asp.net

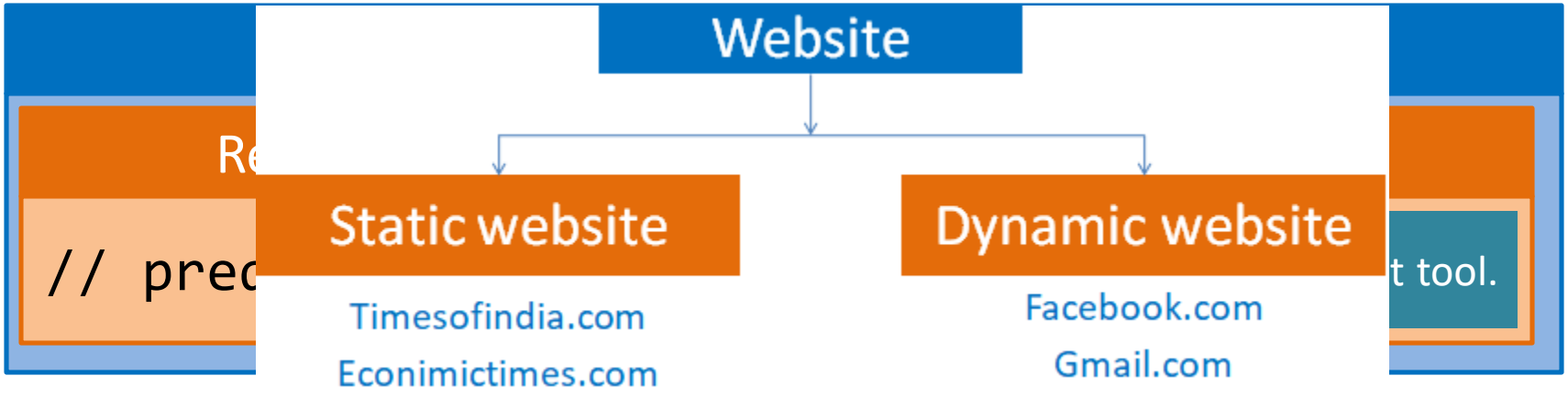
Palle Technologies

V1.0

What is **ASP. Net** ?

- ASP. Net is a open source **server side** web application **framework**.
- ASP. Net is used for create Dynamic WebPages.

A framework contains



Files supported in ASP. Net

_____.aspx	_____.ascx	_____.master	web.config
<%@Page … %> <html> … </html>	<%@Control..%> html tags css js c# asp.net tags	<%@Master … %> <html> … </html>	<configuration> … </configuration>
_____.aspx.cs	**_____.ascx.cs**	**____.master.cs**	**Global.asax**
{ … }	{ … }	{ … }	<%@ Application … %>

- **.aspx**, **.ascx**, **.master** files are called as **Markup files**.
- **.aspx.cs**, **.ascs.cs**, **.master.cs** files are called as **Code files**.

code supported in markup files

_____.aspx

html tags + html controls
javascript
asp.net controls code
C# Code

_____.master

html tags + html controls
javascript
asp.net controls code
C# Code

_____.ascx

html tags + html controls
javascript
asp.net controls code
C# Code

create ASP. Net website

- Open **Visual Studio**
- File > New > Website
- In New website Dialog > select **ASP. Net Empty Website** > select folder to save the new project > and click on OK.

- In the solution explorer window -> right click on **project name** -> click on Add New Item
  (shortcut: **ctrl + shift + A**).

- In Add new item dialog > select **Web Form** > Enter the **file name** (do not delete or modify .aspx exension) -> click on Add.
- Now ____**.aspx** and ____**.aspx.cs** files are created.

- To run > right click on project name > click on **View in Browser.**

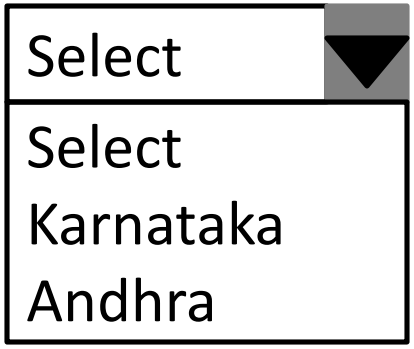
ASP. Net Controls

- ASP. Net tags are Case-sensitive
- During rendering time all ASP. Net controls are converted into HTML tags.

```
<asp:TextBox ID="tbName" runat="server">
</asp:TextBox>
```


```
<asp:RadioButton ID="rdMale" runat="server"
 Text="Male"></asp:RadioButton>
```


```
<asp:CheckBox ID="chkBg" runat="server"
Text="O+ve"></asp:CheckBox>
```




```
<asp:DropDownList ID="ddlState" runat="server">
<asp:ListItem>Select</asp:ListItem>
<asp:ListItem>Karnataka</asp:ListItem>
<asp:ListItem>Andhra</asp:ListItem>
</asp:DropDownList>
```


```
<asp:Button ID="btnSubmit" runat="server"
 Text="Submit"></asp:Button>
```

Conversion of ASP. Net Controls to HTML Controls

```
<asp:TextBox ID="tbName"
runat="server">
</asp:TextBox>
```

```
<input type="text" id="tbName" />
```

```
<asp:RadioButton ID="rdMale"
runat="server" Text="Male">
</asp:RadioButton>
```

```
<input type="radio" id="rdMale" />
<label for="rdMale">Male<label>
```

```
<asp:CheckBox ID="chkBg"
runat="server" Text="O+ve">
</asp:CheckBox>
```

```
<input type="checkbox" id="chkBg" />
<label for="chkBg">O+ve<label>
```

```
<asp:DropDownList ID="ddlState"
runat="server">
<asp:ListItem>Select</asp:ListItem>
<asp:ListItem>Karnataka</asp:ListItem>
<asp:ListItem>Andhra</asp:ListItem>
</asp:DropDownList>
```

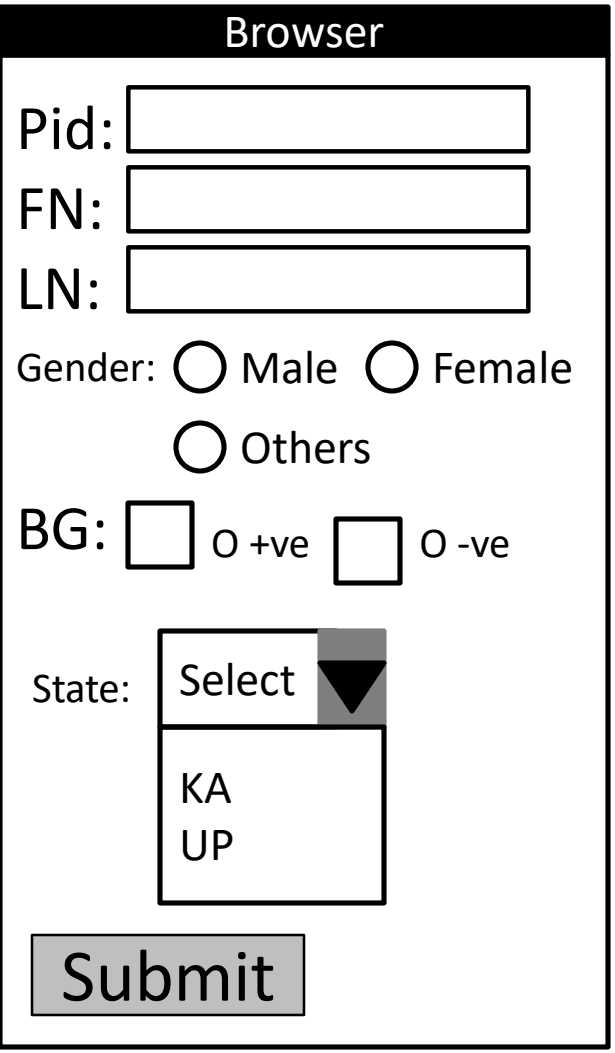
```
<select id="ddlState">
<option>Select</option>
<option>Karnataka</option>
<option>Andhra</option>
</select>
```

```
<asp:Button ID="btnSubmit"
runat="server" Text="Submit">
</asp:Button>
```

```
<input type="submit" id="btnSubmit"
value="Submit" />
```

