentInfo"

After con.Close():

```
for (int i=0; i< dt.Rows.Count; i++)
{
    if ((uname.Equals(dt.Rows[i].ItemArray[1].To
        && (pwd.Equals(dt.Rows[i].ItemArray[3].To
    {
        Response.Write("Logged In");
    }
    else
        Response.Write("Enter correct
                        username or password");
}
```

Parameterized Query (to prevent SQL Injection)

```
SqlParameter id1 = new SqlParameter();
id1.ParameterName = "@id";
id1.ParameterValue =
id1.Value = Convert.ToInt32(TextBox1.Text);
```

Similarly for name, department & password

```
cmd.CommandText = "insert into StudentInfo
                   (Id, Name, Department,
                   Password) value
                   (@id, @name, @departmen
                   @password)";

cmd.Parameters.Add(id);
cmd.Parameters.Add(name);
cmd.Parameters.Add(department);
cmd.Parameters.Add(password);
```

⇒ SqlDataReader dr;
con.Open();
dr = cmd.ExecuteReader();
~~con.Close();~~
while (dr.~~NextResult~~Read()) // ~~dr~~ dr.HasRows
{
  Response.Write(dr[0].ToString());
}
con.Close();

Classes:
i) Shared classes - can point to any database by using DbCommand.

ii) Disconnected classes - We can hold, manipulate and update the data from obj. of these classes even after closing the connection. Ex- DataTable

iii)

OleDB
using System.Data.OleDb

// In each command/property/class, replace Sql with OleDb

Reserved keywords.
insert into Students ([ID], [Name], [Password])
values ...

21/9/19

Session Management

Since HTTP is stateless preserving data across over multiple pages needs to be looked after the application. for this, state management is required.

→ State management is the process to store user specific application state of app

→ State can be saved either on client side or server side

(view state or hidden field) or cookies or control state or query string

By default, sessions are stored in the process memory

State can be maintained in: (on server)

- Appl^{n} obj
- Session obj
- Cache obj
- Database

Best among three is database but, as sessions/state are stored as obj.'s, they need to be serialized before storing it in the database which has an overhead cost.

Lifecycle

Web browser

↓ HTTP GET request

client has never visited site before

↓

IIS & ASP.NET appl^{n} respond by returning HTML rendered by mypage.aspx (page obj)

↓

Along with the page, a cookie with unique ID is sent to client

↓

this ID is used for any further requests from same client

Advantages

* fast
* shared state management among all user

Disadvantages

* State is stored once per server; in multiple server configuration

Database

Advantages

* can be accessed by any server in multi-server config

* need to pay serialization cost

Cache

* like appl^n state but, includes expiration via dependencies

* State stored once per server in multi-server config

→ for cache, data on page should be static (Ex - contact us page)

→ Cookie — simple
Hidden field — simple for page scoped data (doesn't work when navigated to other page)

<%@ ... Trace = "true" ... %>

Add this to see what other info it sent along with req. obj.

⇒ With every subsequent request, the server ID is checked & if it matches, a new updated one is sent along with the response (Matches ⇒ user has visited before)

⇒ LogOnUserIdentity.Name.ToString()
gives the username by which user is logged on onto the machine.

Many other options / info is available with request obj such as user lang. name, date, time, etc (Some might even allow to view the RAM info. on client's machine)

⇒ Http cookies generally have expiration time of 30 min to 60 min. After expiration, any request is considered as a new one.

* <sessions httpCookies = "true" />

⇒ Response.Redirect ("~/Default2.aspx");

HttpCookie c = new HttpCookie("name");
c.Value = "hello";
Response.Cookies.Add(c);

28/8/19

Add → New Item → Class (Student.cs)

```
public class Student          ← Table name
{
    private int id;
    private string name;
    private string dept;

    public Student() { }

    public Student (int id, string name,
                    string dept)
    {
        this.id = id;
        this.name = name;
        this.dept = dept;
    }
    // getter - setter methods
}

Button1_Click {
    Student s1 = new Student();
    s1.id = 1;
    s1.name = "Anand";
    s1.dept = "IT";

    Session["stdobje"] = s1;          // key-value
    Response.Redirect("~/Default2.aspx"); pair
}
```

Default2.aspx

```
Page_Load {
    Student s1 = (Student) Session["stdobje"];
```

Response.Write("" + s1.name);
}

In web config
<sessionState mode="StateServer" />

Started by default in 32-bit machine

When this is done, the object is stored in a separate process (other than the application process). Thus we need to serialize the object.

→ Sessions are created per browser (session id in cookie)
Once a browser session instance is cleared the session is lost.

⇒ Response.Redirect("~/Default3.aspx?"+"Name="+s1.name);

Default3.aspx
Page_Load{
Response.Write(Request.QueryString["Name"]);
}

for serializing:
[Serializable]
public class Student { ... }

→ Add appropriate namespace

```
Button1_Click {
    SqlConnection con = new SqlConnection();
    con.ConnectionString = "------";

    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = "Select * from Student" where id = "+I+";
    cmd.CommandType = CommandType.Text;

    DataTable dt = new DataTable();
    DataSet ds = new DataSet();

    SqlDataAdapter da = new SqlDataAdapter();
    da.SelectCommand = cmd;
    con.Open();
    da.Fill(dt);
    con.Close();

    Session["data"] = dt;
    Response.Redirect("~/Default3.aspx");
}
```

Default3.aspx
Add appropriate namespace.

```
Page_Load {
    DataTable dt = (DataTable) Session["data"];
    TextBox1.Text = dt.Columns[0].ToString();
    TextBox2.Text = dt.Columns[1].ToString();
    TextBox3.Text = dt.Columns[2].ToString();
}
```

displays column name

∴ Use:

dt.Rows[0][0].ToString();

While using DataSet:

dt.Tables[0].Rows[0][0].ToString();

⇒ Instead of passing / storing entire dataset or datatable in session object, we should pass / store object of student class.

⇒ Entity framework is used with .NET for object persistance (like Hibernate)