→ Code for sample1.aspx.cs

```
Protected void btnsubmit_click ( "" )
{
Cache ["a"] = txtsample1.Text;
Cache ["b"] = txtsample2.Text;
Response.Redirect ("sample2.aspx");
```

→ Code for sample2.aspx.cs

```
Protected Void page_Load ( "" )
{
txtsample1.Text = Cache ["a"].Tostring ();
txtsample2.Text = Cache["b"].Tostring ();
}
```

Page Output Catche :-

→ Page output catche is used to Maintain the catche for that Entire Webpage.

→ At first Request of Any client Request will be processed and stored in catche.

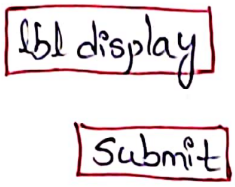
→ From the next request of the same client (or) different client Request Will Not be reprocessed rather, catche result will be displayed.

→ To Maintain the output catche We use

→ %@ Output catche page directive. like

→ <%@ Output Catche Duration = "Time in secs"
Var By param = "none"%>

* Example To implement page output cache:-



Go to .cs file write the following code:-

```
Protected void page_Load ( " " )
{
 lbl display. Text = DateTime.NOW. To string ( );
}

Protected Void btnsubmit_click ( " " )
{
lbl display . Text = DateTime. NOW. To string ( );
}
```

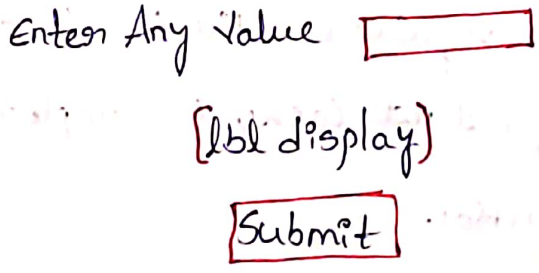
→ To Implement page output cache go to .aspx file Switch to Source Write the following code.

→ <%@ output cache Duration = "120" VaryByparam = "none

* Example With Vary By param :-

→ Usually catche data will be updated after completi of time period that is set in duration.

→ But if would like to update the cache based on a parameter value Then we can use Vary by para attribute.



→ Go to .cs file write the following code for submit button click Event.

```
Protected void btnsubmit_click ( " " )
{
lbl display . Text = DateTime.Now. To string ();
}
```

→ Go to .aspx file Go to Source Write the following code To Implement page out put Cache.

→ <%@ output Cache Duration = "120" Var By param = "txtsample1" %>

→ In this case, Cache will be updated in two Conditions

1. When duration Completes
2. When data in Textbox txtsample1 changes

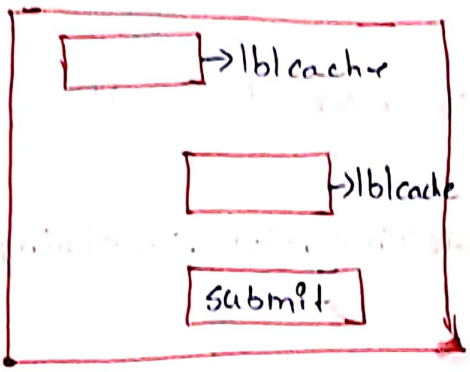
* Partial page cache / fragment Cache :-

→ Partial page cache is used to Implement the cache for some position of the page for not for other position of the page.

→ When we Want to Implement the cache for some position of the page all the controls for Which cache is required Will be in a user control and page out put cache is implemented for the user control.

→ When this user control is included in other Web page, Then cache will be Implemented for the portion of user control for that page and not Implemented for other portion of the Webpage.

* Example With partial page cache



→ Create a web user control with the name UsrCtrlCache.

→ Create a label Go to source and Write the following code.

→ <%@outputcache Duration = "90" VaryByparam = "none"%>

→ Write the following code in page Load Event of the uses control.

```
Protected Void Page_Load (          ||      )
{
lblcache.Text = DateTime.Now.ToString();
}
```

→ Create a New Webpage With the Name Samples.aspx.

→ Create a label and Submit button Write the following code in Source to include the user control.

→ <%@ Register src="~/usrctrl cache.ascx" Tagprefix
="UC" TagName ="pCache"%>

<Form Id =" —>

→ <UC : pCache ID="UC1" runat ="Server" />

→ Go to .cs file Write the following code in Submit button.

```
Protected Void btnsubmit_click ( " " )
{
 lblNo Cache.Text = DateTime.Now.To string ();
}
```

* Control State:

→ Control state is ~~stressed~~ used to Maintain the state for individual control within Asp.net. Control state is responsible to Maintain data. and all other properties state for a control In Asp.net.

→ As a developes We didnot Write any code to Maintain state for the control..

→ Microsoft developers already return code to Maintain state for the control.

* View State:

→ View State is used to Maintain state of a Web page.

→ View State will store the data in Encrypted form. and View State is hidden object. that is User cannot see the view state data.

→ once View State is Created View State always will travel from client request to Server and Server response to client.

→ View State will reduce the application performance in general.. if we want to store our own data in view state We can store like

→ View state ["Key"] = Data;

→ View State will store the data with object type.

→ as a developes We can retrieve the data of view state and can do any modifications if required.

* Page life cycle in Asp.net:

→ Page life cycle represents the various stages Page will undergo once request is submitted to Server until response is delivered to the client.

→ Usually along with stages there are various Events Included in Every page life cycle.