ASP. Net Controls Sample

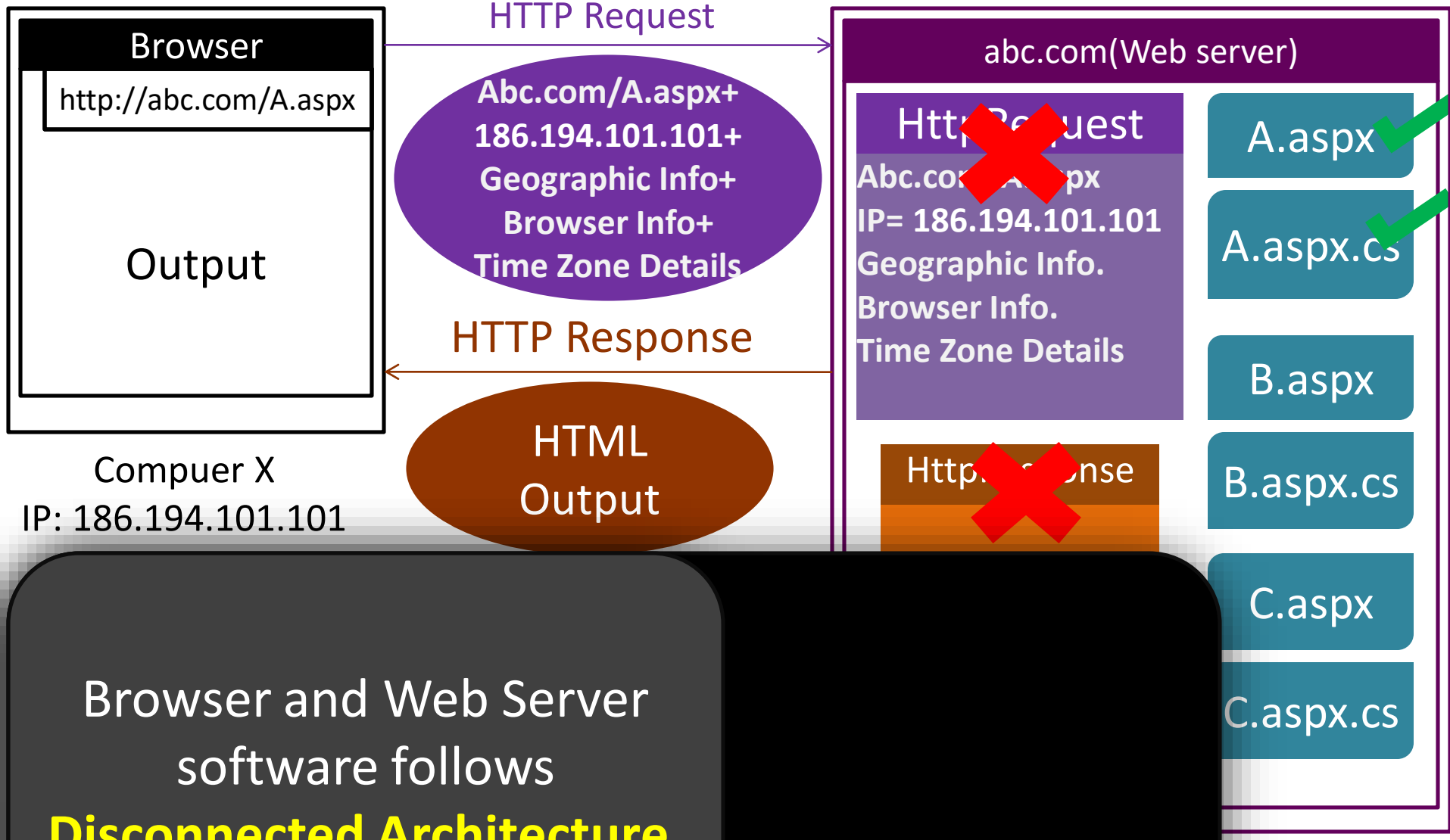
Browser



A.aspx

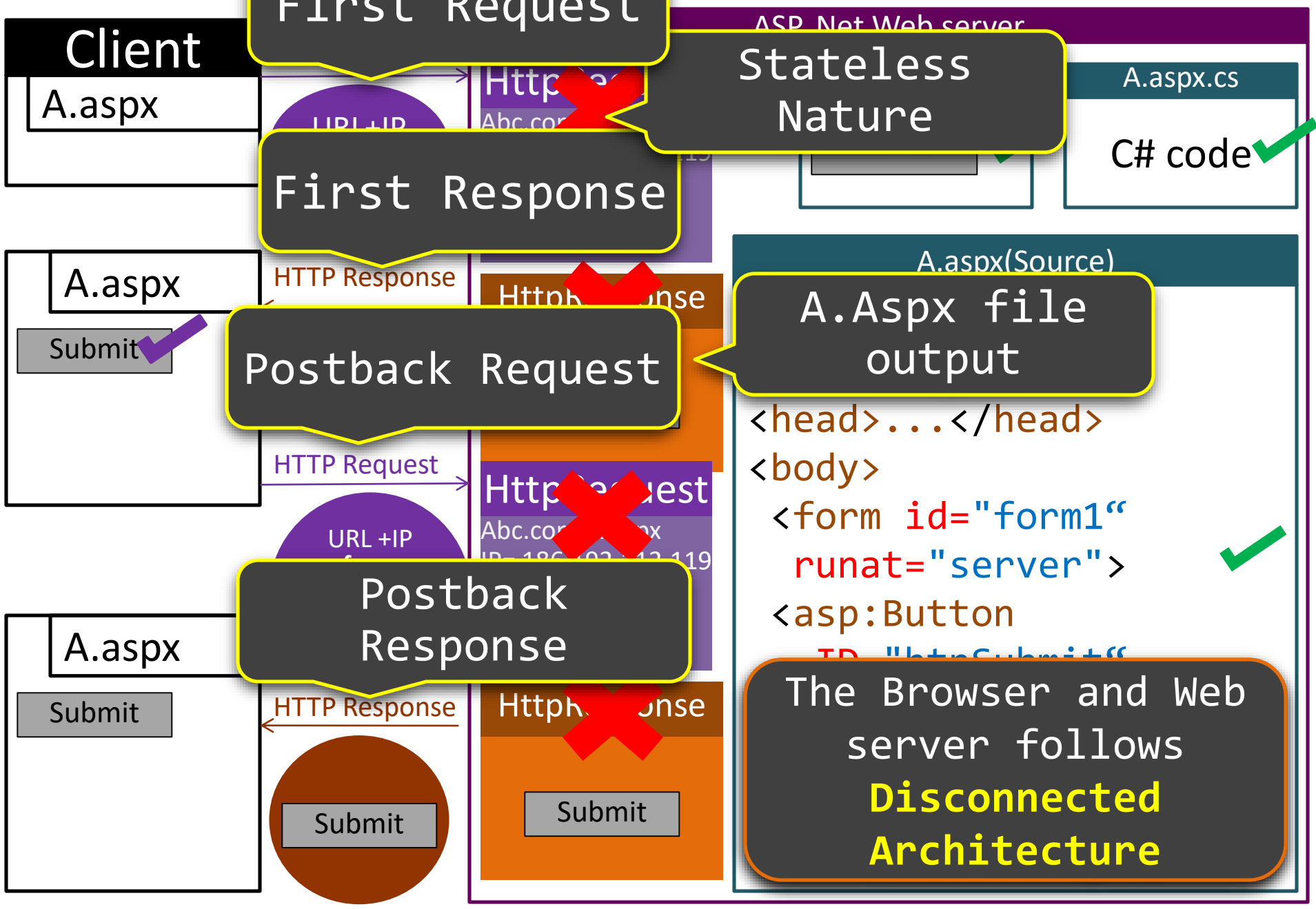
```
<form ID="form1" runat="server">
Pid:<asp:TextBox ID="tbPid" runat="server">
</asp:TextBox><br />
FN:<asp:TextBox ID="tbFn" runat="server">
</asp:TextBox><br />
LN:<asp:TextBox ID="tbLn" runat="server">
</asp:TextBox><br />
Gender:<asp:RadioButton ID="rdMale" runat="server"
 Text="Male"></asp:RadioButton>
<asp:RadioButton ID="rdFemale" runat="server"
 Text="Female"></asp:RadioButton>
<asp:RadioButton ID="rdOthers" runat="server"
 Text="Others"></asp:RadioButton><br />
BG:<asp:CheckBox ID="chkOpve" runat="server"
Text="O+ve"></asp:CheckBox><asp:CheckBox ID="chkOnve"
 runat="server" Text="O-ve"></asp:CheckBox><br />
State:<asp:DropDownList ID="ddlState"
 runat="server">
<asp:ListItem>Select</asp:ListItem>
  <asp:ListItem>KA</asp:ListItem>
  <asp:ListItem>UP</asp:ListItem>
</asp:DropDownList><br />
<asp:Button ID="btnSubmit" ID="Submit"
  runat="server"></asp:Button>
</form>
```

Client Server Communication

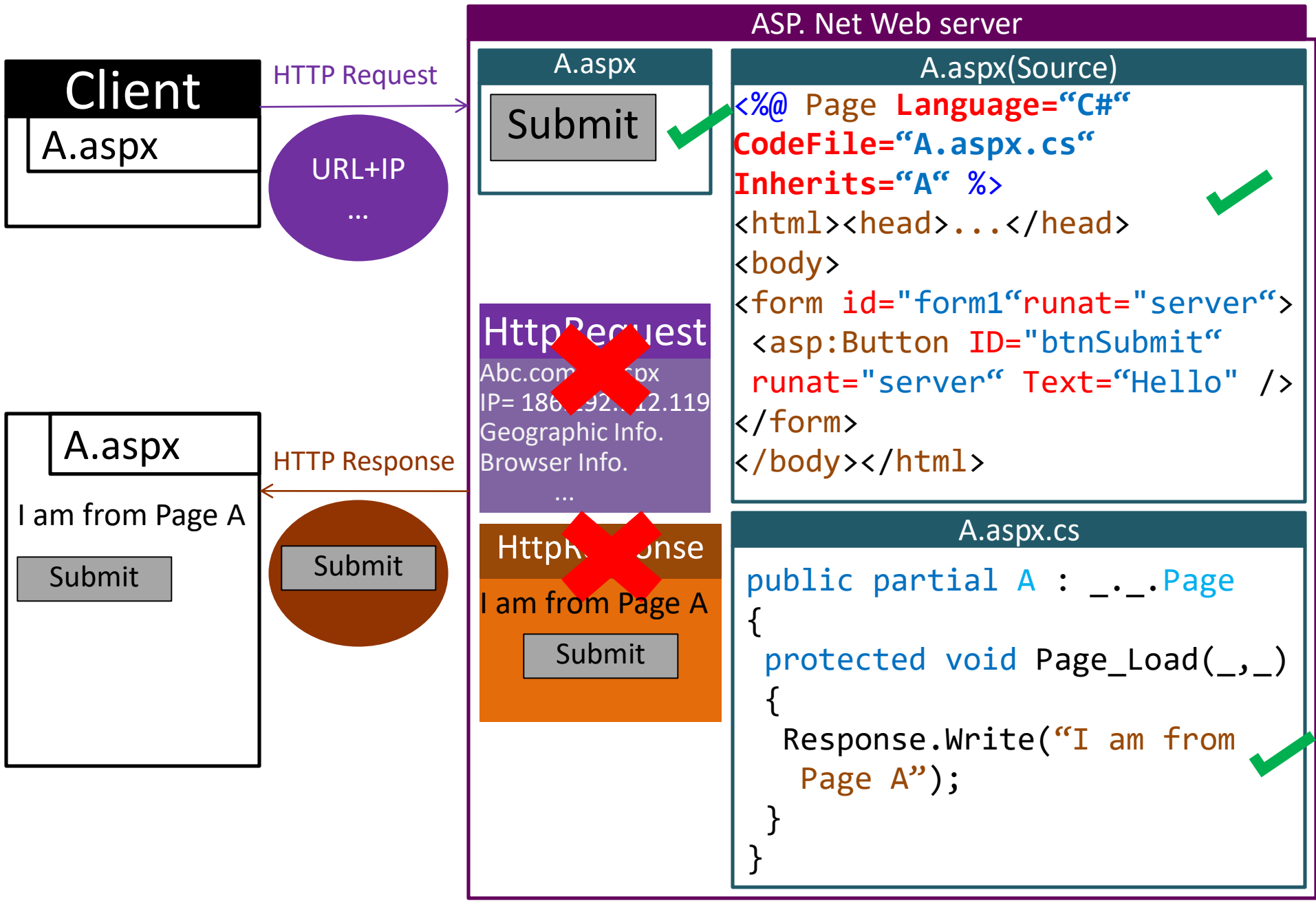


Browser and Web Server software follows **Disconnected Architecture**.

terminology



Relation between .aspx and .aspx.cs files



imp points

- first request is a request sent to the server from client for the first time | when the client currently not holding the output of the asked page.
- post back request is a request sent to the same page (usually this will happen when the user clicks on button)

Response.Write

- Response.Write method is used for adding output directly to the HttpResponse object.
- Response.Write content will be always added before the actual html

runat= “server”

- When we define any control inside form tag with **runat=server**,that control will be visible to the corresponding **.aspx.cs** file

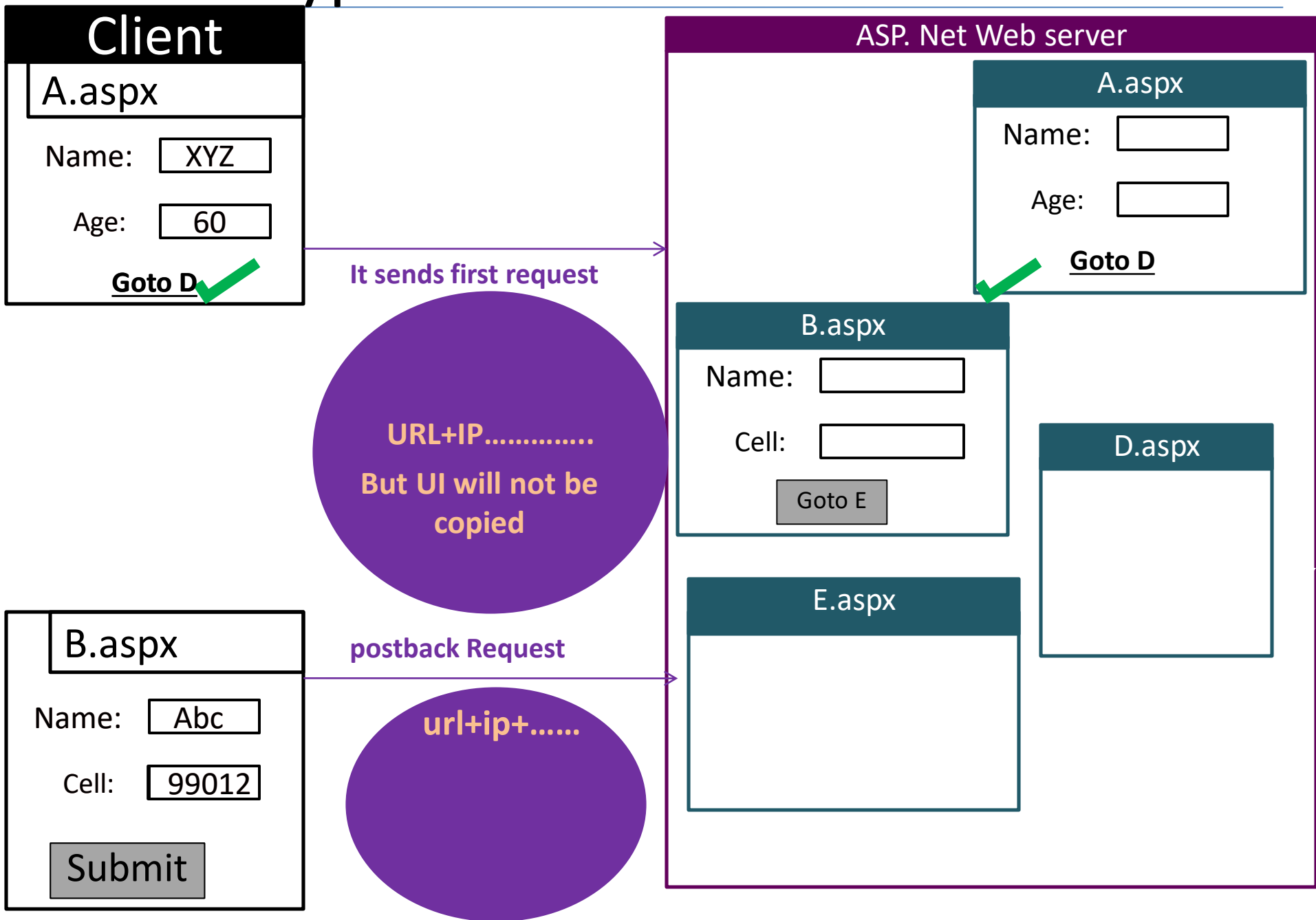
A.aspx(Source)

```
<%@ Page ... %>
<html><head>...</head>
<body>
<form id="form1“ runat="server“>
<asp:TextBox ID=“tb1” runat="server">
</asp:TextBox> br />

<input type=“text” ID=“tb2” runat="server"
</asp:TextBox> br />

<asp:Button ID="btn1“ Text=“Add"/>
</form></body></html>
```

A.aspx.cs

```
p p A : _._.Page
{
 p v Page_Load(_,_)
 {
    form1.
    tb1.
    tb2.
    btn1.

 }
}
```

Hyperlink control vs Button



When To use which Controls

Name: 

When we cant predict the value

Gender: Male

When we have 2 or 3 options and want to select only one

Courses: .Net C

When we have multiple options and want to select multiple

States: Select

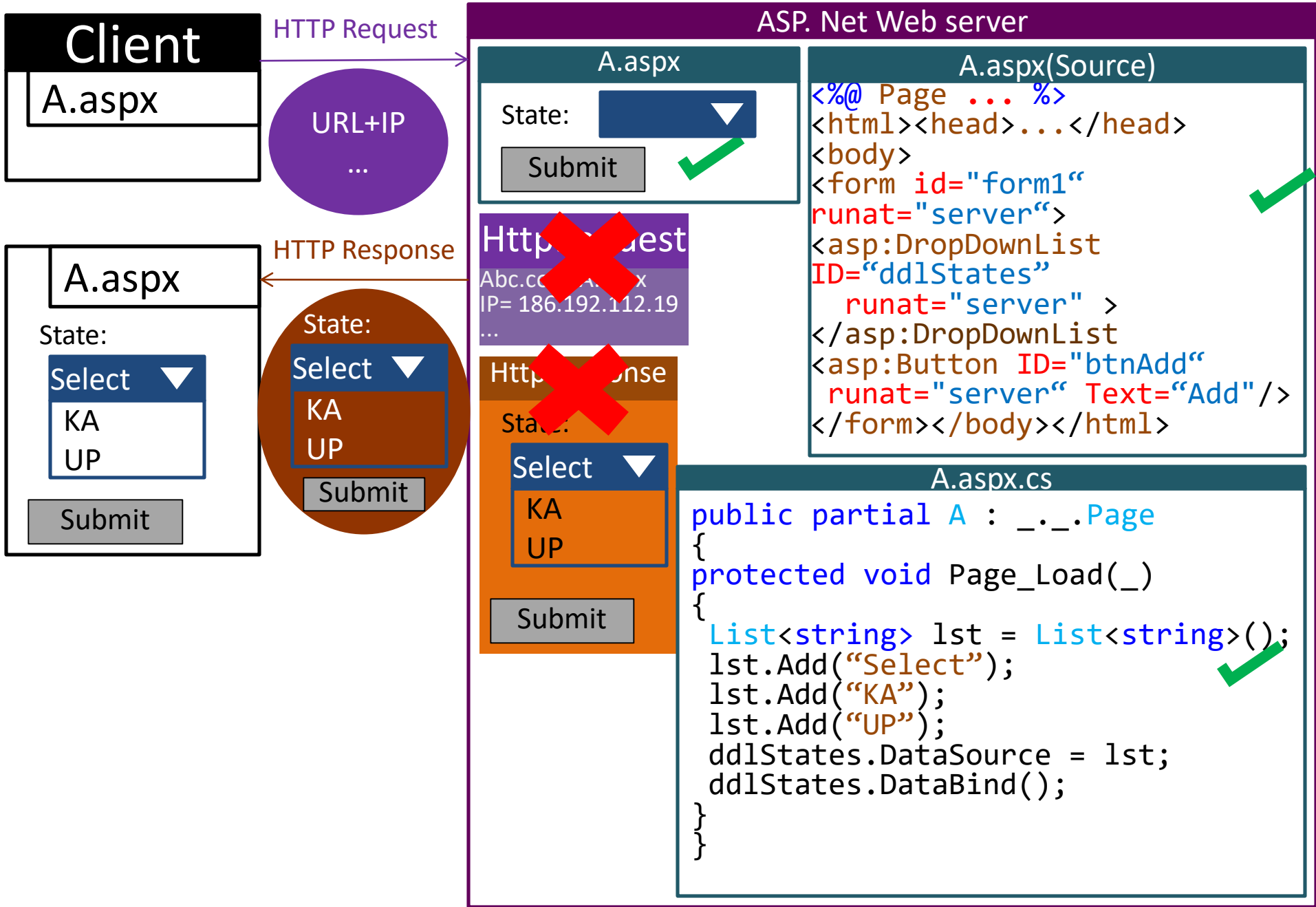
Select
Karnataka
Kerala
Ap

When we have more than 3 options and want to select only one

IsPostBack AutoPostBack

- **IsPostBack** is a property present in Asp.net using which we can identify whether the incoming request is the first request or postback request
- **IsPostBack** is a property defined in Page.class
- During first request **IsPostBack** value will be false.
- During second request **IsPostBack** value will be true.
- Note :- use controlId.Text property for reading the controls data.
- By using **AutoPostBack** property we can give **post back ability** to any control.

