Microsoft® .NET | ASP.NET

Developing Web Apps with ASP.net WebForms 4.5

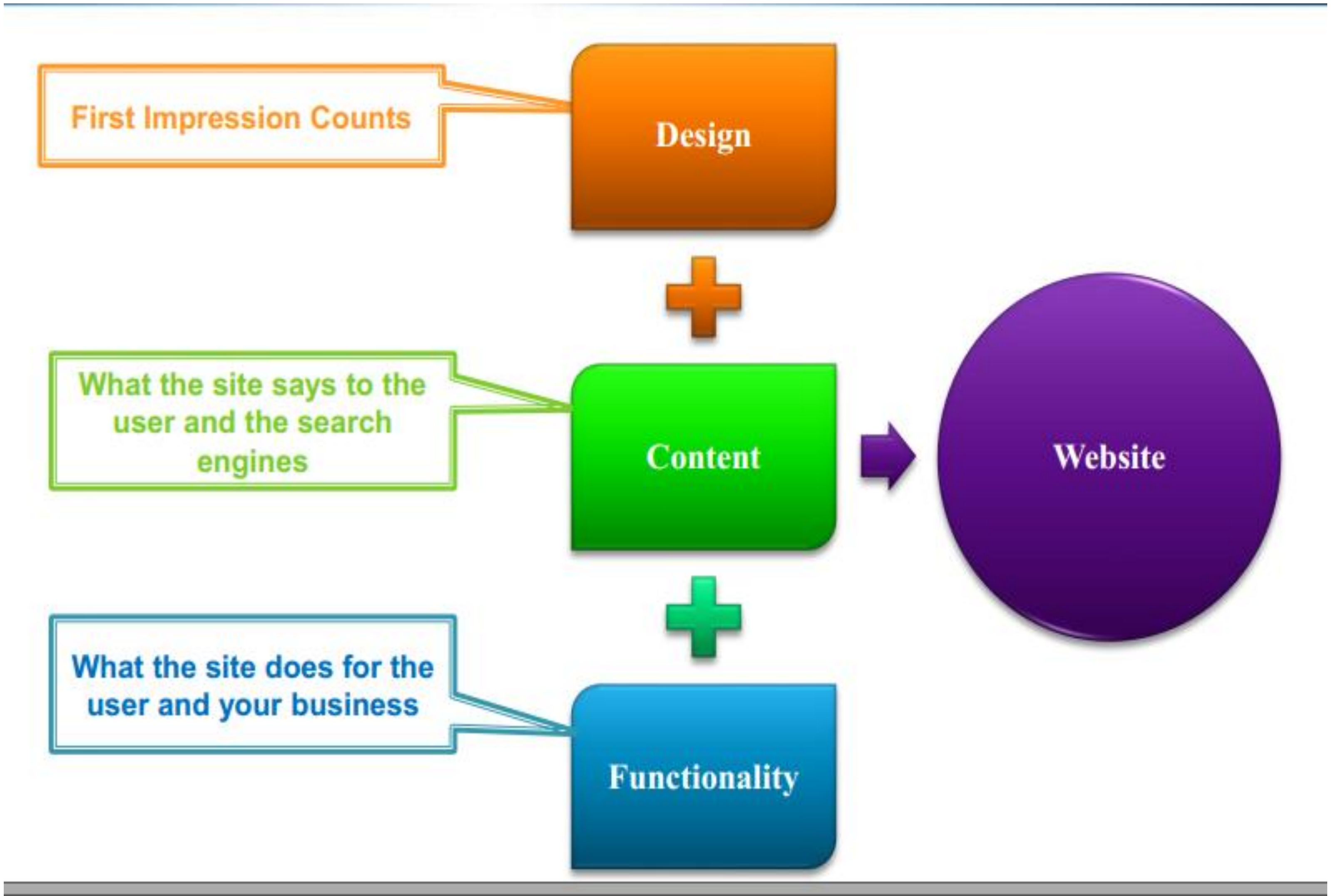
Christen Zarif Foad

Early Web Development

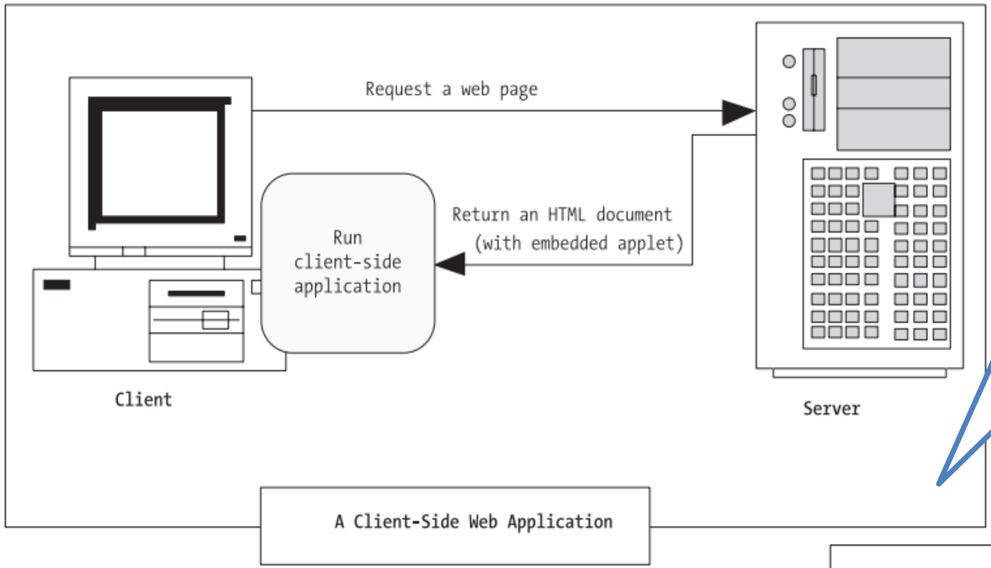
Passed through 3 stages:

1. **Static pages (Already-made)**
    - **HTML pages.**
    - **Describes layout.**
    - **Interactivity achieved mainly using hyperlinks.**
2. **Dynamic pages (Interactive)**
    - **HTML & scripts (ex: javascript).**
    - **Pages contained client-side scripts.**
    - **Can be used to validate user input.**
3. **Active pages**
    - **The need to post requests and parameters to the web server appeared.**
    - **Pages built upon request, based on that specific parameters.**
    - **Ex: ASP & ASP.NET**

Website Creation


First Impression Counts
Design
What the site says to the user and the search engines
Content
Website
What the site does for the user and your business
Functionality

Client Side Programming vs. Server Side Programming

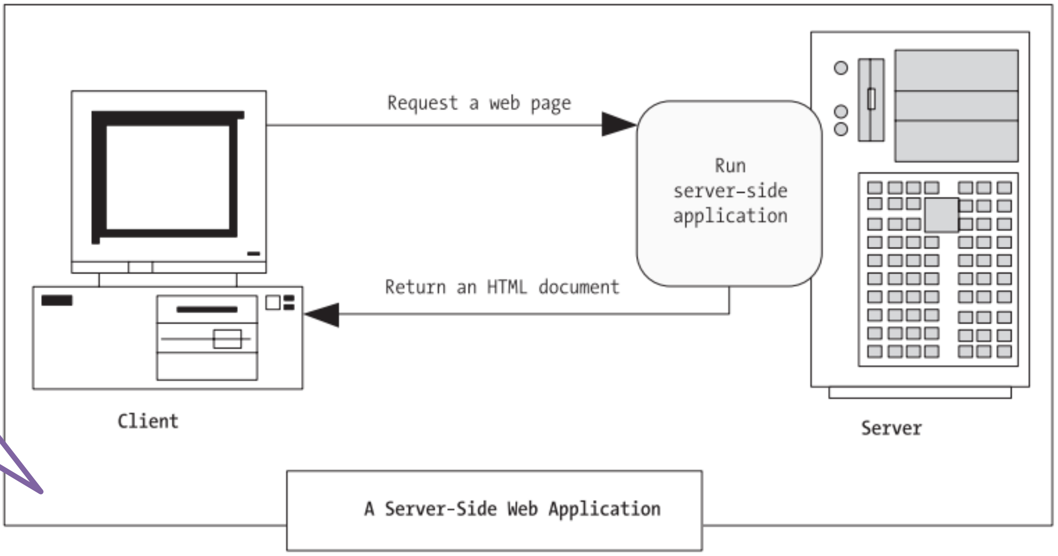

Request a web page
Return an HTML document (with embedded applet)
Run client-side application
Client
Server
A Client-Side Web Application

