- Introduction to ASP applications,
- State maintenance in ASP applications,
- Controlling access and monitoring,
- Planning application,
- Develop a sample project using ASP.

ASP stands for **A**ctive **S**erver **P**ages
ASP is a development framework for building web pages
ASP supports many different development models:

- Classic ASP
- ASP.NET Web Forms
- ASP.NET MVC
- ASP.NET Web Pages
- ASP.NET API
- ASP.NET Core

The ASP Technology

- ASP and ASP.NET are server side technologies.
- Both technologies enable computer code to be executed by an Internet server.
- When a browser requests an ASP or ASP.NET file, the ASP engine reads the file, executes any code in the file, and returns the result to the browser.

Classic ASP - Active Server Pages

- ASP (aka Classic ASP) was introduced in 1998 as Microsoft's first server side scripting language.
- Classic ASP pages have the file extension **.asp** and are normally written in VBScript.

ASP.NET

ASP.NET was released in 2002 as a successor to Classic ASP.

- ASP.NET pages have the extension **.aspx** and are normally written in C# (C sharp).
- ASP.NET 4.6 is the latest official version of ASP.NET.
- ASP.NET 5 was expected to be an important redesign of ASP.NET.
- However, the development of ASP.NET 5 was stopped in favour of ASP.NET Core.

ASP.NET Web Pages

- ASP.NET Web Pages is an SPA application model (Single Page Application).
- The SPA model is quite similar to PHP and Classic ASP.
- ASP.NET Web Pages is being merged into the new ASP.NET Core.

ASP.NET MVC

- ASP.NET MVC is an MVC application model (Model-View-Controller).
- ASP.NET MVC is being merged into the new ASP.NET Core.
- ASP.NET MVC is not covered in this tutorial.

ASP.NET Web API

- ASP.NET API is an API application model (Application Programming Interface).
- ASP.NET API is being merged into the new ASP.NET Core.
- ASP.NET API is not covered in this tutorial.

ASP.NET Web Forms

- ASP.NET Web Forms is an event driven application model.
- ASP.NET Web Forms is **not** a part of the new ASP.NET Core.
- ASP.NET Web Forms is **not** covered in this tutorial.

ASP.NET Core

- ASP.NET Core was released in 2016.
- ASP.NET Core merges ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages into one application framework.

- ASP.NET Core is not covered in this tutorial.

What are the steps requried for desining an ASP based application?

The steps required for designing an ASP based application are listed below:

1. Creating the design requirements
2. Creating the user interface
3. Writing source code for the ASP .NET page
4. Testing the ASP.NET web page

Step 1: Creating the design requirements

While developing software, based on ASP application we need to decide, what is the purpose of developing it along with its features and functionalities. For this, we should list out the requirements and features of software like

- Resources consumption
- Information about users
- Planning activities
- Software feasibility

Step 2: Creating the user interface

The next step after collection of design requirements is creating user interface. User interface (UI) maintains the user interaction with the application. Inputs and outputs from the user are synchronized in this phase. UI is the HTML portion of ASP.NET web page. It includes adding needed web controls to the ASP.NET web page. Web controls like textbox, button, checkbox etc can be used for inputs and label can be used for output. Validation technologies can also be applied in this phase.

Step 3: Writing the source code for ASP.NET web page

After completion of HTML portion of our ASP.NET web page, source code will be next step. Source code will read the user's inputs, performs necessary computation and provides output. Source code includes reading values from web controls like textbox or when some event triggers, displaying outputs, page loading etc. Source code can be written in any scripting language like JScript, VBScript, JavaScript etc.

Step 4: Testing the ASP.NET web page

Testing is done in order to check whether developed web page works as per need of user or not. It is done simply by taking some inputs and visualizing the outputs. Then output is matched with user requirements and deviation is calculated. Further deviation is reduced by reanalysing the steps

State Management in ASP

State management is the process by which you maintain state and page information over multiple requests for the same or different pages.

Types of State Management

1. Client – Side State Management

This stores information on the client's computer by embedding the information into a Web page, a uniform resource locator(url), or a cookie. The techniques available to store the state information at the client end are listed down below:

a) **View State –** Asp.Net uses View State to track the values in the Controls. You can add custom values to the view state. It is used by the Asp.net page framework to automatically save the values of the page and of each control just prior to rendering to the page. When the page is posted, one of the first tasks performed by page processing is to restore view state.

b) **Control State –** If you create a custom control that requires view state to work properly, you should use control state to ensure other developers don't break your control by disabling view state.

c) **Hidden fields –** Like view state, hidden fields store data in an HTML form without displaying it in the user's browser. The data is available only when the form is processed.

d) **Cookies –** Cookies store a value in the user's browser that the browser sends with every page request to the same server. Cookies are the best way to store state data that must be available for multiple Web pages on a web site.