Programmatically filling data into DropDownList



ASP.Net Control Classes

- During Compilation time all **ASP.Net control tags** **are converted to equivalent C# Class Object.** (or)
- All Control tags which are having **runat="server"** **converted to equivalent C# Class Object.**

ASP.Net Control	C# Class
`<asp:TextBox ID="tbName" runat="server" />`	TextBox
`<asp:Button ID="btnSubmit" runat="server" />`	Button
`<asp:DropDownList ID="ddlState" runat="server"> </asp:DropDownList>`	DropDownList
`<asp:RadioButton ID="rdMale" runat="server" />`	RadioButton
`<asp:CheckBox ID="chkBg" runat="server" />`	CheckBox
`<asp:ListBox ID="ddlState" runat="server"> </asp:ListBox>`	ListBox
`<asp:Label ID="lblFor" runat="server"></asp:Label>`	Label
`<input type="text" id="lblFor" runat="server"/>`	HtmlInputText
`<form id="form1" runat="server"/>`	HtmlForm

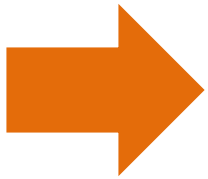
ASP.Net Page compilation cycle

A.aspx

Submit

A.aspx(Source)

```
<%@ Page Language="C#"%>
<html><head>...</head>
<body>
<form
 id="form1"
 runat="server">
 <asp:Button
  ID="btnSubmit"
  runat="server"
  Text="Submit"/>
</form>
</body>
</html>
```



ASP. Net Compiler

A.aspx.cs

```
public partial A : _._.Page
{
 protected HtmlForm form1 = new HtmlForm();
 protected Button btnSubmit = new Button();
 protected void Page_Load(_,_)
 {
 }
}
```

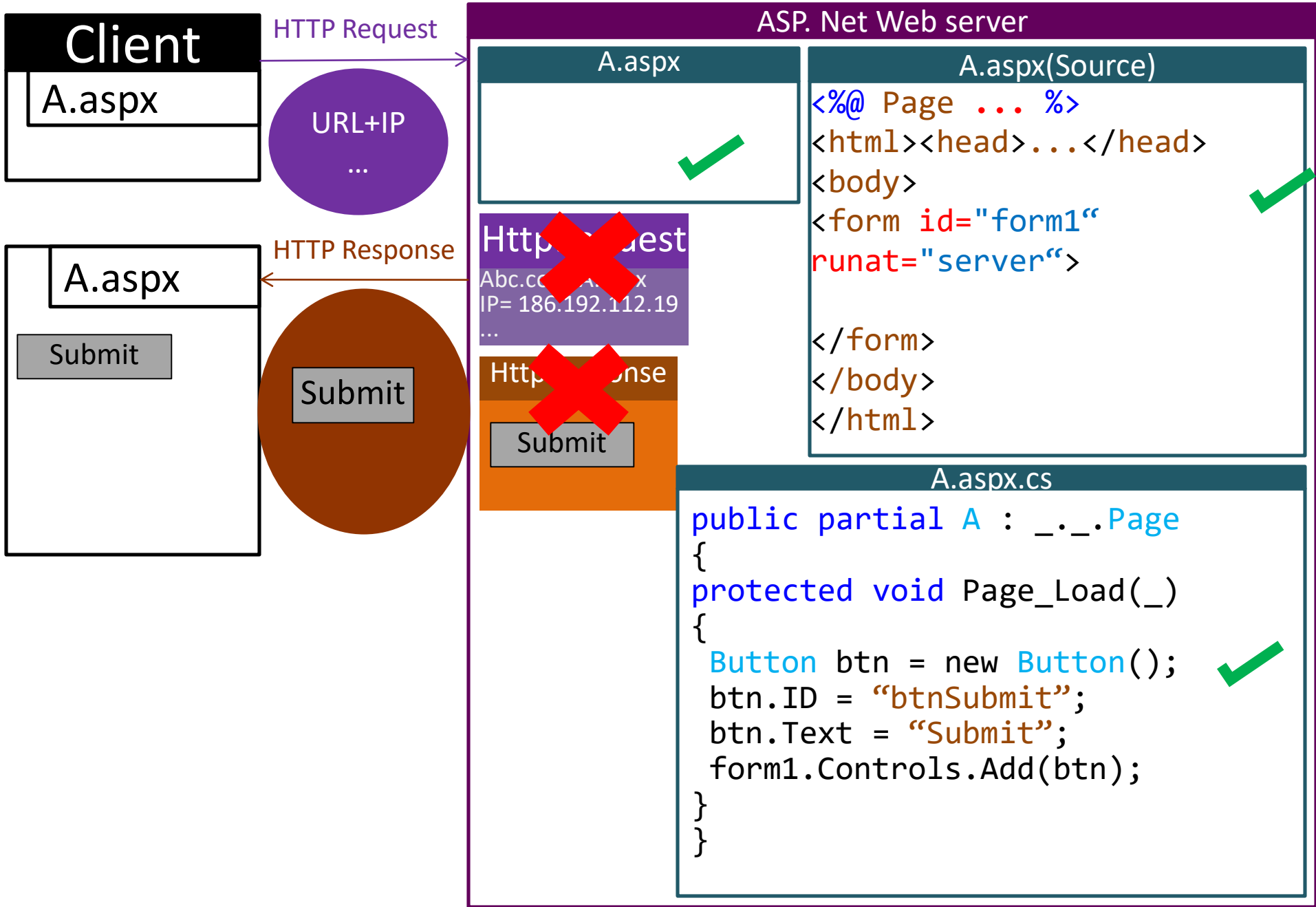
ASP.Net Page life cycle methods

1. **Page_PreInit** – Always executes
2. **Page_Init** – Always executes
3. **LoadViewState** – Executes in Postback Request.
4. **Page_Load** – Always executes
5. **Click event methods or Postback methods**
   – Executes in Postback Request.
6. **Page_PreRender** – Always executes
7. **SaveViewState** – Always executes
8. **Render** – Always executes
9. **UnLoad** – Always executes

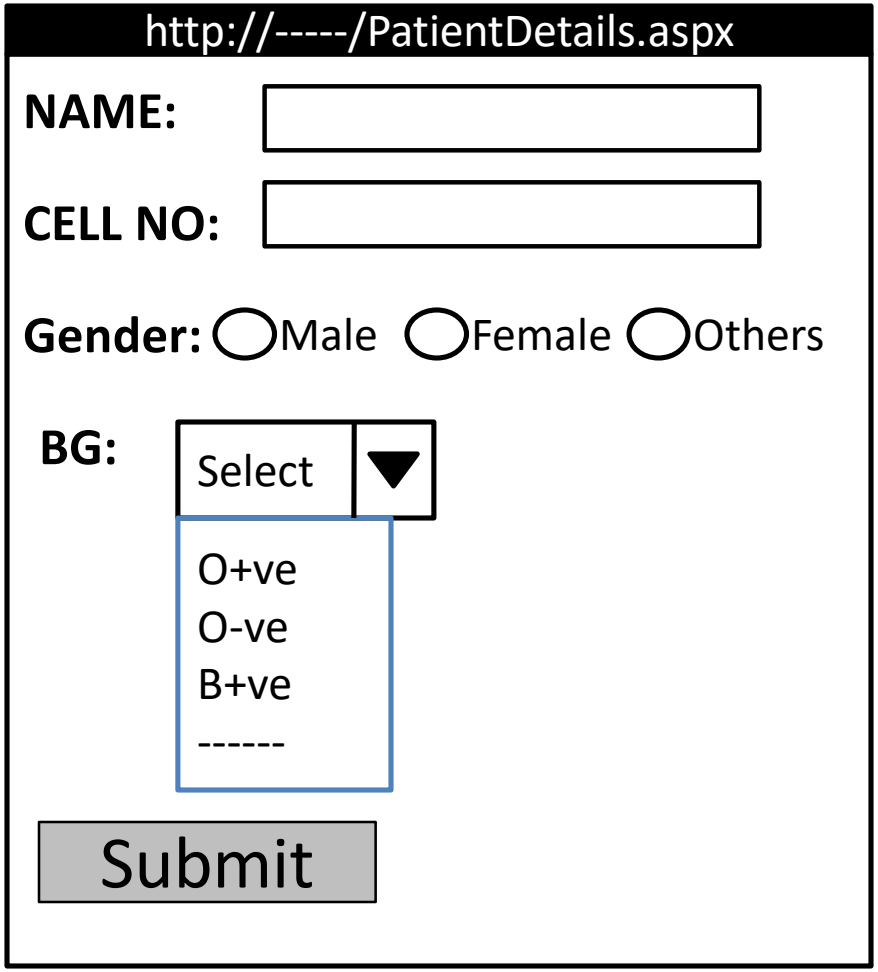
ASP.Net Page life cycle methods

1. **Page_PreInit** – in this method we can change master pages dynamically.
2. **Page_Init** – in this method all controls ID properties initialized.
3. **LoadViewState** – (View state)
4. **Page_Load** – in this method all controls remaining properties are initialized except ID property.
5. **Click event methods or Postback methods** – when control events are triggered.
6. **Page_PreRender** – this method must be used or considered as last method for changing the content in Response Object.
7. **SaveViewState** – (View state)
8. **Render** – this method is responsible for converting all control objects into equivalent HTML tags
9. **UnLoad** – All control objects, httpRequest, httpResponse objects are deleted from Web server memory.

Creating controls Dynamically



Dynamic controls Assignment



PatientDetails.aspx

```
<form
id="form1“
runat="server“>
</form>
```

PatientDetails.aspx.cs

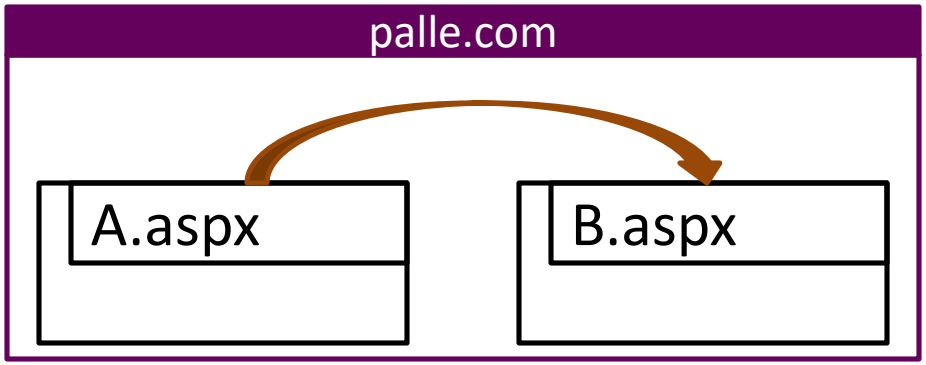
all controls must be created dynamically in **PatientDetails.aspx.cs**

Use this code for adding Br tags dynamically

```
HtmlGenericControl gc = new HtmlGenericControl("br");
form1.Controls.Add(gc);
```

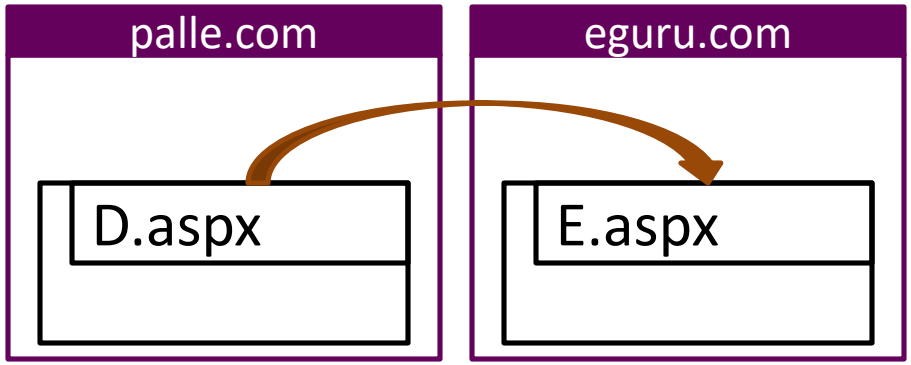
Redirecting users from one page to another page

- **Hyper links** (Anchor tag/Action link).
- **Response.Redirect**
- **Server.Transfer**
- **Server.Execute**