Client Side Programming

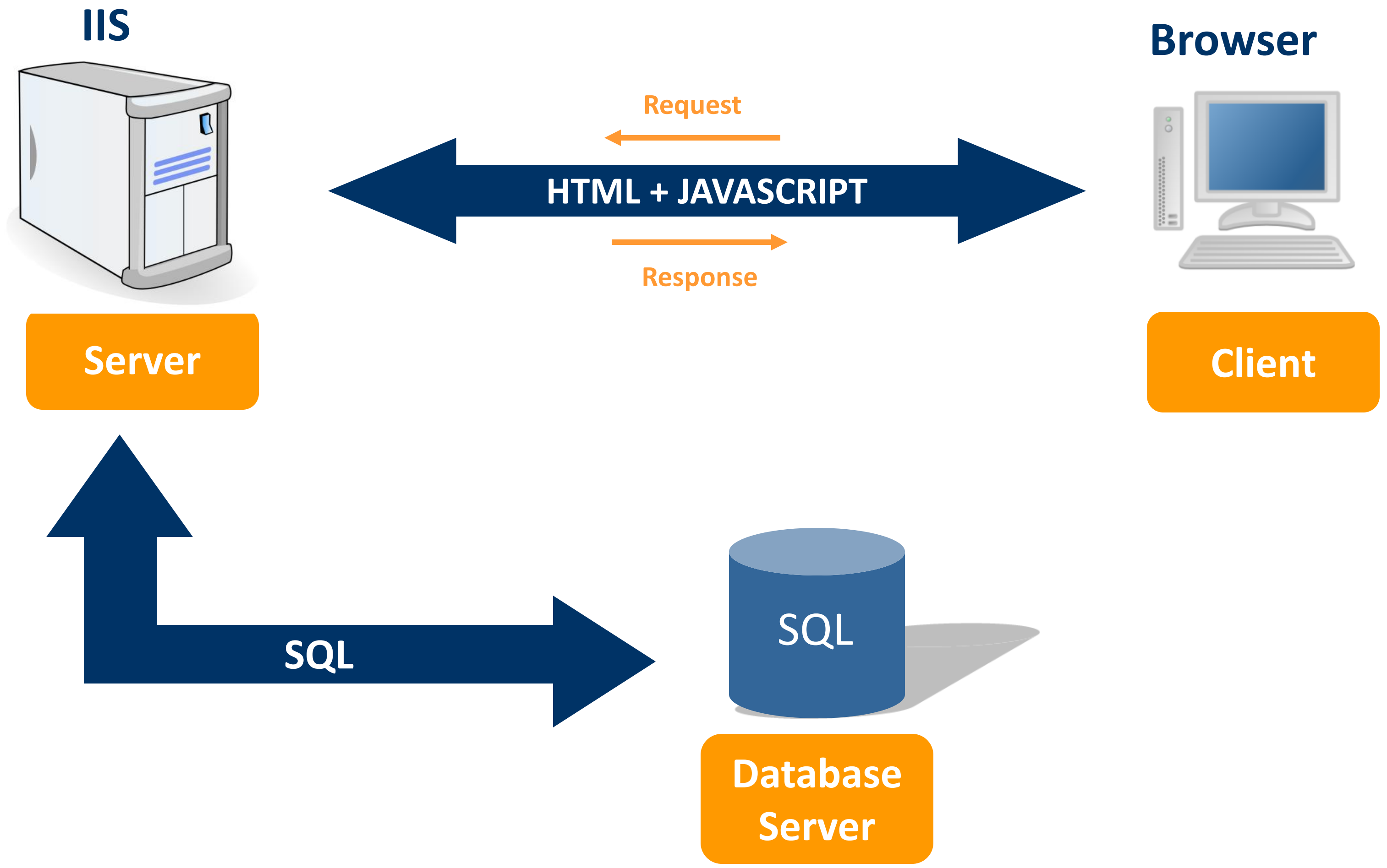
Enhancing web pages by embedding miniature applets built with JavaScript, ActiveX, Java, and Flash into web pages. These client-side technologies don't involve any server processing. Instead, the complete application is downloaded to the client browser, which executes it locally.

Server Side Programming

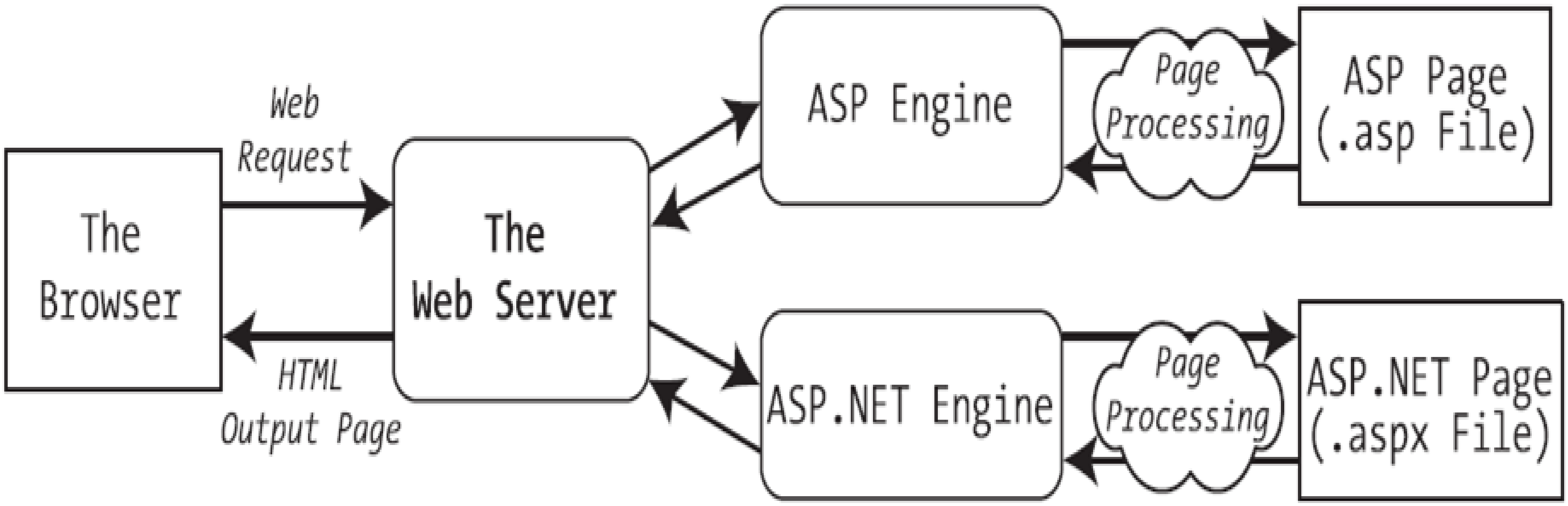
All code executes on the server. When the code is finished executing, the user receives an HTML Page



Web page Round-Trip



How IIS Handles an asp file request



Overview of Classic ASP

- **ASP stands for Active Server Pages.**
- **Microsoft's previous server side scripting technology.**
- **ASP Features:**
  1. **Code written inline with HTML (Spaghetti code!)**
  2. **Interpreted**
  3. **Does not need any framework to run**
  4. **Not Object Oriented**
     - **Lack of code separation**
     - **Lack of code reusability**
  5. **Lack of debugging support**

Overview of Classic ASP (cont.)

