State Management

UNIT - 3

State

- HTTP (Hyper Text Transfer Protocol) is a stateless protocol. When the client disconnects from the server, the ASP.Net engine discards the page objects.
- This way each web application can scale up to serve numerous requests simultaneously without running out of server memory.
- However, there need to be some technique to store the information between requests and to retrieve it when required.
- This information i.e., the current value of all the controls and variables for the current user in the current session is called the **State**.

State Management Techniques

- ASP.NET provides us with 2 ways to manage the state of an application.
- It is basically divided into the 2 categories:
  - Client Side State Management
  - Server Side State Management

Client Side State Management

- It is a way in which the information which is being added by the user or the information about the interaction happened between the user and the server is stored on the client's machine or in the page itself.
- The server resources (e.g. server's memory) is not at all utilized during the process. This management technique basically makes use of the following:
  - View State
  - Hidden Fields
  - Query String
  - Cookies

View State

- View State can be used to maintain the State at a page level. The term "Page Level" means that the information is being stored for a specific page and until that specific page is active (i.e. the page which is being currently viewed by the user).
- Once the user is re-directed or goes to some other page, the information stored in the View State gets lost.
- It basically makes use of a "Dictionary Object" to store data, which means that the information is stored in a key and value pair.
- It stores this information in a Hidden field on the page itself in a hashed format.
- View State can store a string value only of a specific length. If the length is exceeded then the excess information is stored in another hidden field.

View State

- You can set View State on/off for each control using EnableViewState property. By default, EnableViewState property will be set to true.
- View state information of all the controls on the page will be submitted to server on each post back.
- To reduce performance penalty, disable View State for all the controls for which you don't need state. (Data grid usually doesn't need to maintain state).
- You can also disable View State for the entire page by adding EnableViewState=false to @page directive
- // Add item to ViewState

  ViewState["myviewstate"] = myValue;
- //Reading items from ViewState
  Response.Write(ViewState["myviewstate"]);

View State

- Advantages
  - It is very simple to use.
  - Data is stored in hashed format and hence a layman won't be able to understand the value of the View State (It still can be hacked by Hackers, so to make it more secure we should try to store the value in an encrypted format.).
  - It is customizable
- Disadvantages
  - Information is not encrypted, so it can be easy for a Hacker to get its value.
  - Cannot be used to store sensitive data (eg: Passwords, Credit Card Pins, etc).
  - Might make a page heavy if lots of data is stored in View State.

Query strings

- Query strings are usually used to send information from one page to another page.
- They are passed along with URL in clear text.
- Now that cross page posting feature is back in asp.net 2.0, Query strings seem to be redundant.
- Most browsers impose a limit of 255 characters on URL length.
- We can only pass smaller amounts of data using query strings.
- Since Query strings are sent in clear text, we can also encrypt query values.
- Also, keep in mind that characters that are not valid in a URL must be encoded using Server.UrlEncode.

Query strings

- Let's assume that we have a list of products, and a hyperlink in the grid that goes to a product detail page.
- it would be an ideal use of the Query String to include the product ID in the Query String of the link to the product details page

  (for example, productdetails.aspx? productid=4).
- When product details page is being requested, the product information can be obtained by using the following codes:

```
string productid;
productid=Request.Params["productid"];
```

Query strings

- Advantages
  - Simple to Implement
- Disadvantages
  - Human Readable
  - Client browser limit on URL length
  - Cross paging functionality makes it redundant
  - Easily modified by end user

Cookies

- cookie is a small piece of text stored on user's computer.
- Usually, information is stored as name- value pairs. Cookies are used by websites to keep track of visitors.
- Every time a user visits a website, cookies are retrieved from user machine and help identify the user.
- Advantages
  - Very easy to use.
  - Stored on the client's machine, hence no server resources are utilized.
- Disadvantages
  - A user can disable cookies using browser settings.
  - Cookies are transmitted for each HTTP request/response causing overhead on bandwidth
  - Inappropriate for sensitive data

Cookies

- Let's see an example which makes use of cookies to customize web page.

```
if (Request.Cookies["UserId"] != null)
        lbMessage.text = " Welcome to our website " +
        Request.Cookies["UserId"].Value ;
else
        lbMessage.text = "Guest,welcome to our website!";
```

//If you want to store client's information use the below code Response.Cookies["UserId"].Value=username;

Server Side State Management

- It is another way which ASP.NET provides to store the user's specific information or the state of the application on the server machine.
- It completely makes use of server resources (the server's memory) to store information.
- This management technique basically makes use of the following:
  - Application State
  - Session State

Application State

- Application object is used to store data which is visible across entire application and shared across multiple user sessions.
- Data which needs to be persisted for entire life of application should be stored in application object.
- In classic ASP, application object is used to store connection strings. It's a great place to store data which changes infrequently.
- There are 3 events of the Application which are as follows
  Application_Start

  Application_Error

  Application_End
- we have to set the Page title in the Application Start event of the Global.asax file.

Application State

Code for setting value to the Application Object - "PageTitle" is the Key and "Welcome to State Management Application" is the value.

```
void Application_Start(object sender, EventArgs e)
{
    this.Application["PageTitle"] = "Welcome to State Management Application";
}
```

Code for reading value from the Application Object

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        this.Page.Title = Convert.ToString(this.Application["PageName"]);

    }
}
```

Session State

- When a user connects to an ASP.Net website, a new session object is created. When session state is turned on, a new session state object is created for each new request. This session state object becomes part of the context and it is available through the page.
- Session state is generally used for storing application data like inventory or supplier list, or a customer record or shopping cart. It can also keep information about the user and his preference and keep track of pending operations.
- Sessions are identified and tracked with a 120-bit SessionID, which is passed from client to server and back as cookie or a modified URL. The SessionID is globally unique and random.
- The session state object is created from the HttpSessionState class, which defines a collection of session state items.

Session State

The HttpSessionState class has the following properties:

Properties	Description
SessionID	The unique session identifier
Item(name)	The value of the session state item with the specified name. This is the default property of the HttpSessionState class
Count	The number of items in the session state collection
TimeOut	Gets and sets the amount of time, in minutes, allowed between requests before the session-state provider terminates the session.

The HttpSessionState class has the following methods:

Methods	Description
Add(name, value)	Adds an item to the session state collection
Clear	Removes all the items from session state collection
Remove(name)	Removes the specified item from the session state collection
RemoveAll	Removes all keys and values from the session-state collection.
RemoveAt	Deletes an item at a specified index from the session-state collection.

Session State

Code to write values into a Session

```
protected void btnSubmit_Click(Object sender, EventArgs e)
{
    Session["Username"] = txtUsername.Text.Trim();
    Response.Redirect("Default.aspx");
}
```

Code to read values from Session

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        lblUsername.Text = Convert.ToString(Session["Username"]);
    }
}
```