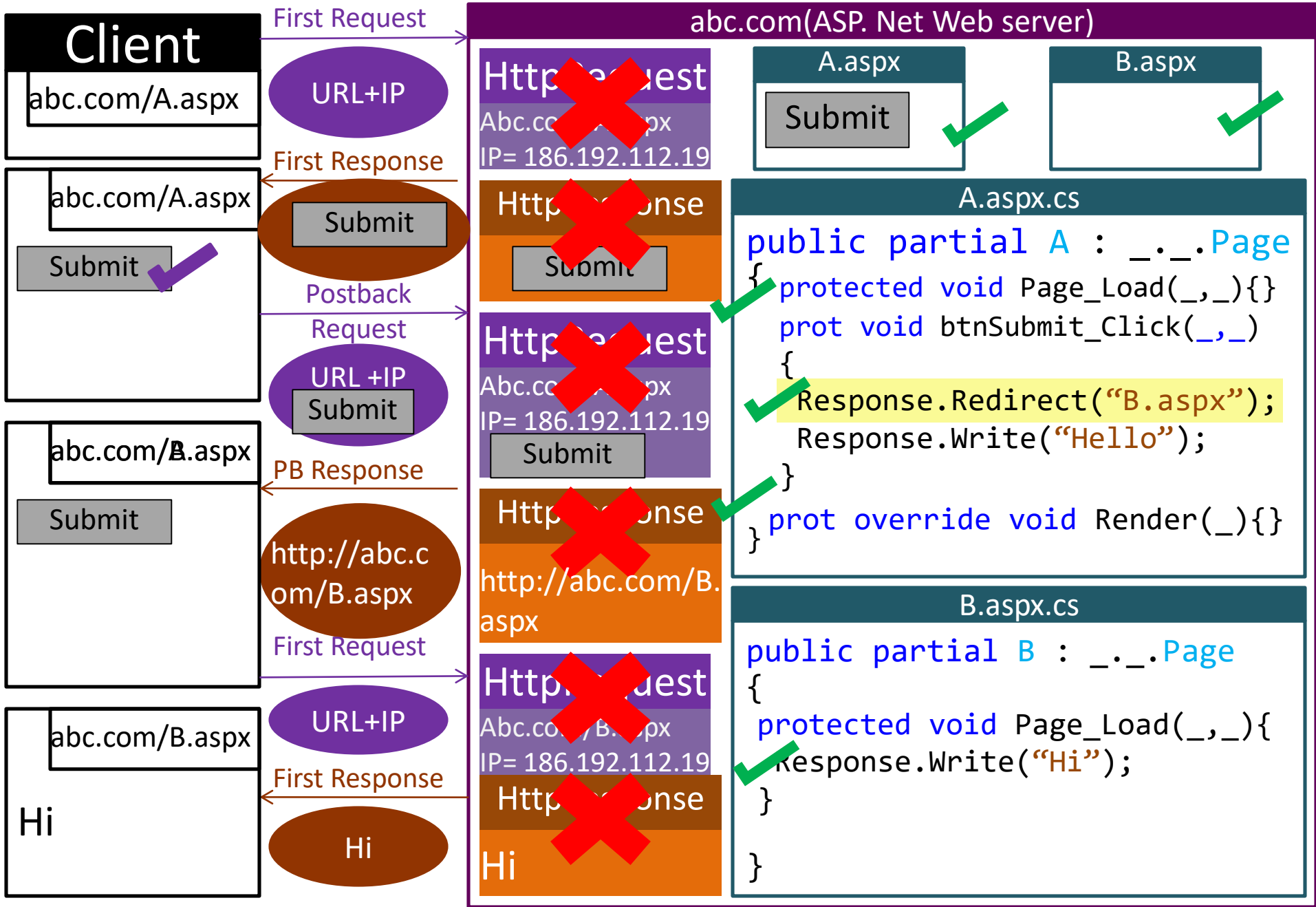


Absolute URL:

```
<a href="http://eguru.com/E.aspx">Goto E</a>
```



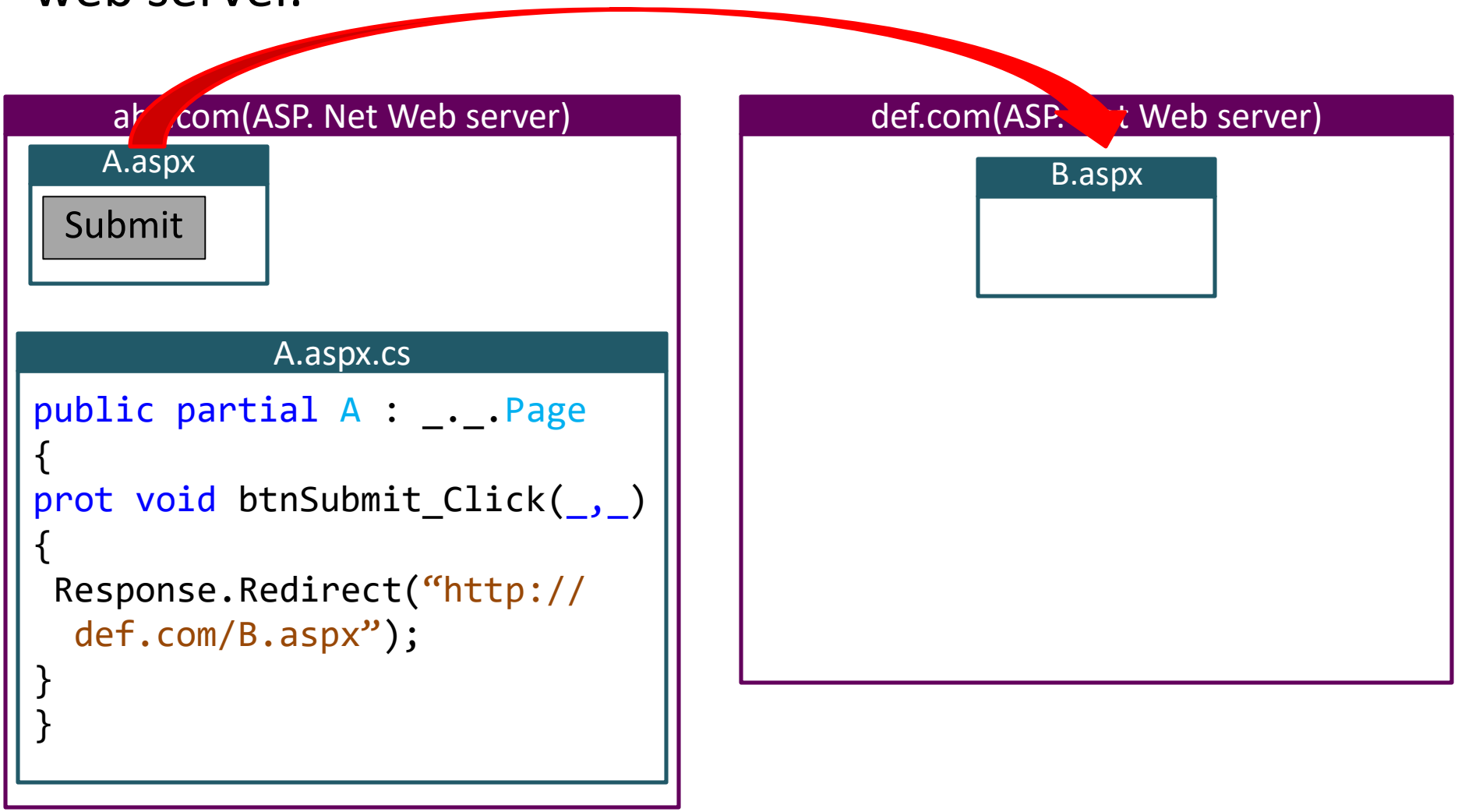
Response.Redirect



Response.Redirect

Response.Redirect() supports **Absolute URL**.

So it is possible to redirect from one seb server to another web server.



Versions of Response.Redirect

1. Response.Redirect(string url)
2. Response.Redirect(string url, bool endresponse)

```
Response.Redirect(string url) =
Response.Redirect(string url, true)

Response.Redirect(string url) !=
Response.Redirect(string url, false)
```

server.transfer vs server.execute

server.transfer transfers the execution to the destination page and the execution will not come back

server.execute transfers the execution to the destination page and the execution will come back to the current page.

ASP.Net Validation Controls

1. **Required Field** Validator
2. **Range** Validator
3. **Compare** Validator
4. **Regular Expression** Validator
5. **Custom** Validator
6. **Validation Summary** (not a validator)

Note:

- By default all **ASP.Net** controls performs **client side and server side validation**.
- If we want we can **disable client side validation** for any validation control by setting `EnableClientScript="false".`
- We **can't disable server side validation**(its Compulsory).

Response.Output.Write

- **Response.Write** will **not support Data formatting**.
- For data formatting we use concatenation while using **Response.Write**
- **Response.Output.Write** supports **Data Formatting**.

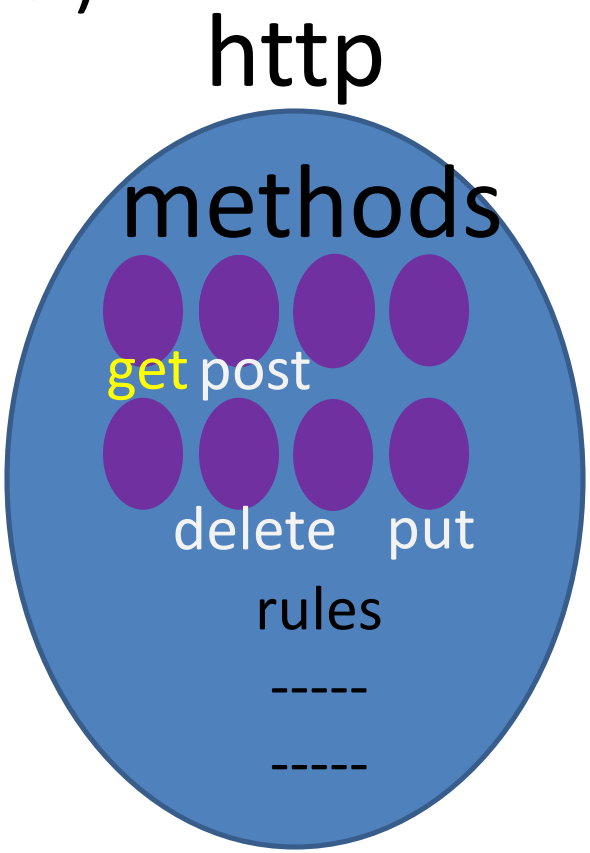
```
int  x = 10;
string y = "palle";
Response.Write("x= "+ x + " y= "+y);
Response.Output.Write("x={0}  y={1}", x, y);
```

Req: Display the output as shown below using a single printing statement.

x=10 y=palle

http protocol

- http is a protocol (protocol → rules + methods)

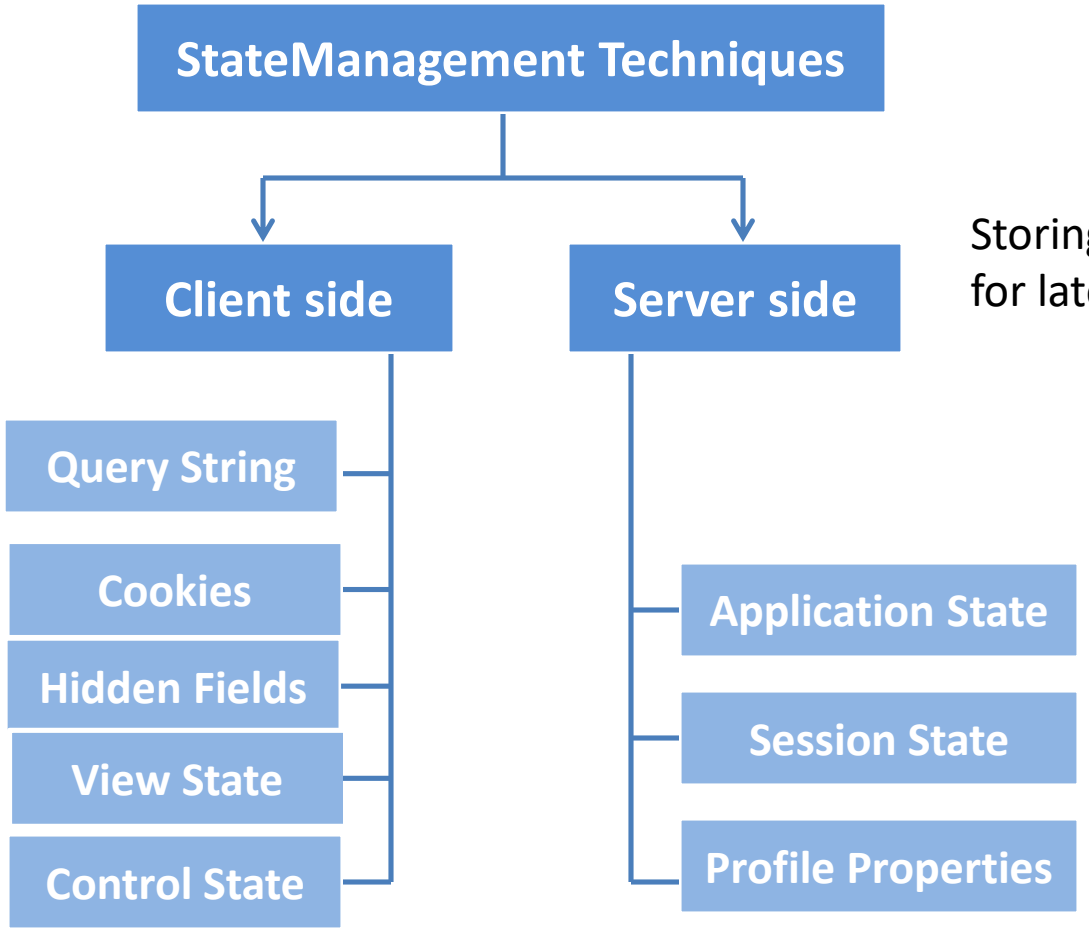


http get and http post method

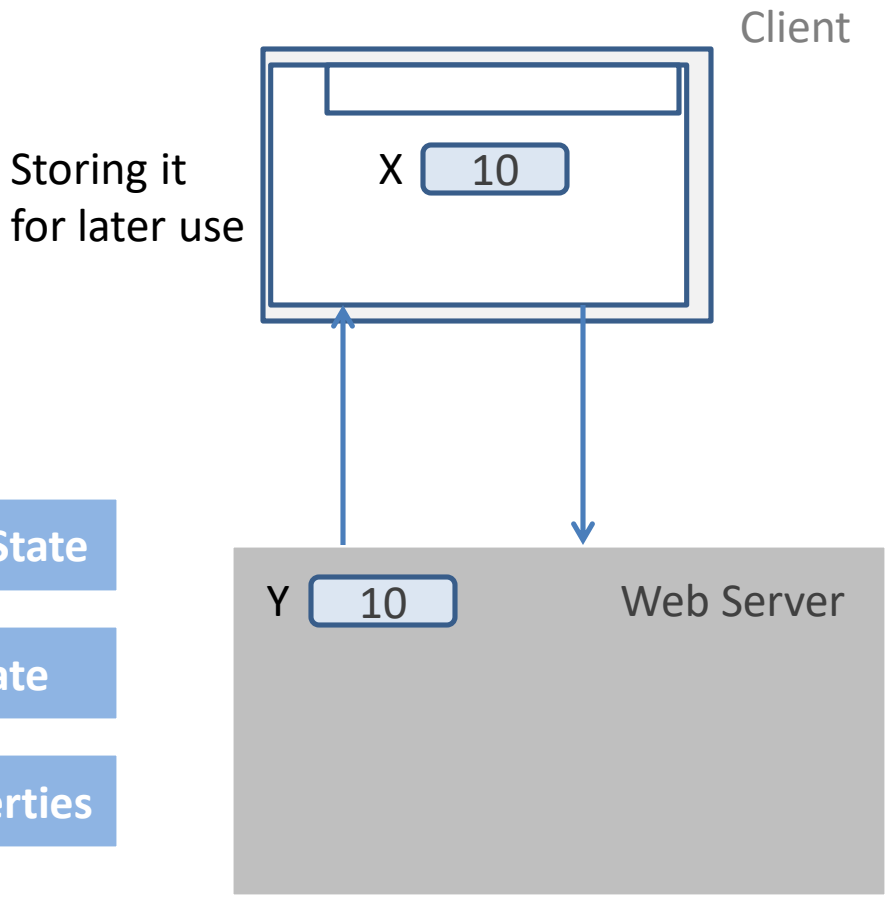
http get	http post
1. This method will usually carry most of the data in **header.** Usually this **data will be visible** to end users in **browsers address bar**.	1. This method will usually send most of the data in **http Body**. Usually http Body **data will not be visible** to end users in browsers address bar.
2. http get will Usually **carry very less data**.	2. http post can **carry more data**.
3. http get output is **cacheable**	3. http post output is **not cacheable.**
4. http get request is **book mark able.**	4. http post request is **not book markable.**
5. http get is **not used for modifying data in Server**.	5. http post is **used for modifying data in Server**.