When a client browser requests an asp page:

1. **IIS passes the request to the ASP engine.**
2. **The ASP engine reads the ASP file, line by line, and executes the scripts in the file (Interpreted).**
3. **Finally, the ASP file is returned to the browser as plain HTML**

ASP.Net

- **ASP.NET is Not ASP.**
- **ASP.NET is the next generation ASP, but it's not an upgraded version of ASP.**
- **ASP.NET is an entirely new technology for server-side scripting.**

ASP.Net vs. ASP

Classic ASP

- Script Based Language
- Executable Code must be integrated with HTML
- Does not support classes

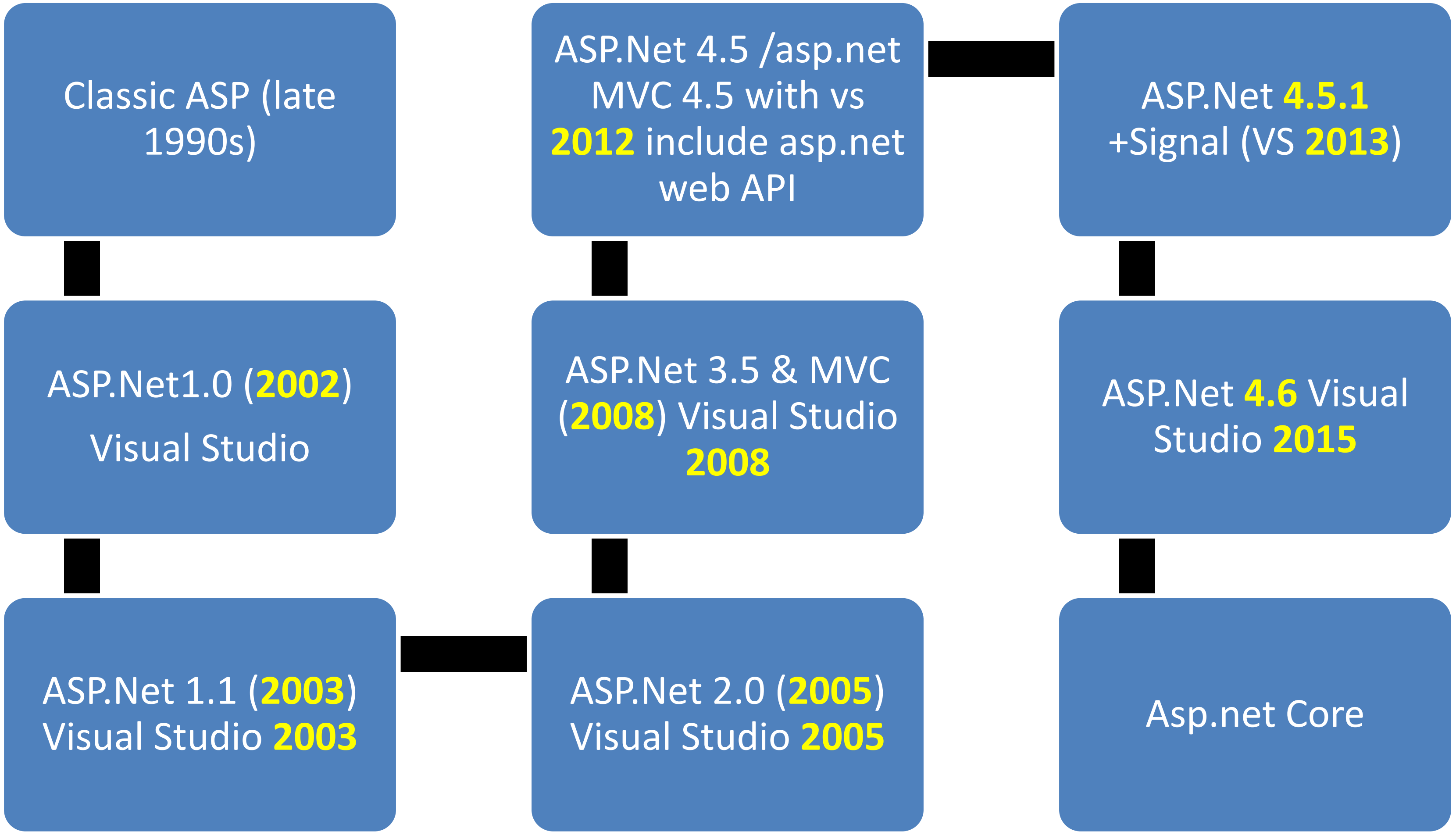
ASP.NET

- Compiled Language
- Code can separated out into separate layers
- Natively based in OOP

ASP.Net

- ASP.NET is an open source web framework for building modern web apps and services with .NET.
- ASP.NET creates websites based on HTML5, CSS, and JavaScript that are simple, fast, and can scale to millions of users.

Brief of History


Classic ASP (late 1990s)
ASP.Net1.0 (**2002**) Visual Studio
ASP.Net 1.1 (**2003**) Visual Studio **2003**
ASP.Net 2.0 (**2005**) Visual Studio **2005**
ASP.Net 3.5 & MVC (**2008**) Visual Studio **2008**
ASP.Net 4.5 /asp.net MVC 4.5 with vs **2012** include asp.net web API
ASP.Net **4.5.1** +Signal (VS **2013**)
ASP.Net **4.6** Visual Studio **2015**
Asp.net Core

ASP.NET Core

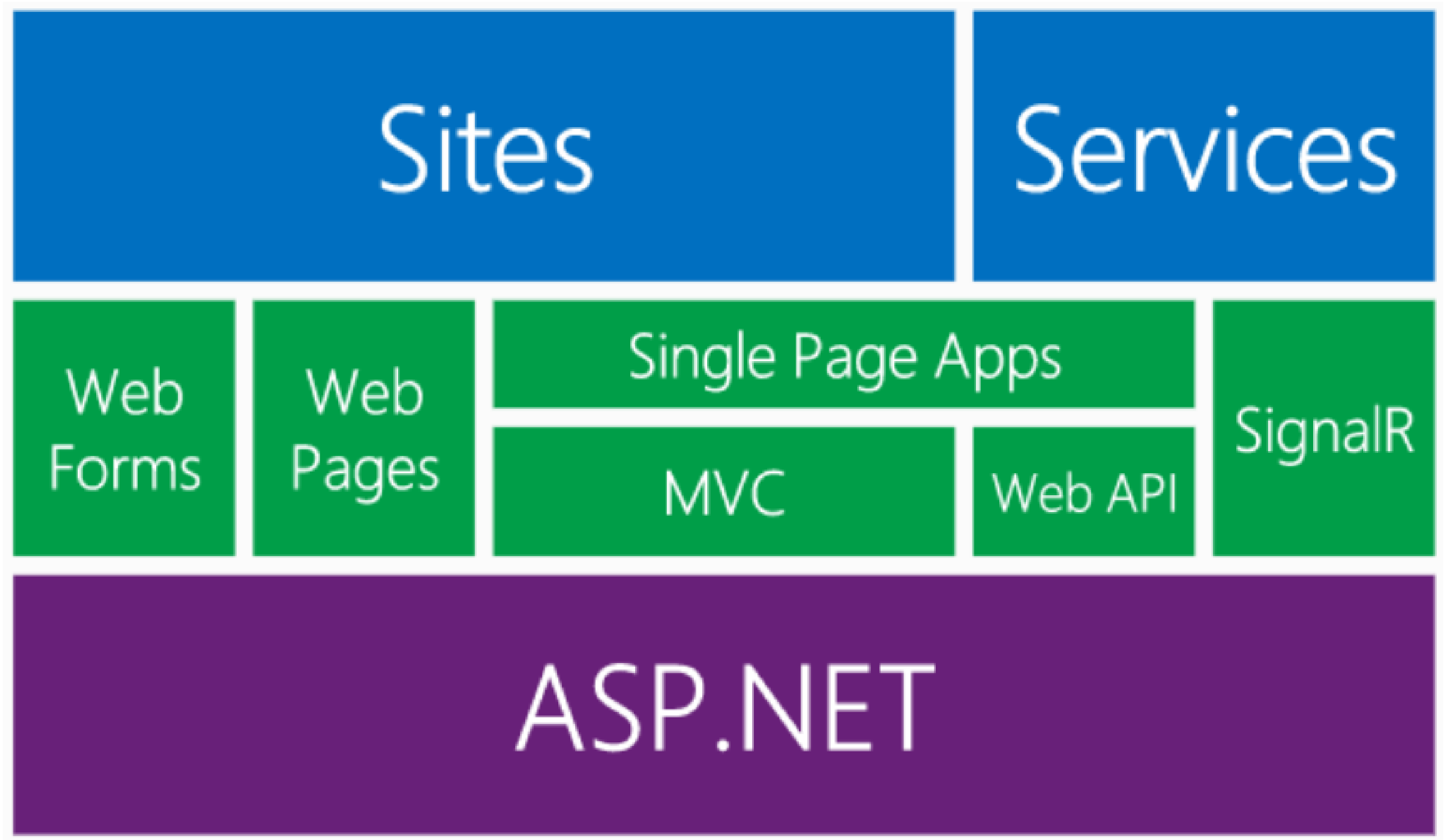
- is a significant redesign of ASP.NET
- ASP.NET Core is a new open-source and cross-platform framework for building modern cloud based internet connected applications
- ASP.NET Core apps can run on .NET Core or on the full .NET Framework.
- It was architected to provide an optimized development framework for apps that are deployed to the cloud or run on-premises.
- You can develop and run your ASP.NET Core apps cross-platform on Windows, Mac and Linux.
- ASP.NET Core is open source at GitHub.
- https://docs.microsoft.com/en-us/aspnet/core/

