∴ 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)

9/19 Web Services.

New Website → Add → New Item → Web Service (Asmx) → Name: AdditionService.asmx

```
[WebService (Namespace = "——")]
[WebServiceBinding ...]

public class AdditionService : System.Web.Services.
                                WebService
{
    public AdditionService()
    { }

    [WebMethod]
    public int Add (int a, int b)
    {
        return (a+b);
    }
}
```

(TestService)

New Website → Right click → Select servi
reference → Provide address (Path of
WSDL of serv.)

In Testservice : Add new Web Page.
Using Addition Service Reference;

```
Button1_Click (...)
{
    AdditionService Reference.AdditionServiceSoapClient
client = new AdditionServiceSoapClient ();

    int i = client.Add(4,5);
    Response.Write(i);
}
```

⇒ In AdditionService:
Add SQL Database Server
Create Table [Person]
Id - int
Name - varchar

Add class (Person.cs)

```
public class Person
{
    private int id;
    private string name;

    public Person()
    { }
```

public int ID
{
set { id = value; }
get { return id; }
}
public String NAME
{
set { name = value; }
get { return name; }
}

- In AdditionService.asmx:
Add namespace: System.Data;
.Common;
.SqlClient;

[WebMethod] by default Serializable
public DataSet GetPerson()
{
DataSet ds = new DataSet();
SqlConnection con = new SqlConnection();
con.ConnectionString = "__";
SqlCommand cmd = new SqlCommand();
cmd.CommandText = "Select * from Person";
cmd.CommandType = CommandType.Text;
cmd.Connection = con;
SqlDataAdapter da = new SqlDataAdapter();
da.SelectCommand = cmd;

da.Fill(ds);

return ds;

}

⇒ In TestService

DataSet ds = client.GetPerson();
GridView1.DataSource = ds;
GridView1.DataBind();

⇒

Person p = new Person();
p.ID = (ds.Tables[0].Rows[0].ItemArray[0]);
Convert.ToInt32(
p.NAME = ds.Tables[0].Rows[0].ItemArray[0];
return p; // instead of ds.

// Do it in TestService

Person ds = client.GetPerson();
Response.Write(" "+ds.ID+" "+ds.NAME);

18/9/19 Caching

If data is static and we are sure that it won't change over a period of time, caching can be used.
Response time is decreased.
Pages rendered from cache memory on server rather than ASP.NET engine.

Page No.
Date

In Secs

In Source view:

<%@ OutputCache Duration = "5"
~~VerifyByParam~~ VaryByParam = "*" %>

VaryByParam - Different parameters in query string are cached.

Location attr. : Server // default
Client // not advisable

System.Web.Caching // namespace

protected void Create (object Sender,
EventArgs e)
{
Cache["address"] = "ddu"; } independent
Cache["city"] = "nadiad";
String[] keydep;
keydep = new string[2];
keydep[0] = "address";
keydep[1] = "city";
CacheDependency keydependency =
new CacheDependency(null, keydep);
Cache.Insert("phone", "123-456", keydependency);
mylabel1.Text = "Cache Full");
}
key value
dependent on address/city

protected void Display (...)
if (Cache["phone"] == null)
{
mylabel1.Text = "Cache empty";
}

```
else
{
    myLabel1.Text = "Address: " + Cache["address"]
        + "City: " + Cache["city"] +
        "Phone: " + Cache["phone"];
}
}

protected void Delete(...)
{
    Cache.Remove("address");
    myLabel1.Text = "Address removed";
}
```

File Dependency.

Add XML file → Add.xml

Add XML file → Dave.xml

Add.xml

```
<address>
  <add1>
    <nm> Anand </nm>
    <add> nad
    <pno> 123
  </add1>
  <add1>
    <nm> Dave
    <add> Nadiad
    <pno> 456
  </add1>
</address>
```

Dave.xml

```
<text> Data
</text>
```

Page No.
Date

Create
{
DataSet ds = new DataSet();
ds.ReadXml(@"_____"); → path of XML file: Add.xml

CacheDependency filedep = new
CacheDependency(@"_____"); ← Path of file: Data.xml

Cache.Insert("emp", ds, filedep); ← when this file is modified cache is cleared
mylabel1.Text = "Cache full";
}