statemanagement techniques

- Using StateManagement Technique we can persist (or) store some data for later use



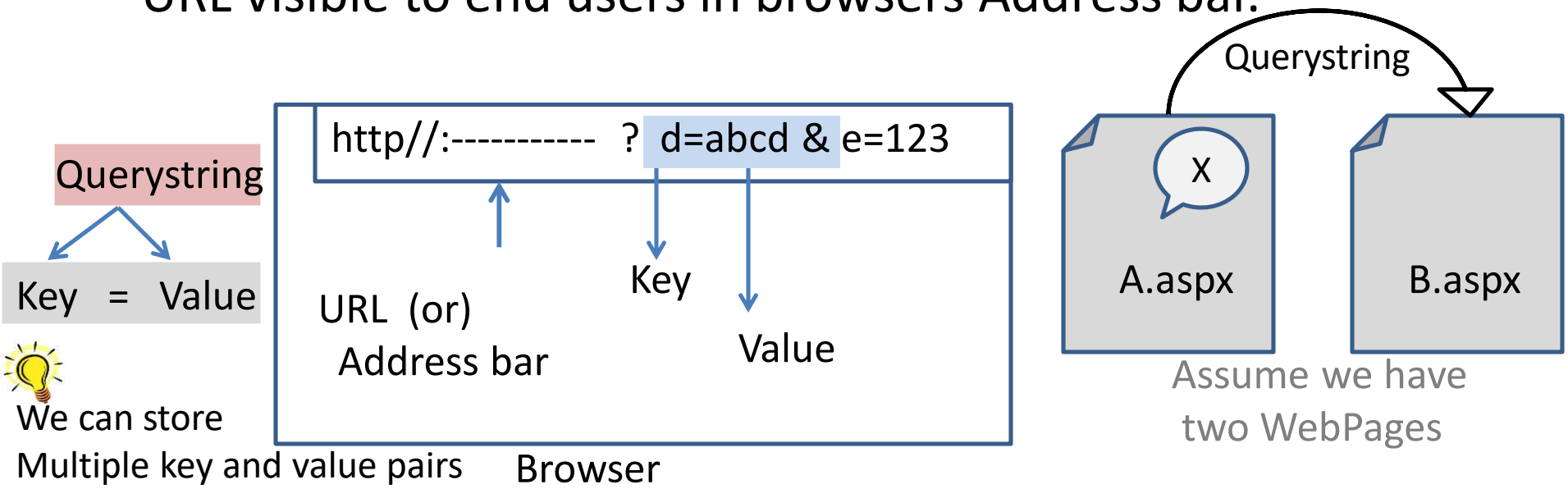
Client side StateManagement technique



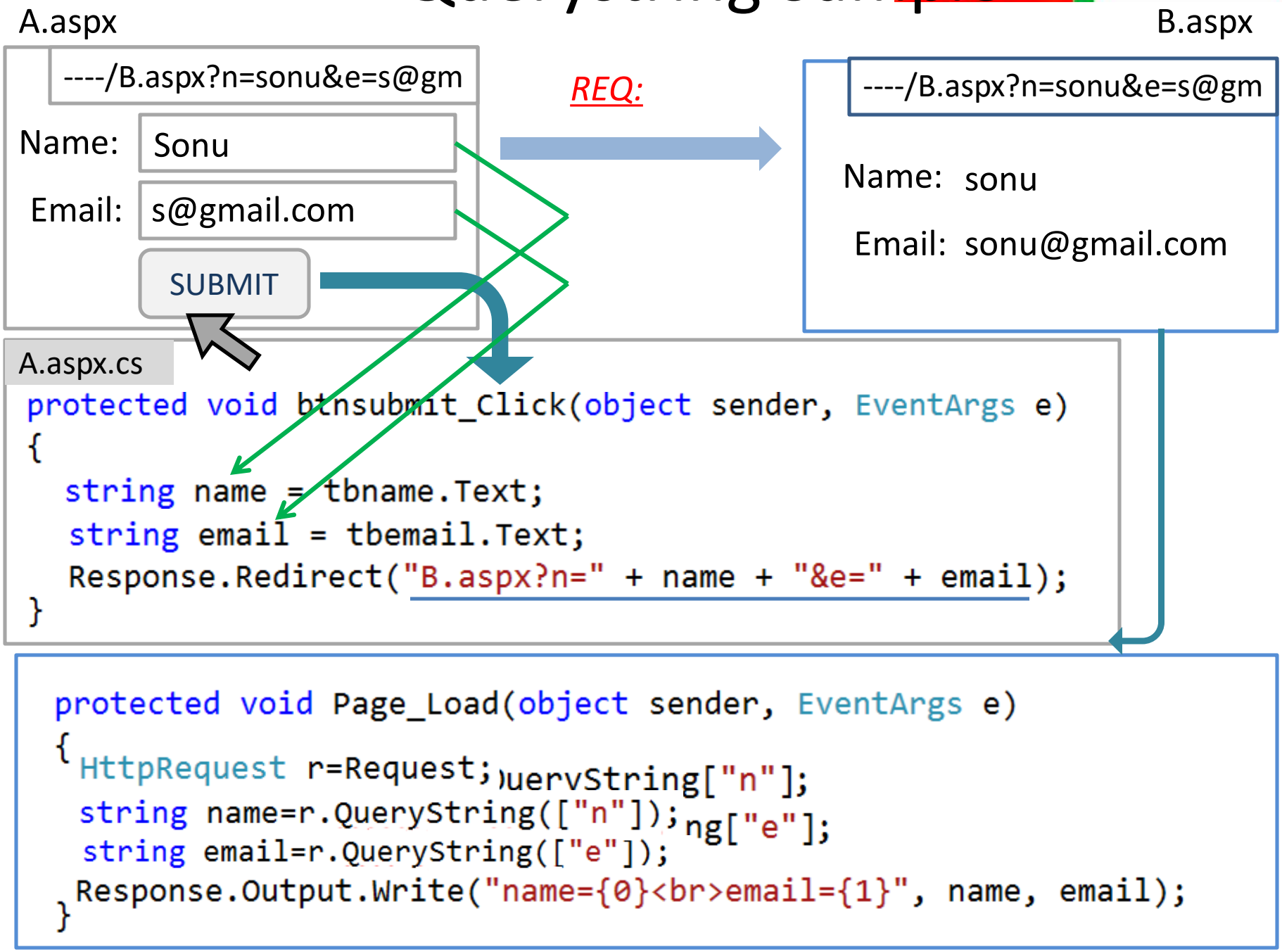
Server side StateManagement technique

query string

- **Query String** te considered as Client side **State management technique**.
- Usually **Query String** data is stored in the **URL**.
- **Query String** is visble to the end user since Query string is stored in URLs.
- It is not recommended to store sensitive data / secured data in **Query String** since query string is part of URL and URL visible to end users in browsers Address bar.



Querystring Sample



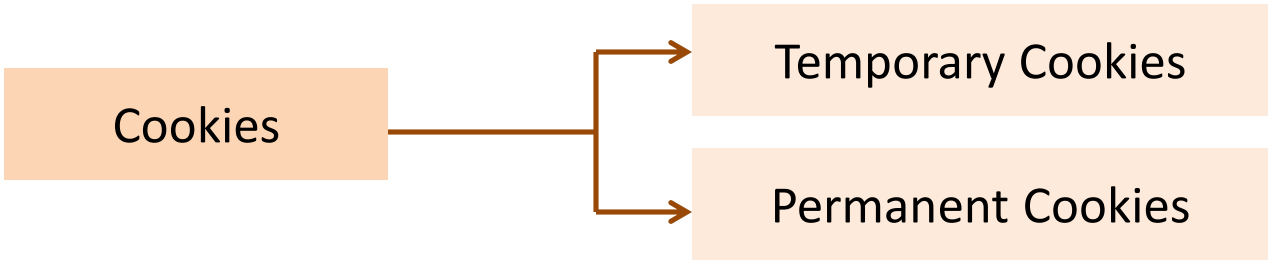
Cookies

- Using **Cookies** we can store **User specific data**.
- A **Cookie** is a **piece of data** which is **sent from web server to browser computer.**
- **Cookies** technique is considered as **Client Side State Management Technique**.
- Usually **Cookies** are created in the Web server and Cookies are stored in the Browser Computer.
- Types of **Cookies** supported in ASP.Net are
  1. Temporary Cookie.
  2. Permanent Cookie.
- It is not recommended to store sensitive data / secured data in **Cookies.**

cookies

- Using cookies we can store user specific data.
- Cookies are created in web-server and sent to browser.

- Cookies are usually stored in browser and hence it is considered as Client side StateManagement Technique.
- It is not recommended to store sensitive data using Cookies



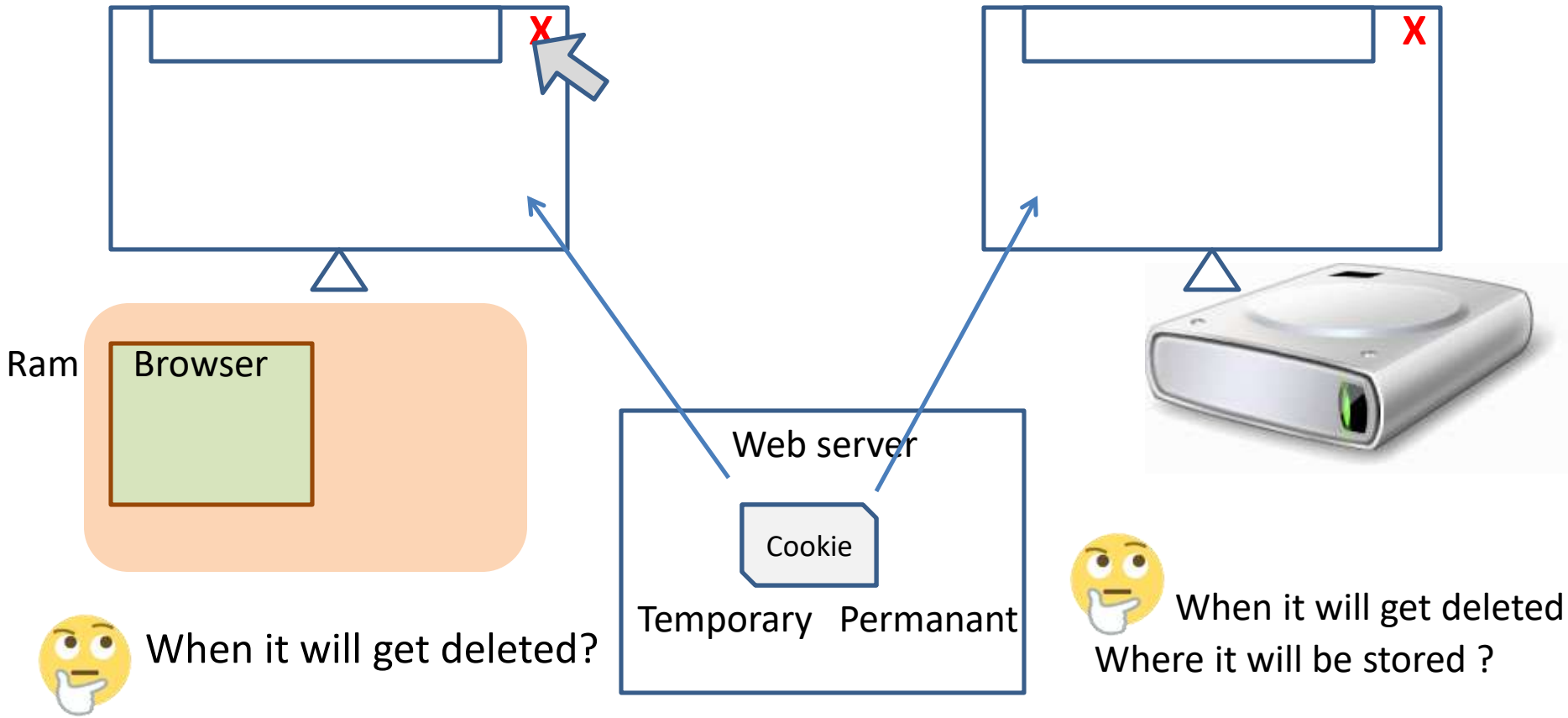
Difference b/w temporary and permanent cookies

Temporary Cookies

1. Temporary cookies are stored in Browser memory
2. Deleted when browser is closed

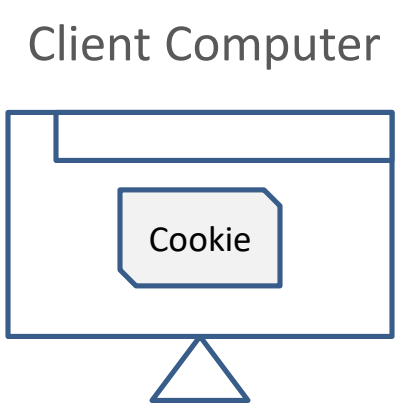
Permanent Cookies

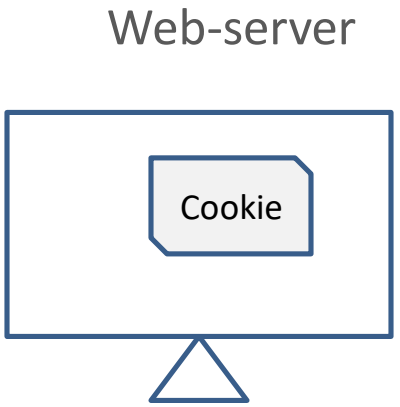
1. Permanent cookies are stored in Hard disk memory
2. Permanent cookie will be deleted when the expiry time elapsed



Cookies between browser and Server

- once the cookies are created and sent to the browser, it will be travelling between client and the web server until the cookie gets deleted





How long it will be travelling?