Asp.net Vs Asp.net Core

ASP.NET	**ASP.NET Core**
ASP.NET is a mature web platform that provides all the services that you require to build **enterprise-class** server-based web applications using .NET on Windows.	ASP.NET Core is a new open-source and cross-platform .NET framework for building **modern cloud-based web applications** on Windows, Mac, or Linux.
Benefits	
Build for Windows Use Web Forms, SignalR, MVC, or Web Pages One version per machine Develop with Visual Studio using C#, VB or F# Mature platform High performance	Build for Windows, Mac, or Linux Use MVC, or Web API Multiple versions per machine Develop with Visual Studio or Visual Studio Code using C# New platform Ultra performance

One ASP.Net

- One ASP.net : A framework for us all



Building Websites in ASP.NET

- ASP.NET offers three frameworks for creating web applications: Web Forms, ASP.NET MVC, and ASP.NET Web Pages.
- Each framework targets a different development style.
- All three frameworks will be supported, updated, and improved in future releases of ASP.NET.
- https://www.asp.net/aspnet/overview/making-websites-with-aspnet/making-websites-with-aspnet

Compare

- The one you choose depends on a combination of your programming assets (knowledge, skills, and development experience)

	If you have experi ence in	Development Style	Expertise
Web Pages	Classic ASP, PHP	HTML markup and your code together in the same file	New, Mid-Level
Web Forms	Win Forms, WPF, .NET	Rapid development using a rich library of controls that encapsulate HTML markup	Mid-Level, Advanced RAD
MVC	Ruby on Rails, .NET	Full control over HTML markup, code and markup separated, and easy to write tests. The best choice for mobile and single-page applications (SPA).	Mid-Level, Advanced

ASP.NET Web Pages

- ASP.NET Web Pages and the Razor syntax provide a fast, approachable, and lightweight way to combine server code with HTML to create dynamic web content. Connect to databases, add video, link to social networking sites, and include many more features that help you create beautiful sites that conform to the latest web standards.
- https://www.asp.net/web-pages

Asp.net MVC

- ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development. ASP.NET MVC includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards.
- https://www.asp.net/mvc

Asp.net Web Forms

- With ASP.NET Web Forms, you can build dynamic websites using a familiar drag-and-drop, event-driven model.
- A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.
- https://www.asp.net/web-forms

General Notes

- All three ASP.NET frameworks are based on the .NET Framework and share core functionality of .NET and of ASP.NET. For example,
  - all three frameworks offer a login security model based around membership,
  - all three share the same facilities for managing requests,
  - handling sessions, and so on that are part of the core ASP.NET functionality.

https://www.asp.net/web-forms/what-is-web-forms

Microsoft
ASP.net
Webforms

Asp.net Web Forms

- ASP.NET Web Forms is a part of the ASP.NET web application framework and is included with Visual Studio.
- It is one of the four programming models you can use to create ASP.NET web applications, the others are ASP.NET MVC, ASP.NET Web Pages, and ASP.NET Single Page Applications.

ASP.NET Web Forms Helps You Overcome Challenges

- **Client side Limitation:**
  - **Implementing a rich Web user interface**
  - **Separation of client and server**
    - In a Web application, the client (browser) and server are different programs often running on different computers (and even on different operating systems).
  - **Stateless execution**
    - When a Web server receives a request for a page, it finds the page, processes it, sends it to the browser, and then **discards** all page information. If the user requests the same page again, the server repeats the entire sequence, reprocessing the page from scratch.
  - **Unknown client capabilities(Cross-browser compatibility)**
    - users using different browsers.
  - **Complications with data access**
    - Reading from and writing to a data source in traditional Web applications can be complicated and resource-intensive.

ASP.NET Web Forms Helps You Overcome Challenges(Con.)

- **Client side Limitation:**
  - **Isolation:**
    - **Client-side code can't access server-side resources. For example, a client-side application has no easy way to read a file or interact with a database on the server.**
  - **Security:**
    - **End users can view client-side code, and could often tamper with it.**
  - **Thin clients:**
    - **Web-enabled devices can communicate with web servers, but they don't support all the features of a traditional browser, thin clients might not support client-side features such as JavaScript and Flash (Network)**

The Seven Keywords about ASP.NET

#1:
- ASP.NET Is Integrated with the .NET Framework

#2:
- ASP.NET Is Compiled, Not Interpreted

#3:
- ASP.NET Is Multilanguage

#4:
- ASP.NET Is Hosted by the Common Language Runtime

#5:
- ASP.NET Is Object-Oriented

#6:
- ASP.NET Supports all Browsers

#7:
- ASP.NET Is Easy to Deploy and Configure

ASP.NET Is Hosted by the CLR

ASP.NET engine runs inside the runtime environment of the CLR; which brings the following benefits:

1. Automatic memory management and garbage collection
2. Type safety
3. Structured error handling
4. Multithreading

ASP.NET Is Easy to Deploy and Configure

- An ASP.NET application is relatively simple as every installation of the .NET Framework provides the same core classes.
- Simply copy all the files to a virtual directory on a production server.
- ASP.NET makes the deployment process easier by minimizing the dependence on settings in IIS and storing the settings in a dedicated web.config file.