Display
{
DataSet ds1 = new DataSet();
if (Cache["emp"] == null)
{
GridView → gv.DataSource = null
gv.DataBind();
mylabel1.Text = "cache empty";
}
else
{ ds1.ReadXml(@"_____"); ← Path of Add.xml
gv.DataSource = ds1;
gv.DataBind();
}
ds1 = (DataSet)Cache["emp"];
}

Time Based dependency (Sliding Window)

Load

```
{
    DataSet ds = new DataSet();
    ds.ReadXml(@"---");
    Cache.Insert("add", ds, null,
                 Cache.NoAbsoluteExpiration,
                 TimeSpan.FromSeconds(10));
    myLabel1.Text = "Cache Full";
}
```

(key, value, dependency)

↑ if page accessed within 10s, it is cached for next 10s

Page_Load

```
{
    Label1.Text = DateTime.Now.ToLongTimeString();
}
```

Display

```
{
    DataSet ds1 = new DataSet();
    if (Cache["emp"] == null)
    {
        gv.DataSource = null;
        gv.DataBind();
        myLabel1.Text = "Cache empty";
    }
    else {
        ds1 = (DataSet)Cache["emp"];
        gv.DataSource = ds1;
        gv.DataBind();
    }
}
```

⇒ Cache.Insert("add", ds, null,
DateTime.Now.AddMinutes(1),
TimeSpan.Zero);

clears cache after 1 min

Nowadays, due to asyn. comm., caching is ~~use~~ not used.

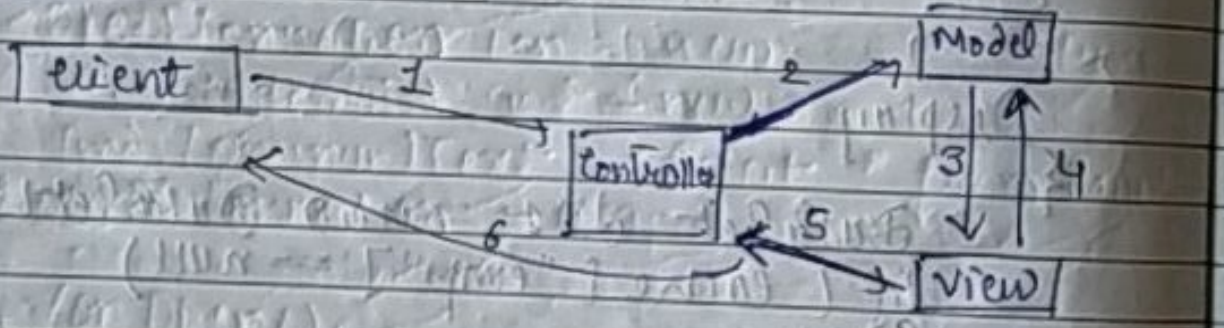
Cached obj. are stored in mem. allocated to your appln.

If Output caching used, the entire page is cached.

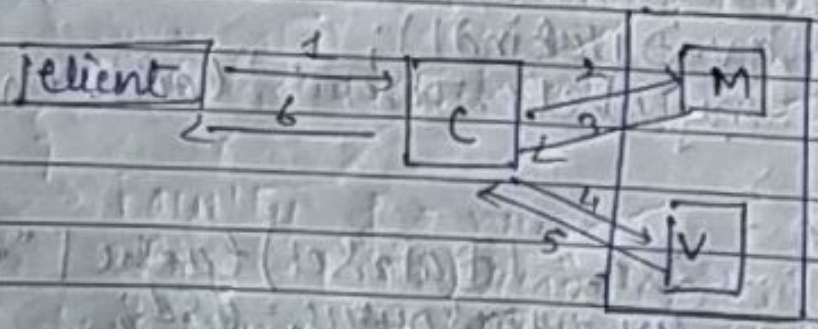
Cache mem. size is lower than RAM. A larger size would mean more no. of page faults and more response time which in turn degrades performance. Hence, page should not be cached if it isn't absolutely necessary.