How will you create a cookie

We will understand about creating cookies in next slide

Creating & reading temporary cookie


```
protected void Page_Load(object sender, EventArgs e)
{
  HttpCookie c = new HttpCookie("pu");
  c.Values["n"] = "10";
  Response.Cookies.Add(c);
  Response.Redirect("B.aspx");
}
```

```
protected void Page_Load(object sender, EventArgs e)
{
  HttpCookie c1 = Request.Cookies["pu"];
  string n1 =Convert.ToString(c1.Values["n"]);
  Response.Output.Write("{0}", n1);
}
```

add following code for creating permanent cookie

```
c.Expires = DateTime.Now.AddMinutes(30);
```

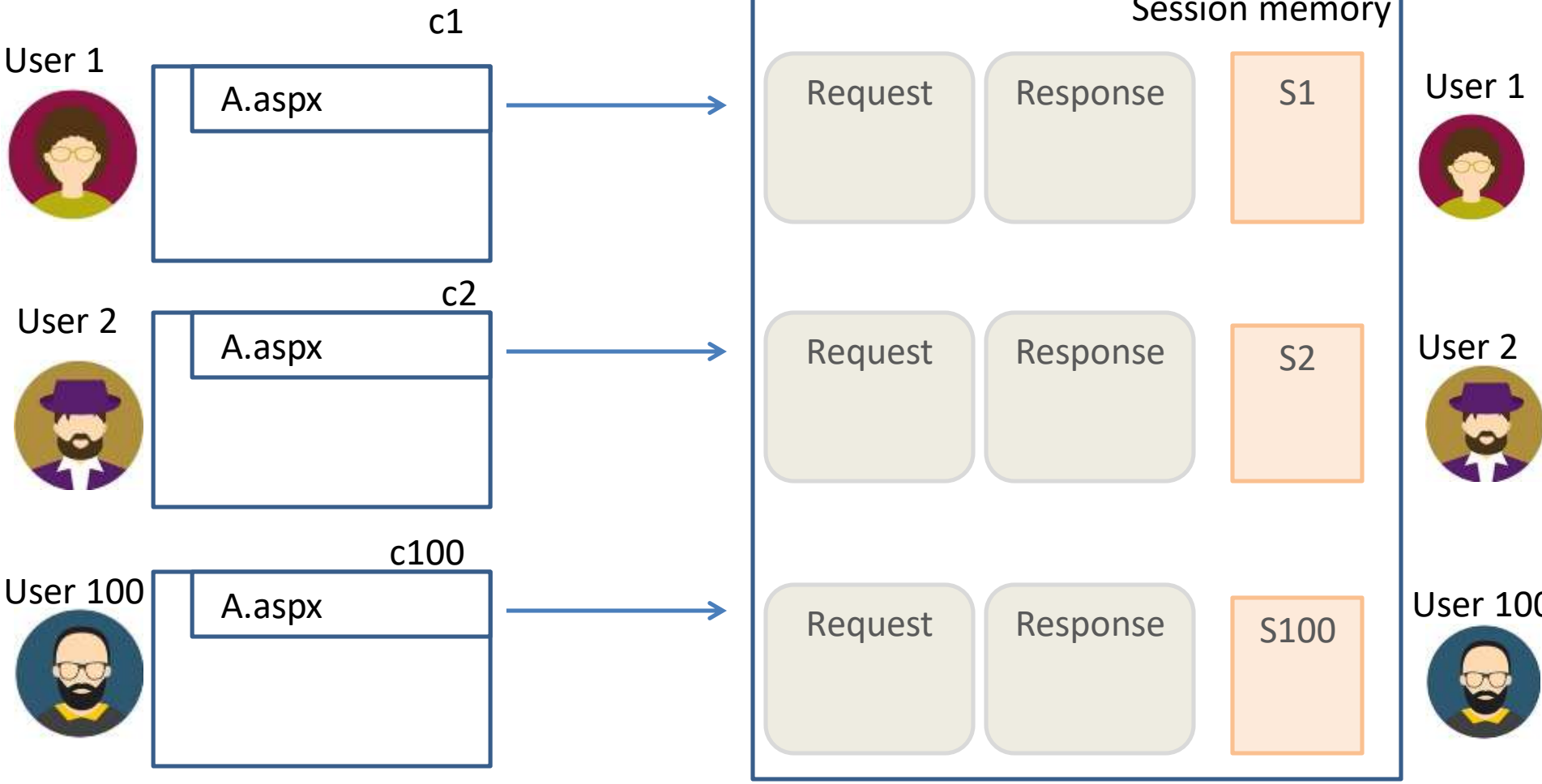
session state

- session state is considered as server side state management technique.
- session state is used for storing user specific data
- when a session state is enabled for each and every user a session memory will be allocated
- usually session will be stored in the webserver for 20 min.

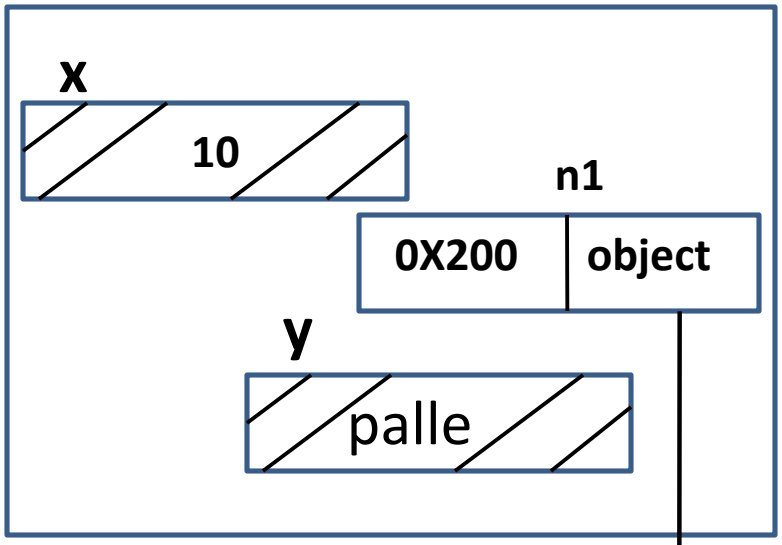
session creation (when?)

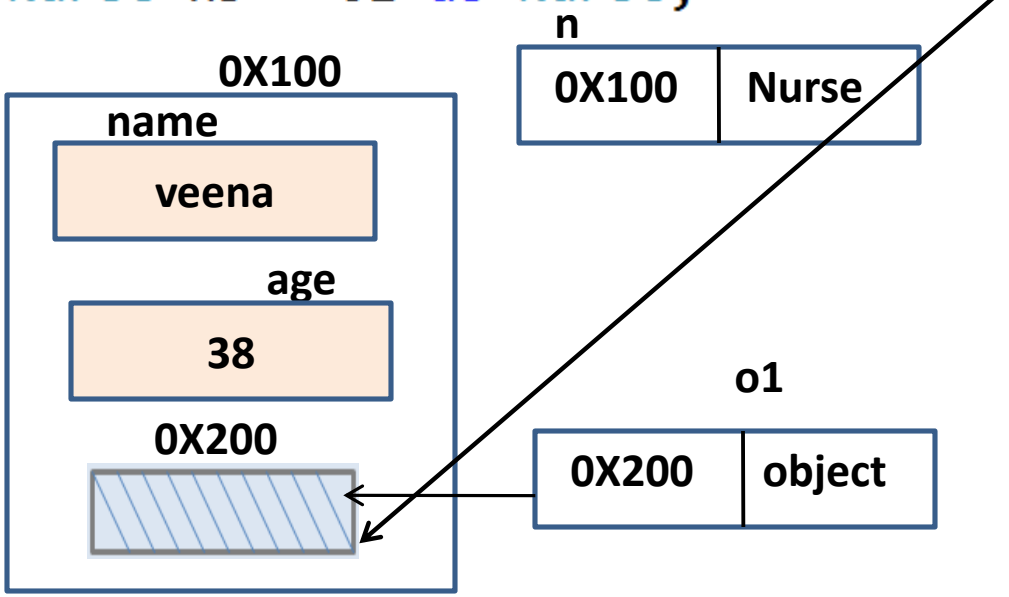
The session memory s1 is created uniquely for user1.

For each User a unique session memory is created



```
Session["x"] = 10;
Session["y"] = "palle";
int x1 = Session["x"];
int x1 = (int)Session["x"];
string y1=(string)Session["y"];
Nurse n = new Nurse("veena", 38);
session["n1"] = n;
object o1 = session["n1"];
Nurse n2 = (Nurse)o1;
Nurse n3 = o1 as Nurse;
```






```
public class Nurse
{
 public string name;
 public int age;
 public Nurse(string n, int a)
  {
      name = n;
      age = a;
  }
}
```

types of session state

Session State

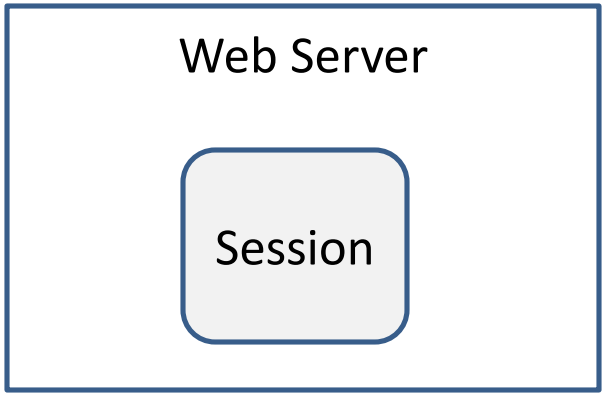
Apart from Web server
Where can we store Sessions

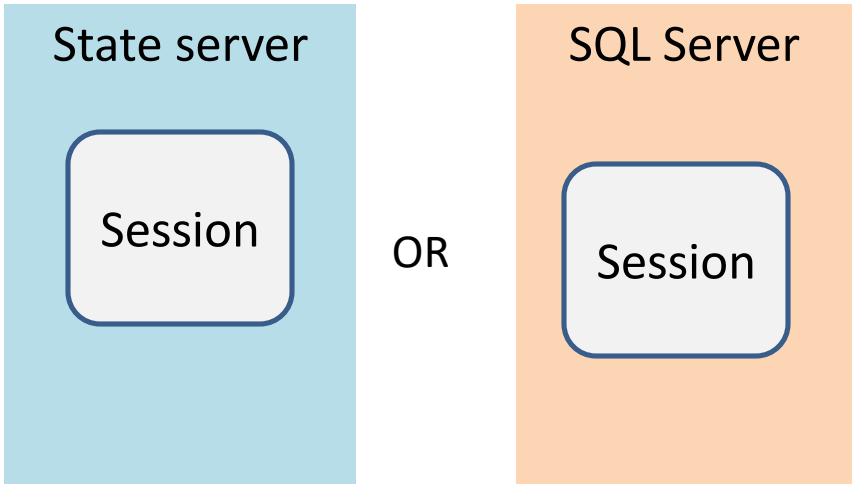
In proc Session State

Out proc Session State

Session stored inside Web server are called In proc session.

Session stored Outside Web server are called Out proc session.





session state internals

User1

RAM

Cookie
Sid=S14

C1

A.aspx

Name:

Age :

Submit

RAM

Cookie
Sid=S27

C2

User2

b.aspx

Name: Paul

Age : 25

A.aspx.cs

```
Page_Load()
 btn_Click()
Session["n"] = tbn.text;
Session["a"] = tba.text;
Response.Redirect
            ("B.aspx");
```

Response
---html---
Cookie
Sid=S14

Request

S14
Session

Cookie
Sid=s14

Response
--html--
Cookie
Sid=S27

Request

S27
Session
n=paul a=25

Cookie
Sid=s27

B.aspx.cs

```
Page_Load()
string n=(String)Session["n"];
int a=(int)Session["a"];
R.O.W("Name={0},Age={1}",n,a);
```

Response
URL-B.aspx

Request
Cookie
sid=s27

Response
Name=paul
Age=25
Cookie

Request
Cookie
sid=s27

Name:

Age :

Submit

A.aspx

B.Aspx

EMPTY

cookieless session

Palle Technologies

Incase of cookieless session sessionid stored in URL

How to enable Cookieless Session?