- Hidden field

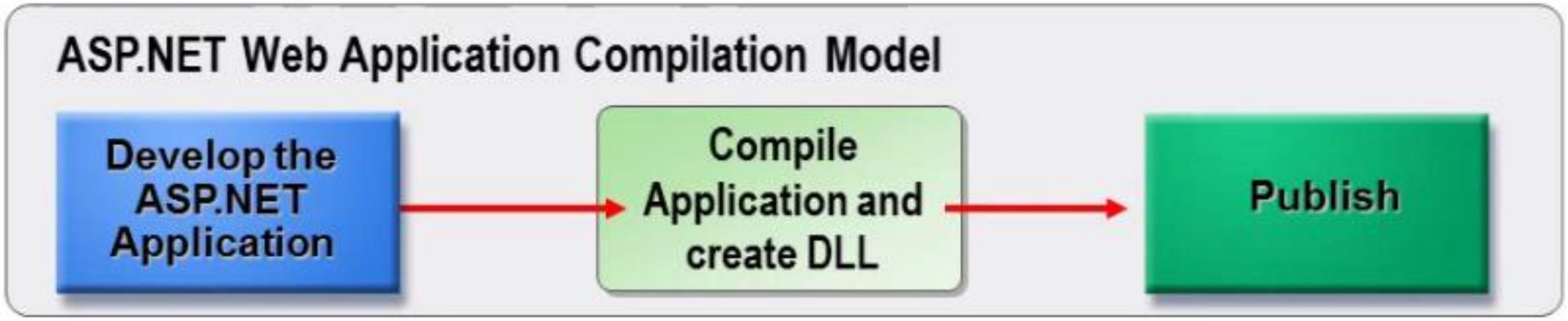
Test Demo

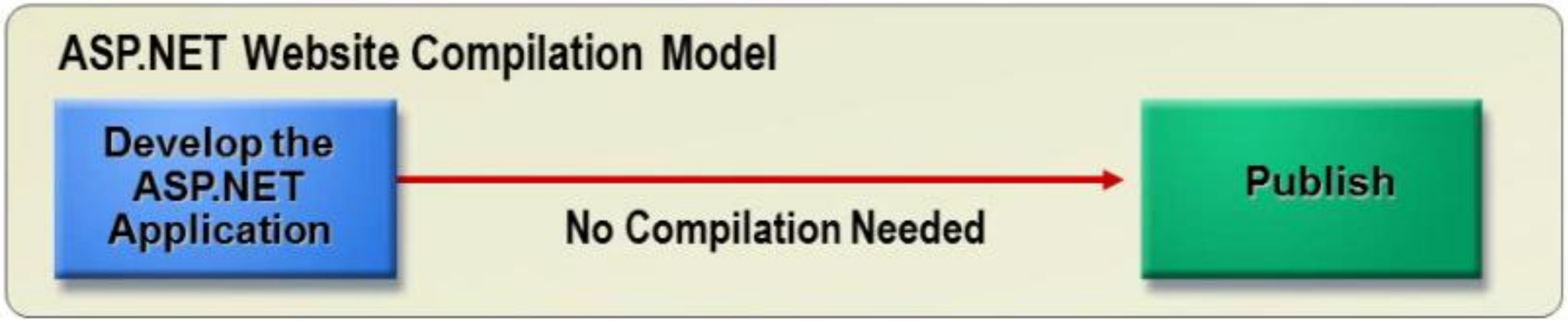
Creating ASP.NET Projects

- **ASP.NET supports two different types of projects:**
  - Web Site Project

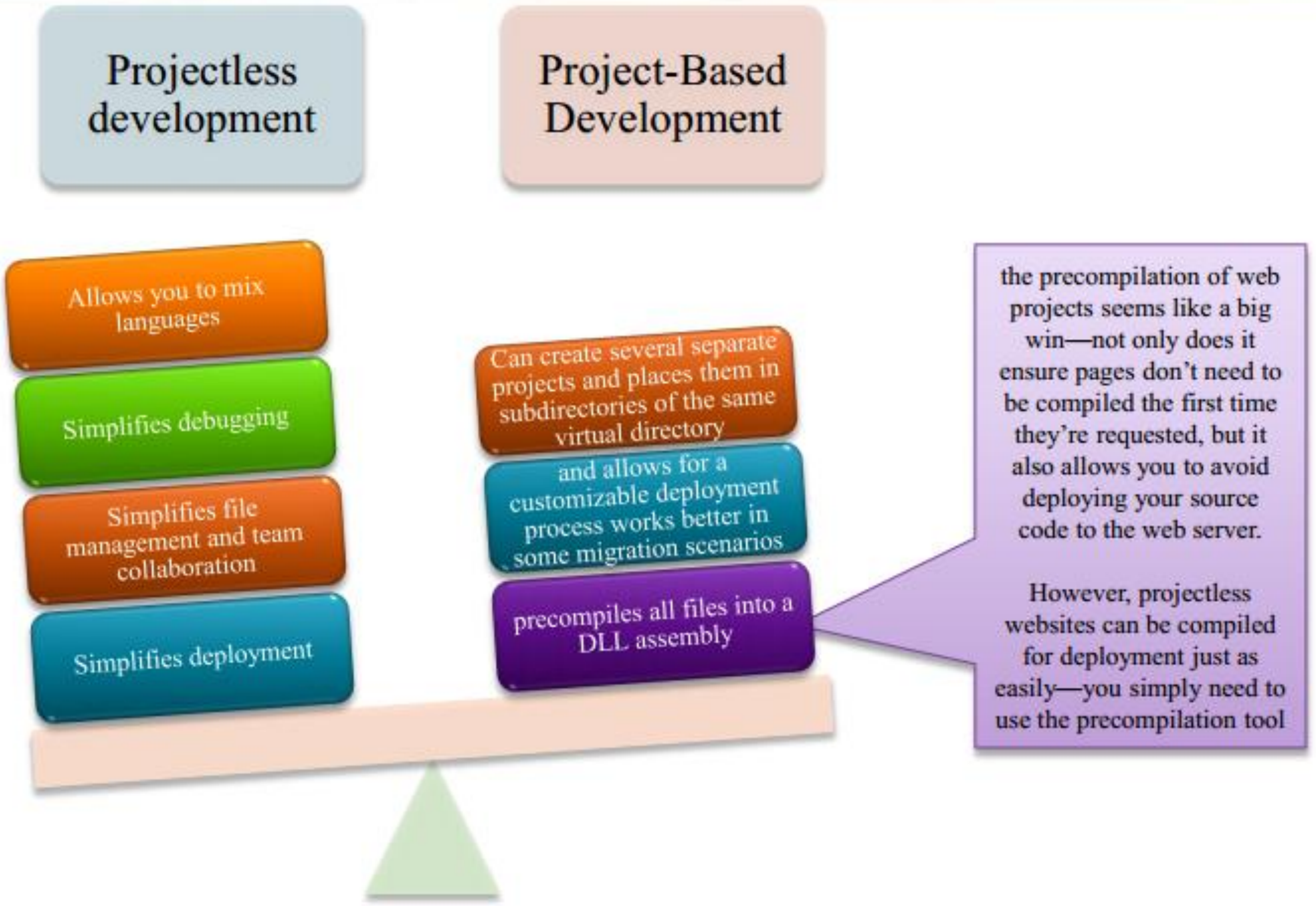
  - Web Application Project

Compiling ASP.NET Websites


ASP.NET Web Application Compilation Model
Develop the ASP.NET Application
Compile Application and create DLL
Publish


ASP.NET Website Compilation Model
Develop the ASP.NET Application
No Compilation Needed
Publish