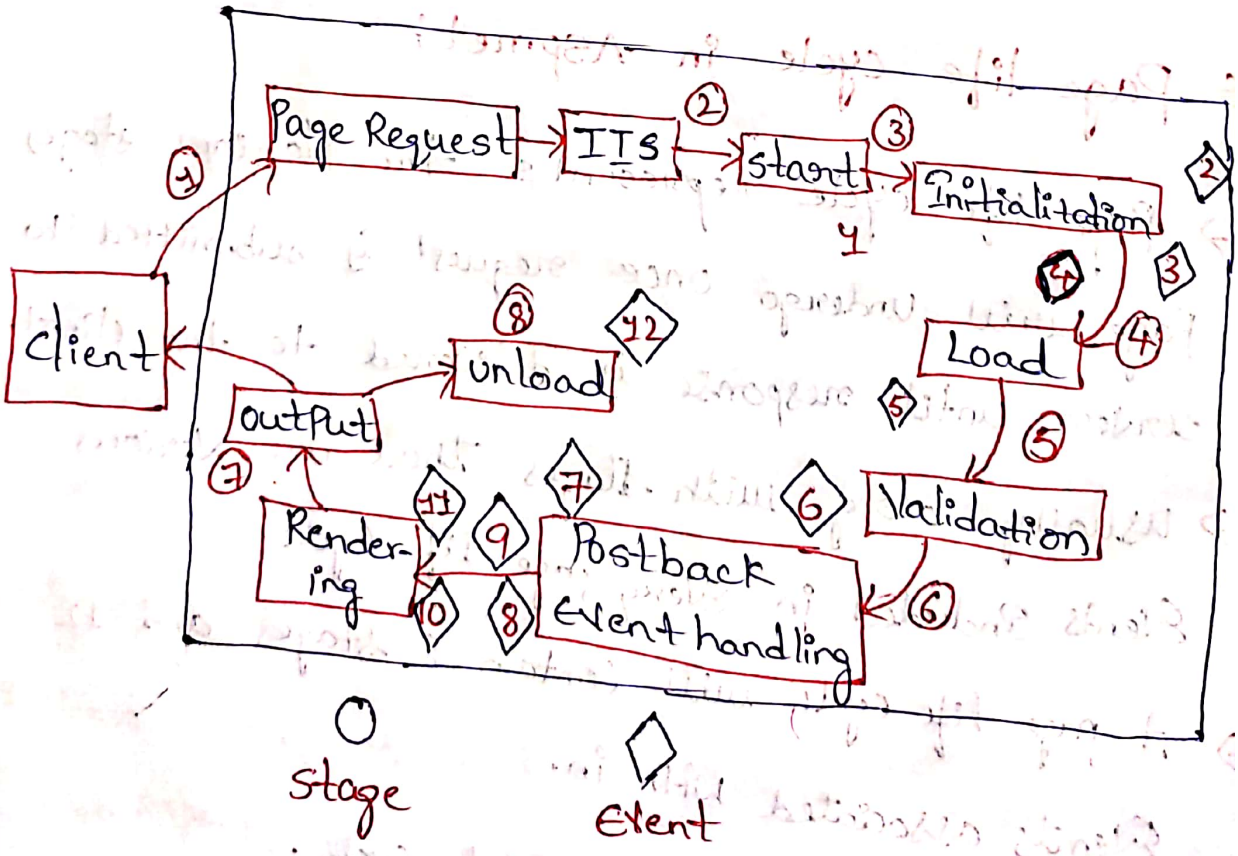
→ A page life cycle will Contain 8 stages and 12 Events associated with in.

→ Stages associated in page life cycle.

1. Page Request
2. Start
3. Initilitation
4. Load
5. Validation
6. Post back Event Handling
7. Rendering
8. Unload.

→ Events associated with page life cycle:-

1. Preinhit
2. Init
3. Initcomplete
4. Pre Load
5. Load
6. Control Event
7. Load complete
8. Pre Render
9. Pre Render complete
10. Save complete
11. Render
12. Unload



1) PreInit
2) Init
3) InitComplete
4) PreLoad
5) Load
6) Control Event
7) LoadCompelte
8) PreRender
9) PreRenderComplete
10) SaveComplete
11) Render
12) UnLoad

Stage 1:- Page Request:-

→ The page Request will begin when the page is requested by the client or user from the browser

→ When request reaches IIS, IIS will check whether cache is available to the page or Entire request should be reprocessed.

Stage:-2 Start:-

→ During this state page properties like

→ Request Response UI Culture Etc (or) set and also it will determine whether the page request is first request or post back request. using post back property.

Stage:-3 Initialization:-

During this stage complete page properties are Initialized, Every Controls unique Ids is available. But, other properties will not be set.

Stage:-4 Load:-

During this stage all properties values of Every control or set, if the request is post back request, all the properties values are recovered from the control state and ViewState.

5. Validation:

This will be executed in case of postback request. for every control Validate Method is executed and is valid property for the controls Individual Validation Controls is set and also for the page.

6. Post back Event Handling:

During this stage code available in the event using which the page request is submitted to the user will be executed.

7. Rendering:

→ Before Rendering the data of the control state and view state is saved then during Rendering Render Method is called for every control Individually and also for the page.

→ Render output is return to the output stream of the Response object for the page. Then output is delivered to client.

8. UNLoad:

Once output is delivered to the client unload stage will begin and during this stage Individual objects of the controls are destroyed.

and page object is also destroyed, any other cleanup will be performed.