How to Create a Cookie?

The "Response.Cookies" command is used to create cookies.

Note: The Response.Cookies command must appear BEFORE the <html> tag.

In the example below, we will create a cookie named "firstname" and assign the value "Alex" to it:

```
<%
Response.Cookies("firstname")="Alex"
%>
```

How to Retrieve a Cookie Value?

The "Request.Cookies" command is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "firstname" and display it on a page:

```
<%
fname=Request.Cookies("firstname")
response.write("Firstname=" & fname)
%>
```

Output: Firstname=Alex

e) **Query Strings -** Query strings store values in the URL that are visible to the user. Use query strings when you want a user to be able to e-mail or instant message state data with a URL.

2. Server – Side State Management

a) **Application State** - Application State information is available to all pages, regardless of which user requests a page.

b) **Session State –** Session State information is available to all pages opened by a user during a single visit.

Both application state and session state information is lost when the application restarts. To persist user data between application restarts, you can store it using profile properties.

Advantages

Client – Side State Management:

- **Better Scalability:** With server-side state management, each client that connects to the Web server consumes memory on the Web server. If a Web site has hundreds or thousands of simultaneous users, the memory consumed by storing state management information can become a limiting factor. Pushing this burden to the clients removes that potential bottleneck.
- **Supports multiple Web servers**: With client-side state management, you can distribute incoming requests across multiple Web servers with no changes to your application because the client provides all the information the Web server needs to process the request. With server-side state

management, if a client switches servers in the middle of the session, the new server does not necessarily have access to the client's state information. You can use multiple servers with server-side state management, but you need either intelligent load-balancing (to always forward requests from a client to the same server) or centralized state management (where state is stored in a central database that all Web servers access).

Server – Side State Management:

- **Better security**: Client-side state management information can be captured (either in transit or while it is stored on the client) or maliciously modified. Therefore, you should never use client-side state management to store confidential information, such as a password, authorization level, or authentication status.
- **Reduced bandwidth**: If you store large amounts of state management information, sending that information back and forth to the client can increase bandwidth utilization and page load times, potentially increasing your costs and reducing scalability. The increased bandwidth usage affects mobile clients most of all, because they often have very slow connections. Instead, you should store large amounts of state management data (say, more than 1 KB) on the server.

Basic file structure of ASP

After installing IIS the web server let us start our first ASP page. Here we assume that ASP engine is working perfectly and server root is set to proper directory were we will keep our fist asp page. Here if the directory path is E:\myfiles\ and our first ASP file name is test.asp then the path becomes E:\myfiles\test.asp. To open this file in web browser mode we have to open this as

```
https://localhost/test.asp
```

or

```
https://127.0.0.1/test.asp
```

Note that in IIS we have set the default directory to E:\myfiles\

Now with this settings let us go for our fist file test.asp

We will start with our ASP code beginning mark <% and end the code with %> . This way we will tell our ASP engine to execute the ASP part of the code within the <% and %> symbol and send the other codes as it is to the browser. So we will start with telling the server that the code we are going to use is VB Script. ASP is not a script or language , it is a technology server uses. So we have to tell the server which language we are using. We can also use JavaScript as server side script. But mostly VBScript is used for server side scripting requirements and we will also use this. Here is the way we will tell to use VBScript language for processing.

```
<%@ Language=VBScript %>
```

In the next line we will tell that all the variables used here are to be declared before using. This has some advantage as we move to words developing complex applications using ASP. This is a good practice to follow. So here is our next line

```
<% Option Explicit %>
```

After this we will keep our html meta tags and then we will go to the body section of the ASP page. We will try to print the message "Hello World" to the screen. Here we will store this message string in a variable and let us name the variable as msg. Please note that we have told ASP engine that all the variables we use are to be declared before, this we have told in the second line of our code by saying <% Option Explicit %> . So we will declare the variable by using the command Dim. Here is the example

Dim msg
msg="Hello World"

Now let us print the message or the variable to the screen by using Response.Write object.

```
Response.Write msg
```

That's our entire first ASP example. Here is the complete code.

```
<%@ Language=VBScript %>
<% Option Explicit %>

<html>
   <head>
            <title>(Type a title for your page here)</title>
    </head>

     <body >
       <%
           Dim msg
            msg="Hello World"
            Response.Write msg
        %>

      </body>
</html>
```