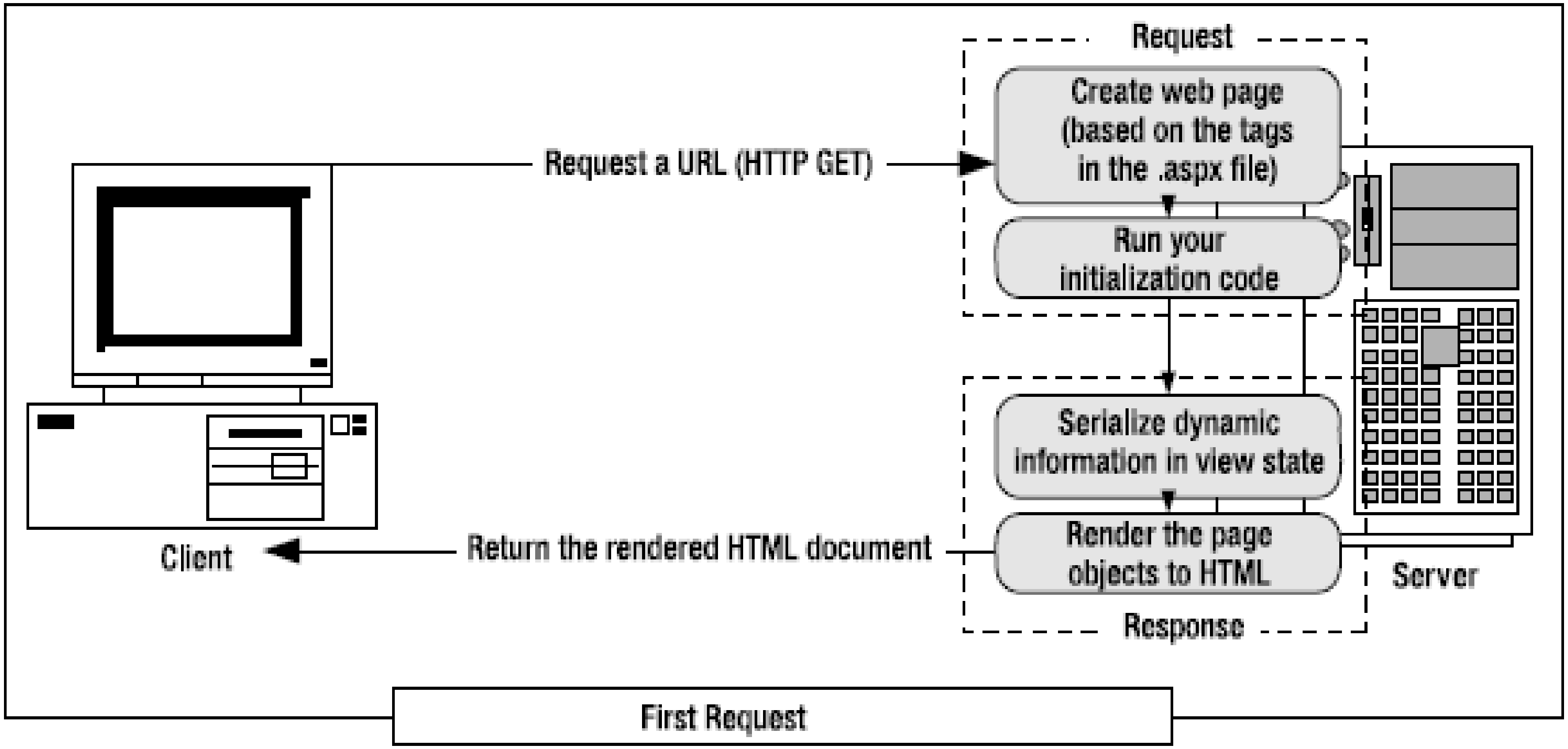
Creating Asp.NET web Projects


Projectless development
Project-Based Development
Allows you to mix languages
Simplifies debugging
Simplifies file management and team collaboration
Simplifies deployment
Can create several separate projects and places them in subdirectories of the same virtual directory
and allows for a customizable deployment process works better in some migration scenarios
precompiles all files into a DLL assembly

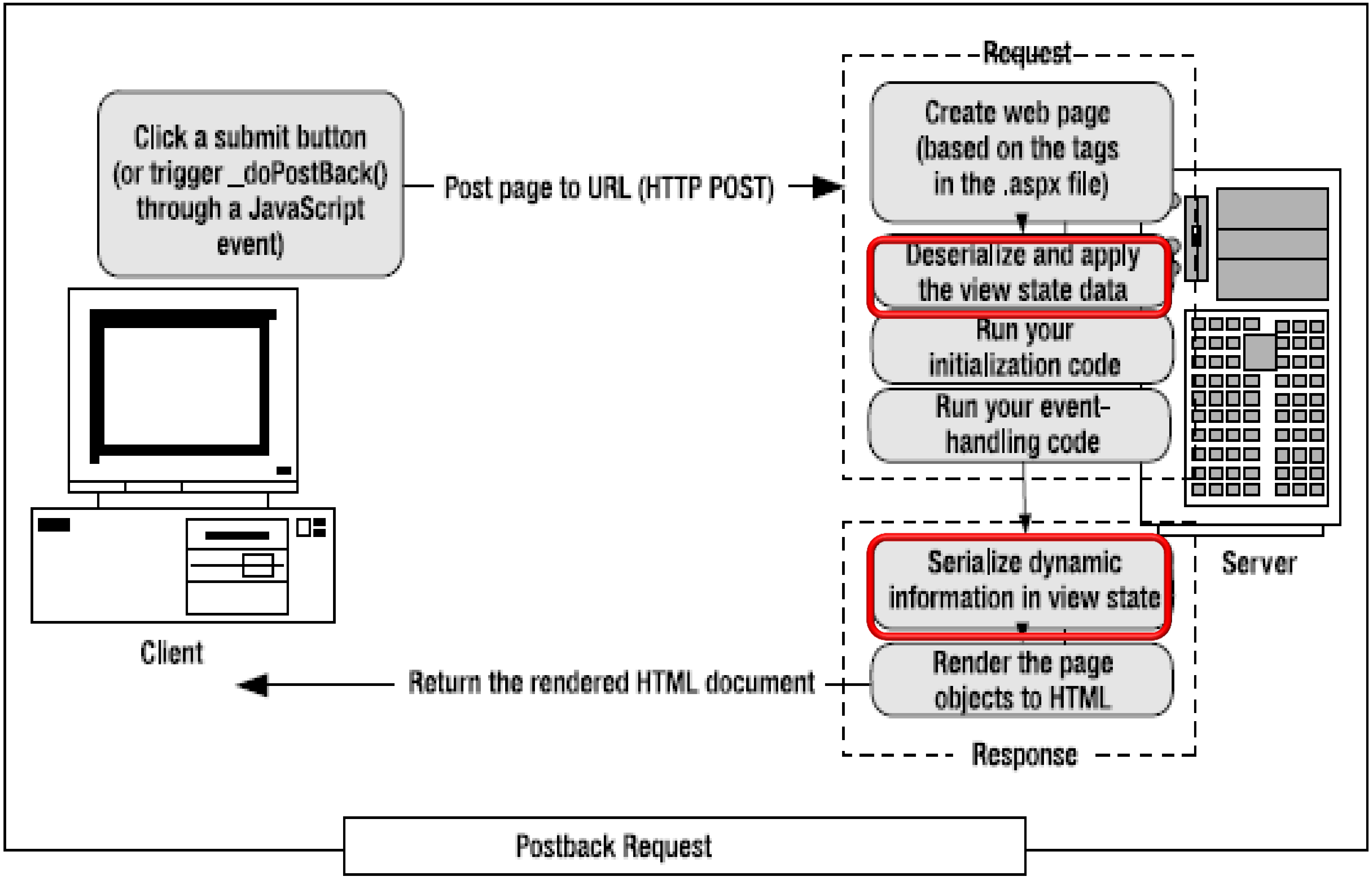
the precompilation of web projects seems like a big win—not only does it ensure pages don't need to be compiled the first time they're requested, but it also allows you to avoid deploying your source code to the web server.

However, projectless websites can be compiled for deployment just as easily—you simply need to use the precompilation tool

Web Page Life cycle --First Request


Request
Create web page (based on the tags in the .aspx file)
Request a URL (HTTP GET)
Run your initialization code
Serialize dynamic information in view state
Render the page objects to HTML
Return the rendered HTML document
Client
Server
Response
First Request

Web Page Life cycle --PostBack Request


Click a submit button (or trigger _doPostBack() through a JavaScript event)
Post page to URL (HTTP POST)
Request
Create web page (based on the tags in the .aspx file)
Deserialize and apply the view state data
Run your initialization code
Run your event-handling code
Serialize dynamic information in view state
Render the page objects to HTML
Response
Server
Client
Return the rendered HTML document
Postback Request

System.Web.UI Namespace

- The System.Web.UI namespace provides classes and interfaces that enable you to create ASP.NET server controls and ASP.NET Web pages for the user interface of your ASP.NET Web applications
- This namespace includes the Control class, which provides a common set of functionality for all server controls, which includes HTML server controls, Web server controls, and user controls

System.Web.UI Namespace (Con.)

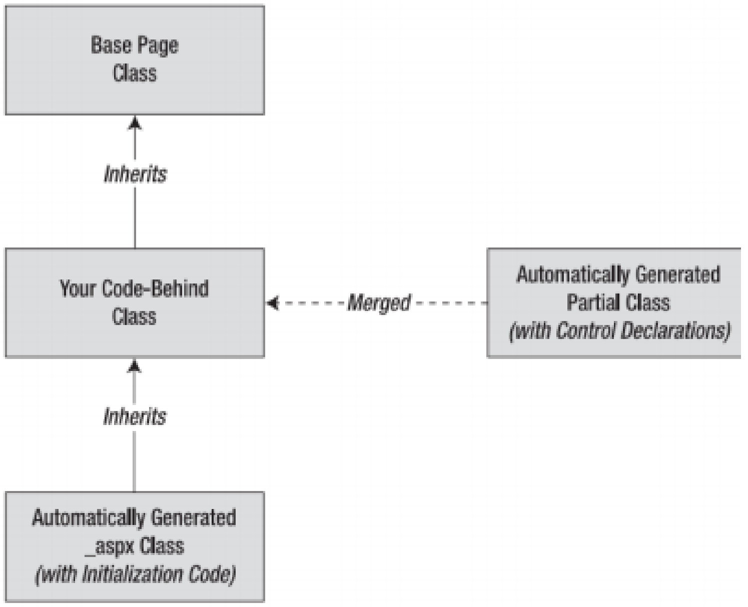
- It also includes the Page class. This class is generated automatically whenever a request is made for an .aspx file in an ASP.NET Web application. You can inherit from both of these classes.
- https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx

ASP.NET Page Class

- ASP.NET creates an instance of a class that represents your page. That class is composed not only of the code that you wrote for the page, but also code that is generated by ASP.NET.
- The Page class is associated with files that have an .aspx extension. These files are compiled at run time as Page objects and cached in server memory.