19/9/19

Active Node:



Passive Node:



⇒ File → New → Project (MVC)

Controllers:

Home Controller

ASP.NET 5 RC → used to build Beta-version

View Engine: a) aspx b) cshtml (Razor) (4.0+ versions)

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
    public ActionResult About()
    {
        ViewBag.Message = "Your app. des. page";
        return View();
    }
    public ActionResult Mypage()
    {
        return View();
    }
    :
}
```

Right click → Add View

Mypage.cshtml

```
@{
    ViewBag.Title = "Mypage";
}
<h1> Welcome </h1>
```

_Layout.cshtml

```
<li>@Html.ActionLink("HDivPage", "Mypage", "Home")</li>
```

Page name → "HDivPage"; Action name → "Mypage"; Controller name → "Home"

⇒ Copy Access DB into App_Data

Model → New Class

```
public class Faculty
{
    public int ID { get; set; }
    public string Name { get; set; }
    public string Dept { get; set; }
}
```

Model → New class

```
public class FacultyDataContext
{
    public static IEnumerable<Faculty>
                    GetFaculties()
    {
        IList<Faculty> Faculties = new List<Faculty>();
        string constr = "Provider=Microsoft.Jet.OLEDB.4.0
                    + @"------";
        OleDbConnection con = new (constr);
        OleDbCommand cmd = new
        cmd.CommandText = "Select * from Faculty";
        cmd.Connection = con;

        con.Open();
        OleDbDataReader reader = cmd.ExecuteReader();

        while (reader.Read())
        {
            Faculty f = new
            f.ID = (int) reader["ID"];
            f.Name = (string) reader["Name"];
            Faculties.Add(f);
        }
        con.Close();
```

Linq query ≈ h2k query
Entity Framework ≈ Hibernate ORM

Page No.
Date :

return Faculties;
}
}

→ Controller → New
public class DataController : Controller
{
public ActionResult GetFaculty()
{ ViewData["Faculty"] = FacultyData.Context.GetFaculties;
return View();
}
}

Add View (without Model)

In View
@using WebApplication8.Models;
@using System.Web;
Collection;
Data;
System;

@{
ViewBag.Title = "Index";
}

<h2> Index </h2>
<h2> List of faculties Are: </h2>

<P>
@Html.BeginForm("GetFacultyData", "Index", "Test")
@{
IEnumerable<Faculty> faculties =
ViewData["Faculty"] as IList<Faculty>;

```
if (faculties != null)
{
}
<ul>
@foreach (Faculty f in faculties)
{
<li>
<a href="/Test/@Html.Encode(
f.ID)">
@Html.Encode(f.Name)
</a>
</li>
}
</ul>
} </p>
```

⇒ Add in _Layout.cshtml

23/9/19 **Calculator**

Add New Controller

```
public ActionResult Calculator()
{
    ViewData["Method"] = "Get";
    return View();
}
```

Right click → Add View (without model)

★ →

In .cshtml: (View)

```
<html>
<body>
<div>
@using (Html.BeginForm("Calculator",
"Calculator"))
```

```
@using System.Web;
@using System.Web.Mvc.Html;
```

Action name

Controller name

```
<br/>
<input type="text" name="lbl1"
       value="Enter Number1" />
<input type="text" name="num1"
       value="@Html.Encode(ViewData
       ["num1"])" />
// Similarly add textbox for num2 &
   result; and add a button for
   addition
}
```

* →

```
[HttpPost]
public ActionResult Calculator(String num1,
{                              String num2,
    ViewData["method"] = "Post";  String add)
    return View();
}
```

name property

⇒ Add link in Layout file.

```
— *(2) — (... , String res)
    int Num1 = Convert.ToInt32(num1);
    int Num2 = Convert.ToInt32(num2);
    int result = Num1 + Num2;
    ViewData["res"] = result;
    res = Convert.ToString(result);
    ViewData["num1"] = Num1;
    ViewData["num2"] = Num2;
```

Acts as container

To Store in DB:

In Controller:

After res = ~~convert T;~~ result.ToString();

String constr = "_____";

OleDbConnection con = new [constr]

OleDbCommand cmd = new [Need to pass]

cmd.CommandText = "insert into [Add] values ('" + Num1 + "', '" + Num2 + "', '" + result + "')";

cmd.Connection = con;

con.Open();

int i = cmd.ExecuteNonQuery();

con.Close();

ViewData["method"] = i.ToString();

return View();

⇒ We can put the DB connectivity code in a class file as well.

- Constructor compulsory
- Name of class different than controller