Web.config

```
<sessionState cookieless="true"/>
```

When user access any page in the website user will see Sessionid in URL

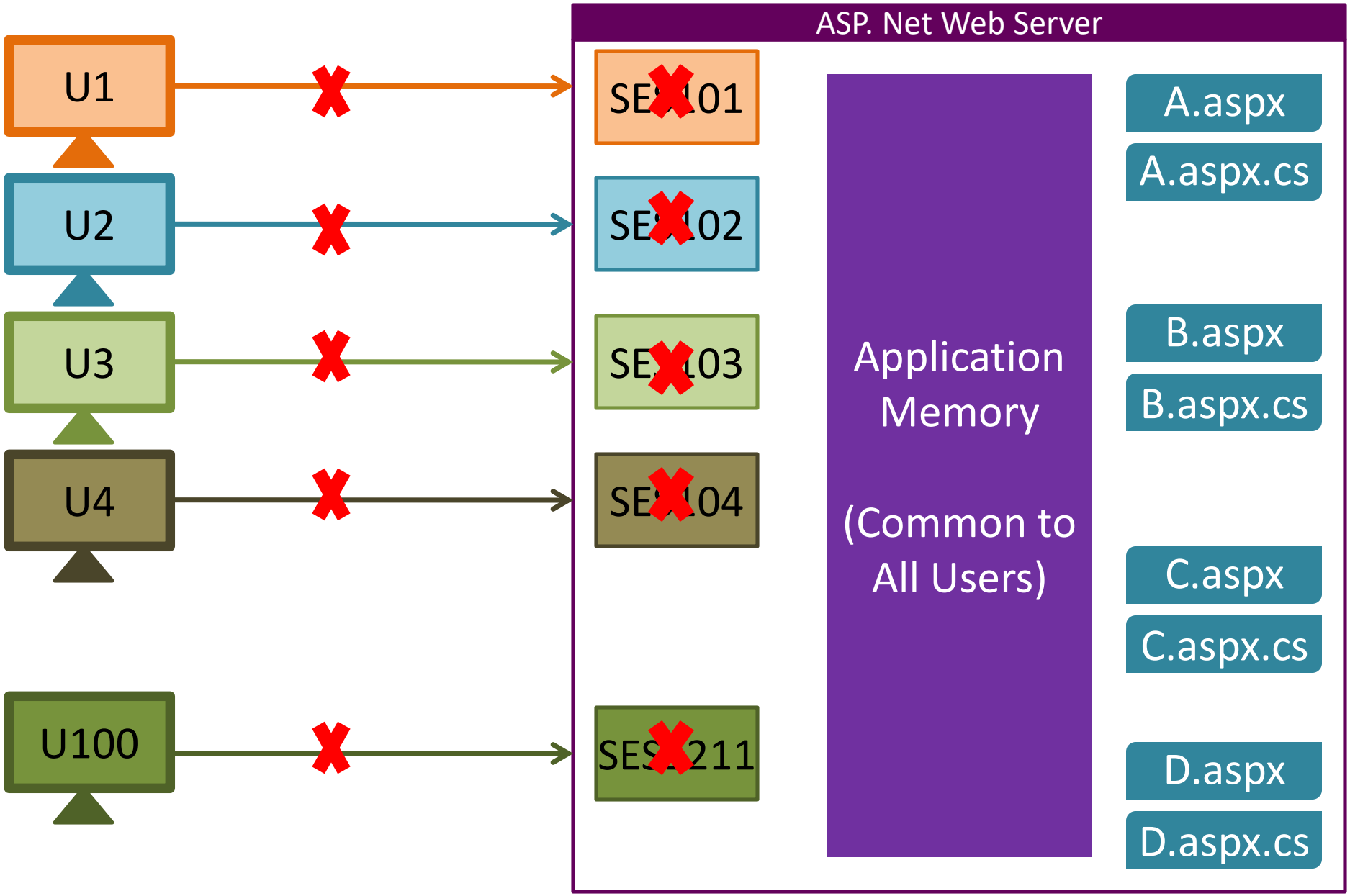
http://----------/(S(ut4ks2twld15yhbv2rneqh4o))/A.aspx

application memory or application state

- **Application state** is a server side **state management technique**.
- **Application state** is **common for all user** (it is recommended to use application memory for storing data which is common for all users).
- **Application memory** is exists until the web server restarts or shutdown.
- It is recommended to write code for inserting or modifying data present in **application memory** between `Application.Lock()` and `Application.UnLock()` to avoid **Thread Synchronization effects**.

```
Application.Lock();
Application["x"] = 100;
Application.UnLock();
```

application memory or application state



reading and adding data to application state

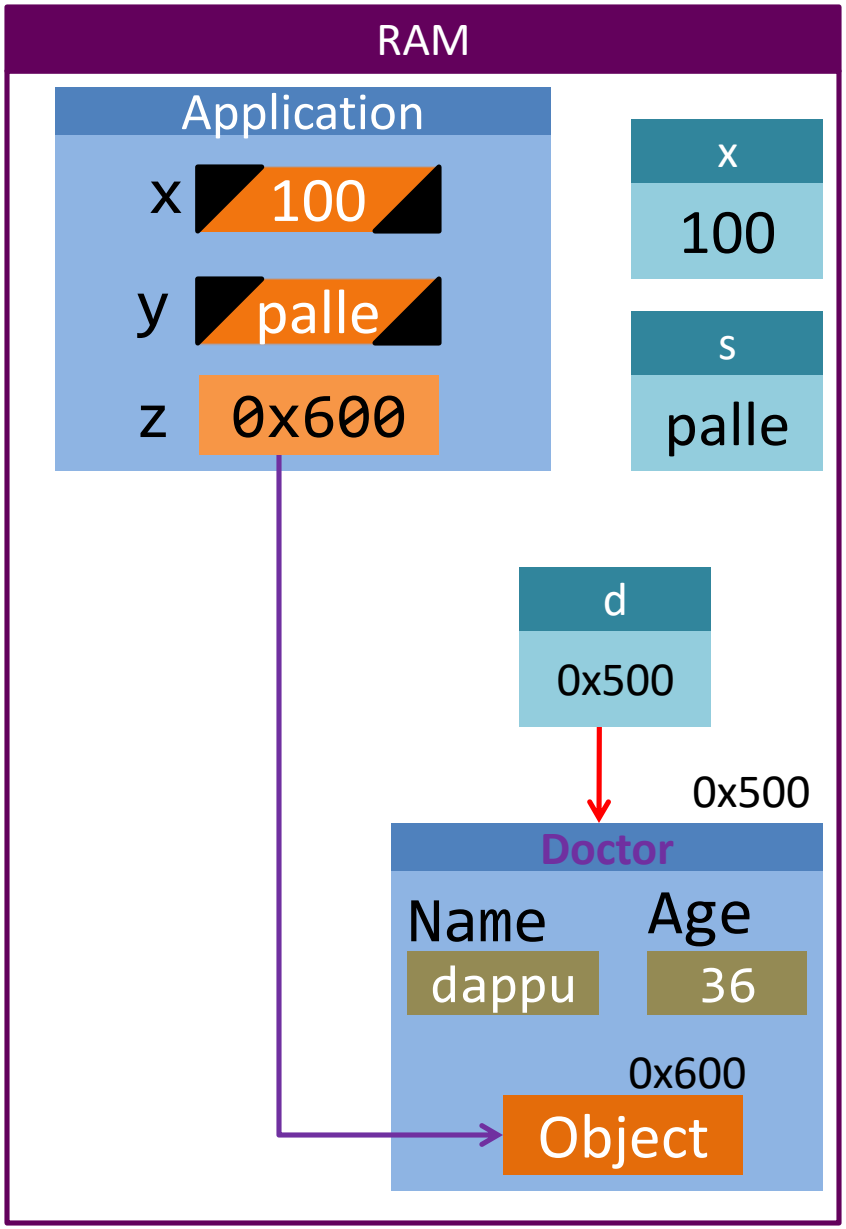
```
Class Doctor{
   public string Name;
   public int Age;
   public Doctor(string n, int a)
   { Name = n; Age = a; }
}
```

Writing to Application State:

```
Application["x"] = 100;
Application["y"] = "palle";
Application["z"] = new
   Doctor("dappu", 36);
```

Reading from Application State:

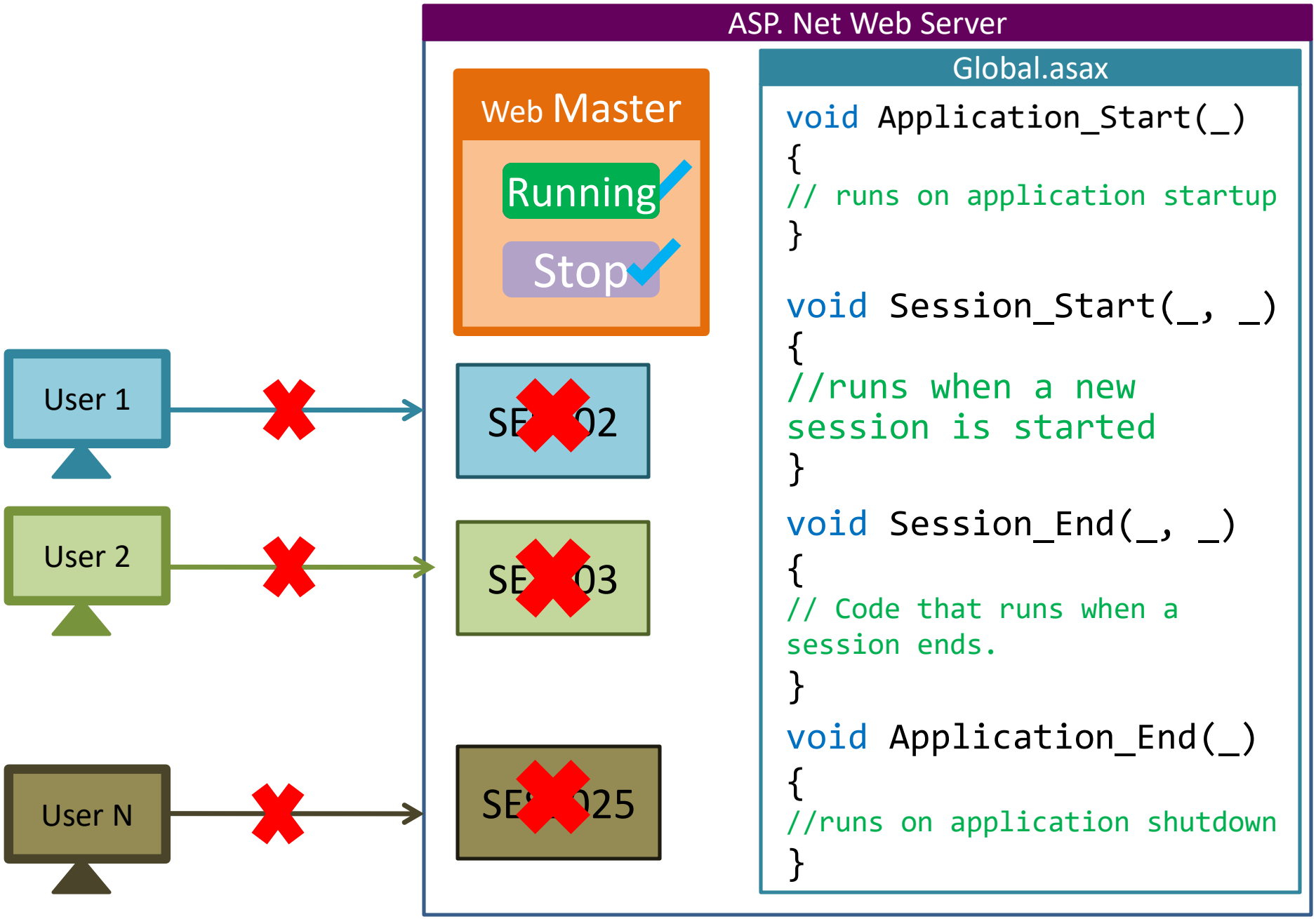
```
int x = (int) Application["x"];
string s = (string) Application["y"];
Doctor d = (Doctor) Application["z"];
```



global.asax

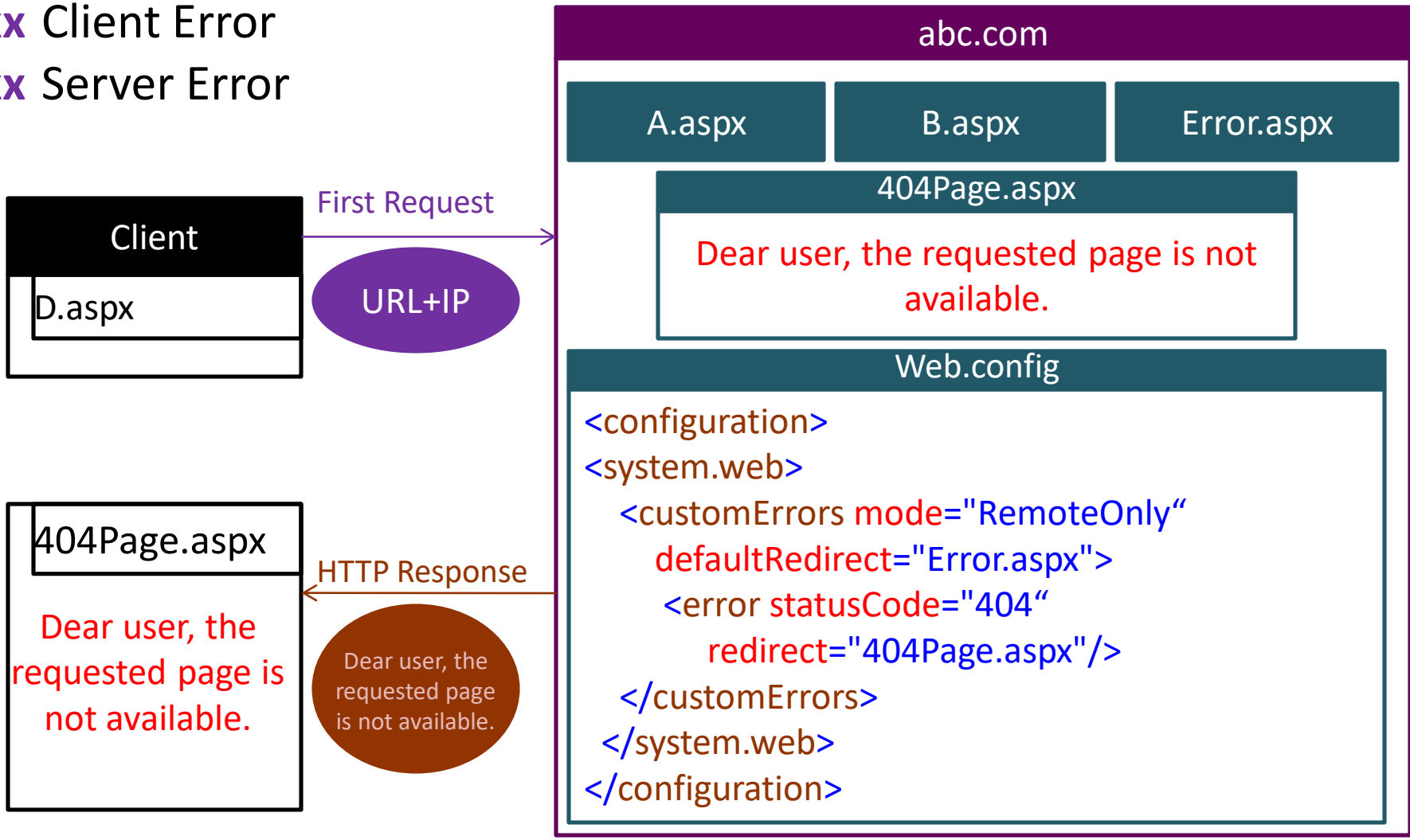
- Global.asax file usually contains application life cycle methods.
- Only one Global.asax file is supported per website.