Default Page

- The page class from the .Net Class Library **define the basic functionality that allows a web page to host other control** , render itself to html
- Your code-behind class inherits from the Page class
- Upon compiling , asp.net merges some extra code into your class defining all the controls on your page as protected variables


Base Page Class
Inherits
Your Code-Behind Class
Merged
Automatically Generated Partial Class (with Control Declarations)
Inherits
Automatically Generated _aspx Class (with Initialization Code)

- The asp.net compiler create one more class to represents the actual .aspx page. This class inherits from your custom code –behind class

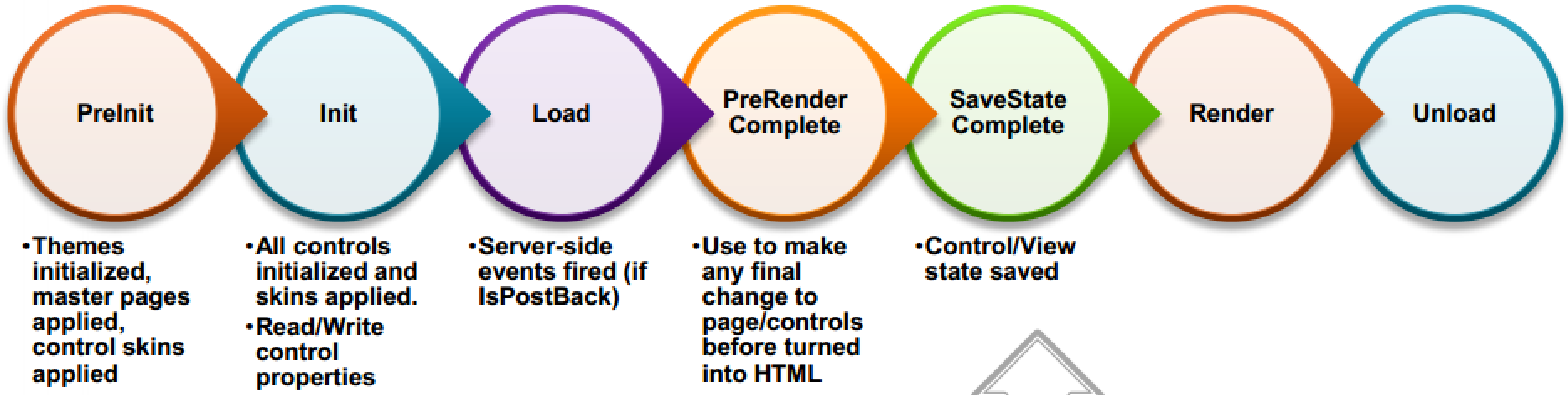
Some of Page Class Property

Property	Description
IsPostBack	Gets a value that indicates whether the page is being rendered for the first time or is being loaded in response to a postback.
Page	Gets a reference to the Page instance that contains the server control.(Inherited from Control.)
Parent	Gets a reference to the server control's parent control in the page control hierarchy.
PreviousPage	Gets the page that transferred control to the current page.
Request	Gets the HttpRequest object for the requested page.
Response	Gets the HttpResponse object associated with the Page object. This object allows you to send HTTP response data to a client and contains information about that response.
Server	Gets the **Server** object, which is an instance of the HttpServerUtility class.
User	Gets information about the user making the page request.

Web Page Life cycle (Page Event)

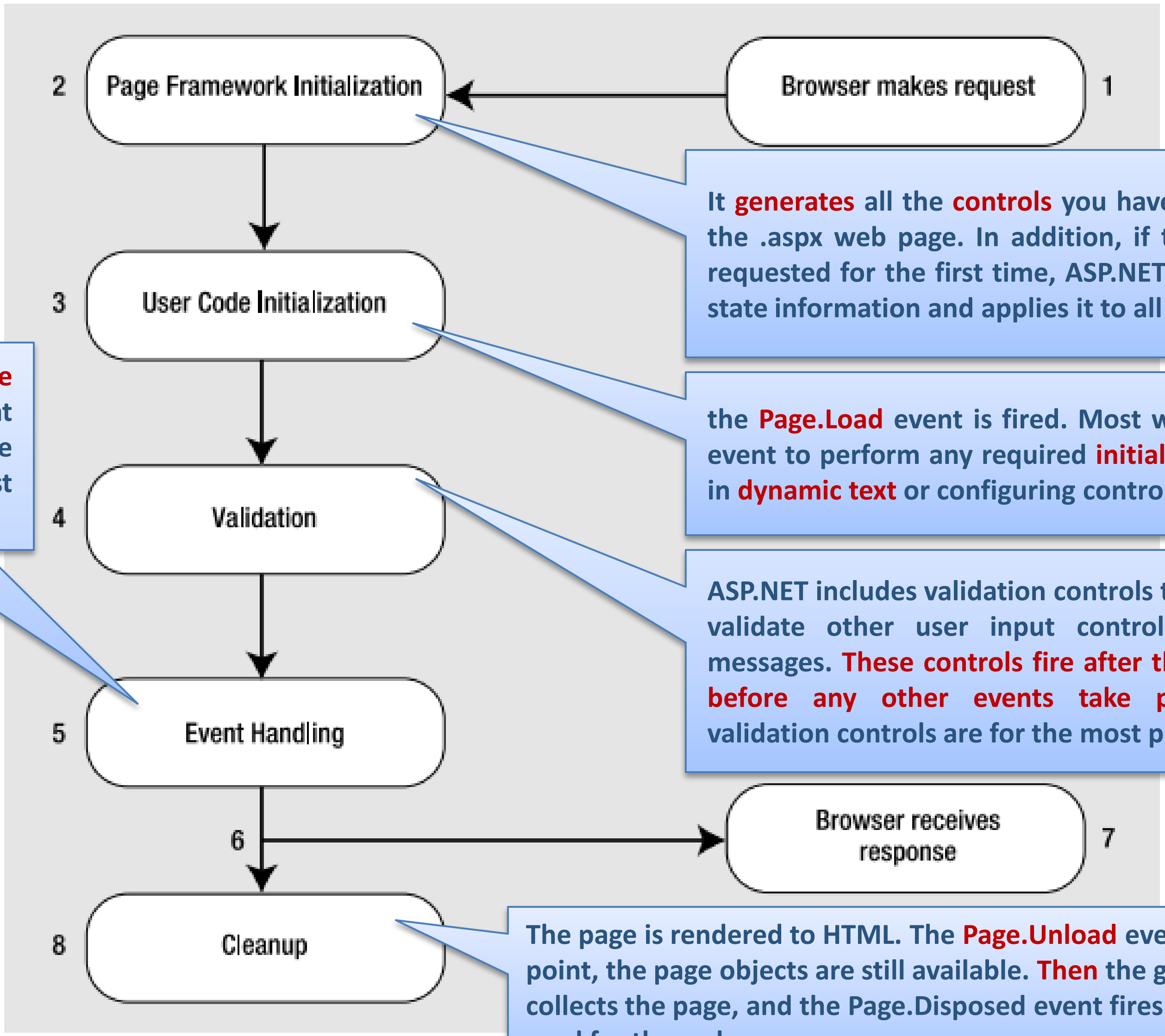
When an ASP.NET page runs, the page goes through a life cycle in which it performs a series of processing steps. These include:

- **initialization**
- **instantiating controls**
- **restoring and maintaining state**
- **running event handler code**
- **rendering**


PreInit
Init
Load
PreRender Complete
SaveState Complete
Render
Unload
•Themes initialized, master pages applied, control skins applied
•All controls initialized and skins applied.
•Read/Write control properties
•Server-side events fired (if IsPostBack)
•Use to make any final change to page/controls before turned into HTML
•Control/View state saved

It is important for you to understand the page life cycle so that you can **write code at the appropriate life-cycle stage** for the **effect** you **intend**.