Application Life cycle Methods



HTTP Status codes

1xx Information messages
2xx Success messages
3xx Redirection messages
4xx Client Error
5xx Server Error



Configuration files

- Using Configuration files we can change application behavior dynamically without re-compiling the application.
- Configuration files contains xml language code.

- Supported configuration files in .Net:

1. **machine.config**
2. **app.config**
3. **web.config**
4. **user.config**

configuration files – part 2

1. one or more web.config file allowed / web application

2. one or more machine.config file per computer

web.config

Success Message
Error Message
Database Connection String
Debug Settings
Session Settings
Etc.,

authentication & authorization

what is authentication: ?

Authentication is all about finding whether user is a valid user or not?

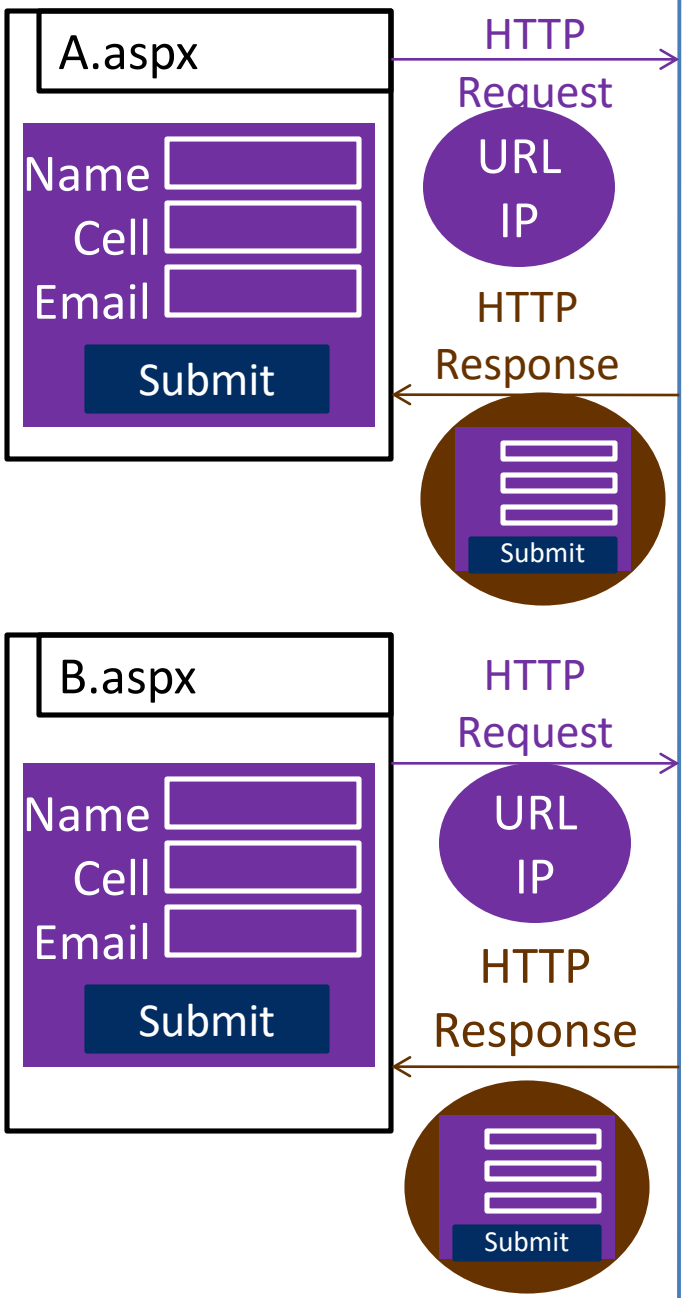
what is authorization: ?

Authorization is all about finding whether a specific user is allowed to access a specific page or not?

user controls

- **User Control** is a composite control (User control contains subcontrols)
- **User Control** are usually used for avoiding/reducing **UI duplication** and **functionality duplication**.
- extension used for user controls file extension is **.ascx** and can contains **HTML**, **C#**, **Javascript**, **CSS** and **ASP.Net** tags.
- Inside **User Control** files we are not allowed to use **<html>**, **<head>** and **<body>** tags
- all **User Control** files inherits **`System.Web.UI.UserControl`** Base class.

creating user controls



QuickEnquiry.ascx

```
<%@ Control … %>
<asp:TextBox ID="tbName" runat="server" >
</asp:TextBox>
<asp:TextBox ID="tbCell" runat="server" >
</asp:TextBox>
<asp:TextBox ID="tbEmail" runat="server" >
</asp:TextBox>
<asp:Button ID="btnSubmit" Text="Add"
runat="server" OnClick="btnSubmit_Click" />
```

A.aspx

```
<%@ Page … %>
<%@ Register Src="~/QuickEnquiry.ascx" TagPrefix="Palle"
TagName="Tech" %>
…
<form name="form1" runat="server">
   <Palle:Tech ID="uc1" runat="server" />
</form>
```

B.aspx

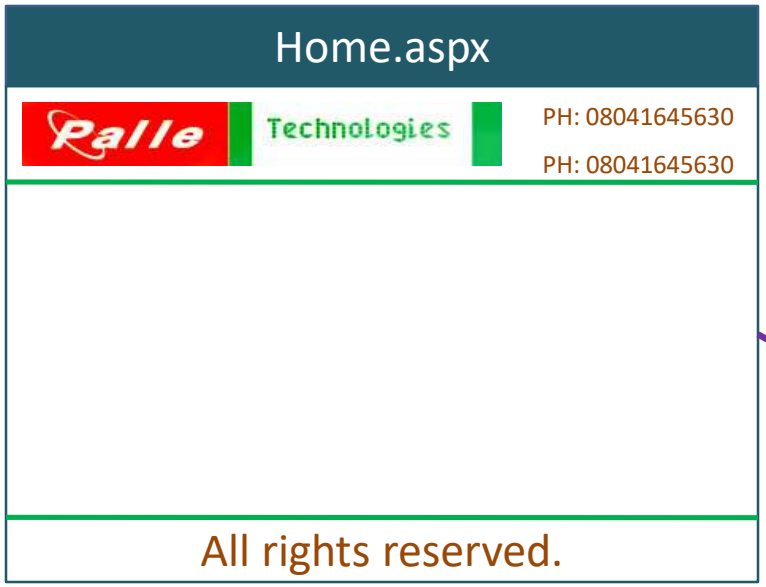
```
<%@ Page … %>
<%@ Register Src="~/QuickEnquiry.ascx" TagPrefix="User"
TagName="Control" %>
…
<form name="form1" runat="server">
   <User:Control ID="uc2" runat="server" />
</form>
```

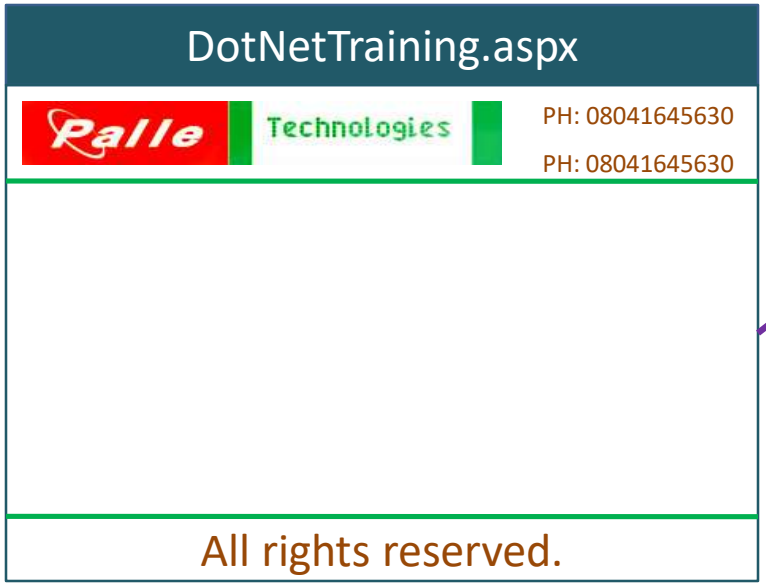
master pages (.master)

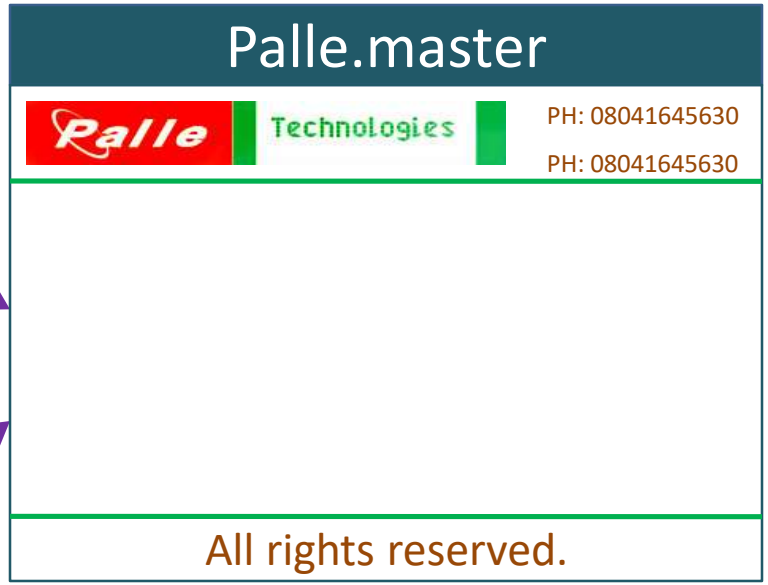
- **Master Pages** are usually used for giving consistent look and feel for multiple web pages present in a website.
- using **Master Pages** we can avoid UI duplication which will occur while creating multiple web pages in a website.
- **____.master** is the extension given for **Master Page** files.
- **Forever .master** file there will be a corresponding **___.master.cs** code file.
- **Master Pages** supports **`<html>, <head>, <body>, <asp:ContentPlaceHolder>`** tags.

  (but User Contols does not supports)

when to use master pages







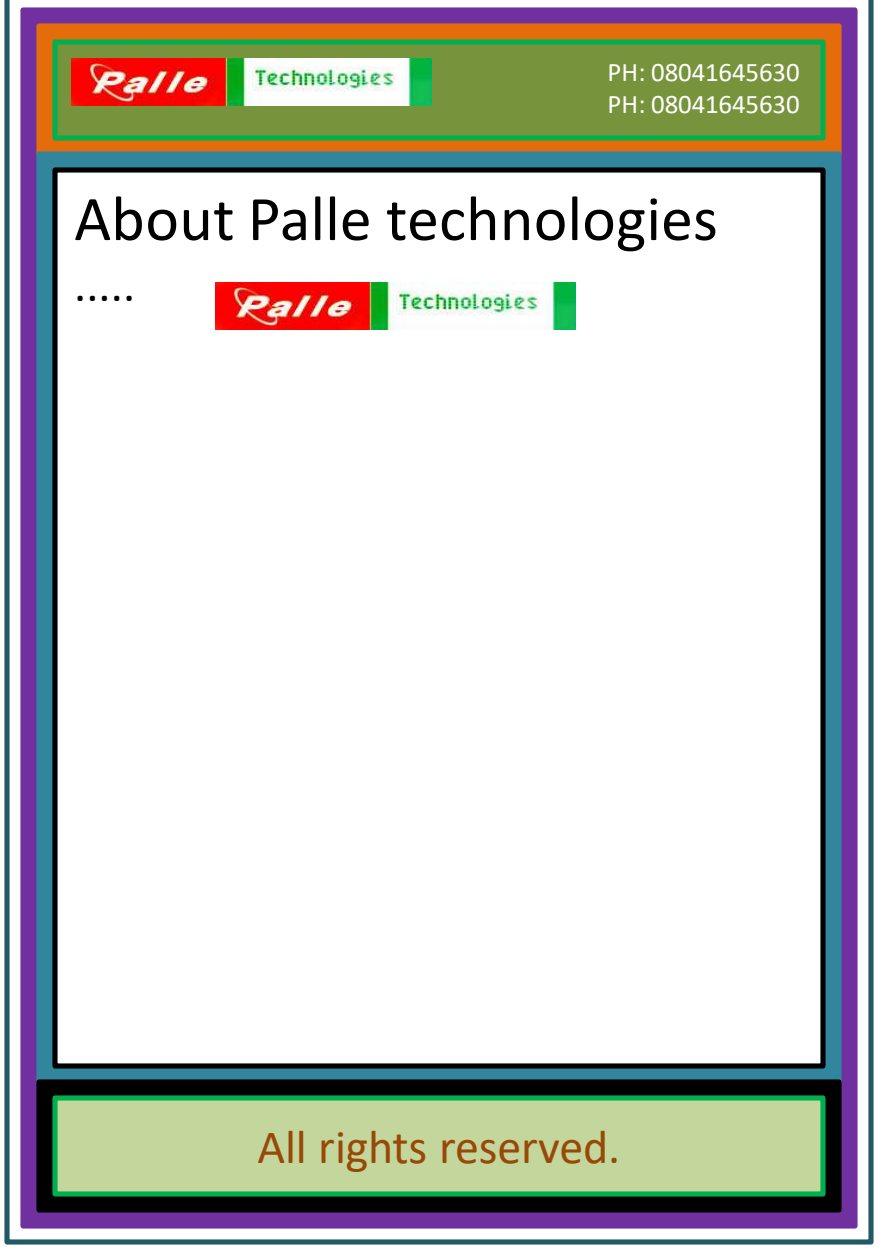
Rules for Content Page:

- **.aspx** file must not contain **<html>, <head>, <body>** tags.
- Every linked **.aspx** file must be linked with a master page.

creating and using master pages

Palle Technologies

Palle.master



Palle.master

```
<%@ Master … %>
<html><head></head>
<body>
<form>
<table>
 <tr style="height:15%;"><th>...</th></tr>
 <tr style="height:70%;"><td>
  <asp:ContentPlaceHolder id="cph1">
  </asp:ContentPlaceHolder>
 </td></tr>
 <tr style="height:15%;"><th>...</th></tr>
</table>
</form>
</body></html>
```

Home.aspx

```
<%@ Page MasterPageFile="~/Palle.master"
… %>
<asp:Content ID="cntHome"
  runat="server"
  ContentPlaceHolderID="cph1">
  <h1>About Palle technologies</h1>
  …
</asp:Content>
```