Web Forms Processing Stages



2 Page Framework Initialization
Browser makes request 1
3 User Code Initialization
4 Validation
5 Event Handling
6
Browser receives response 7
8 Cleanup

It **generates** all the **controls** you have defined with tags in the .aspx web page. In addition, if the page is not being requested for the first time, ASP.NET **deserializes** the view state information and applies it to all the controls.

the **Page.Load** event is fired. Most web pages handle this event to perform any required **initialization** (such as filling in **dynamic text** or configuring controls).

ASP.NET includes validation controls that can automatically validate other user input controls and display error messages. **These controls fire after the page is loaded but before any other events take place**. However, the validation controls are for the most part self-sufficient

ASP.NET will now **fire** all the **events** that have taken place since the last postback.

The page is rendered to HTML. The **Page.Unload** event is fired. At this point, the page objects are still available. **Then** the garbage collector collects the page, and the Page.Disposed event fires. This is the end of the road for the web page.

What's in a Web Form?

- **Web Forms can contain several different items:**

  Directives:

  ```
  <%@ Page Language="C#" AutoEventWireup="True"%>
  ```

  Code Blocks:

  ```
  <script language="C#" runat="Server">….</script>
  ```

  Render Blocks:

  ```
  <%=UserDetails%>
  ```

  Server Controls:

  ```
  <asp:Label id="lblHelloWorld" runat="server" />
  ```

What's in a Web Form? (cont..)

User Controls:

```
<acme:Header id="ucHeader" runat="server" />
```

ASP.NET Expressions:

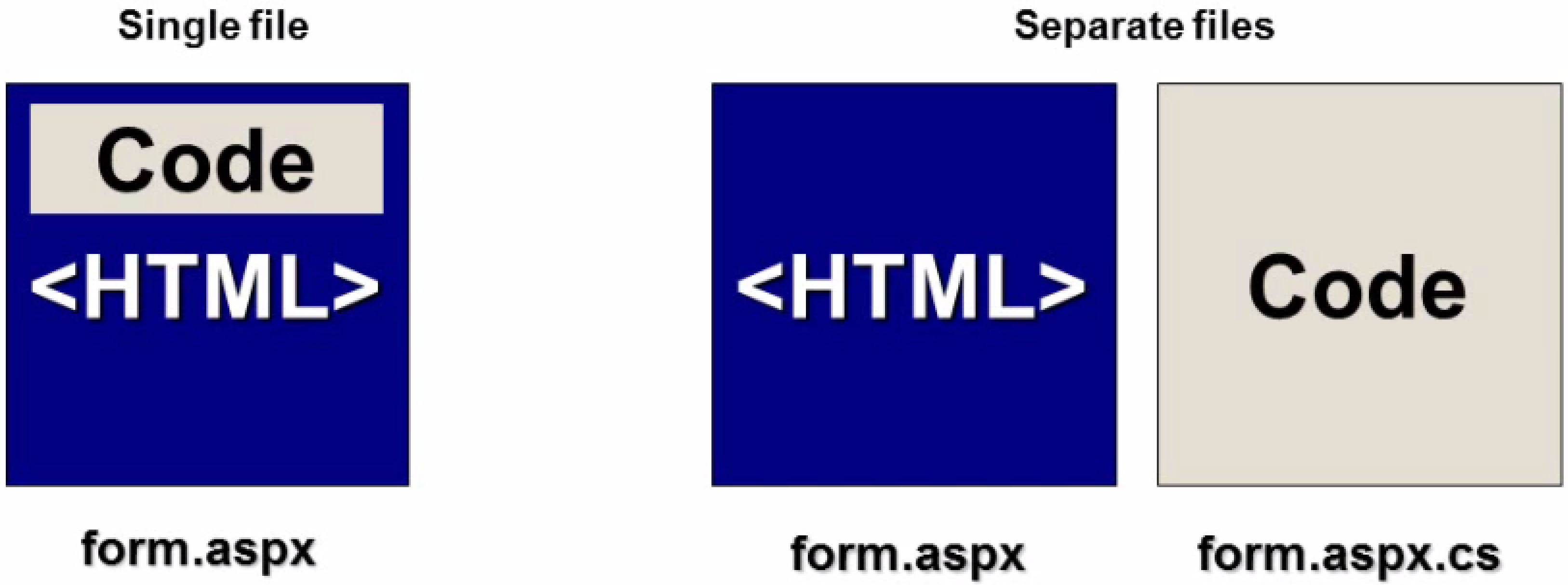
```
<%$ ConnectionStrings: NorthwindConnString %>
```

Data Binding Expressions:

```
<%# Eval("DBFieldName") %>
```

ASP.NET Code Separation

- Programming code can be placed into one file along with the HTML OR stored separately:



Page Directive

- The Page directive is add to the top of each ASP.NET page

  <%@Page Language=“C#”%>

- **Key Features:**
  - Specify the page’s language
  - Maintain scrollbar positions
  - Identify code file paths
  - Turn on or off tracing (logging)
  - Identify themes or master pages used by the page
  - Identify an error page

Page Directive Attributes

```
<%@ Page attribute="value" %>
```

Attribute	Description
Async	When true the generated page class derives from IHttpAsyncHandler which adds asynchronous capabilities.
CodeFile	Specifies the name of the referenced code-separation file to use for the page
EnableTheming	Indicates whether themes can be applied to the page
Language	Target language used within the page (C# or VB)
Trace	Turns tracing functionality on or off for the page.
MaintainScrollPositionOnPostback	JavaScript will be inserted into your rendered page that maintains the scroll position in the browser window for all postbacks
Theme	Specifies the name of the theme to use for the page

Web.Config file

- **Web.config** is the main settings and configuration file for an ASP.NET web application.
- It is an XML document that resides in the root directory of the site or application and contains data about how the web application will act.
- Contain:
  - security configuration,
  - session state configuration
  - application language
  - compilation settings.
- Web.config files can also contain application specific items such as database connection strings.
- https://msdn.microsoft.com/en-us/library/aa306178.aspx

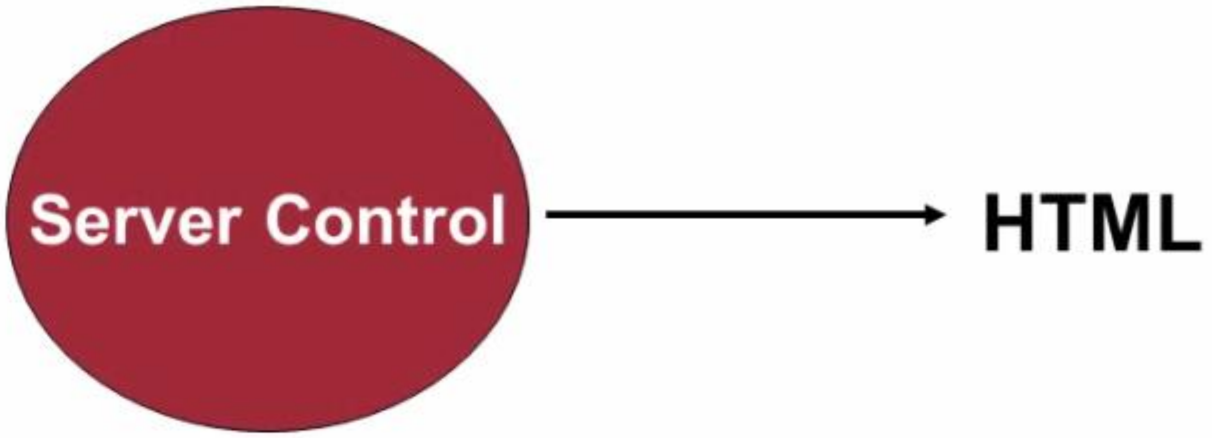
Test Demo

- Page LifeCycle

Test Demo

ASP.NET Web Server Controls

- ASP.NET relies on Web Server Controls to collect, display and validate data
- Server Controls are classes with properties, methods and events
- Server Controls dynamically generate HTML5 compliant code



ASP.NET Server Controls

- ASP.NET Web server controls are objects on ASP.NET Web pages that run when the page is requested and render markup to a browser.
- Many Web server controls are similar to familiar HTML elements, such as buttons and text boxes.
- Other controls encompass complex behavior, such as a calendar controls or controls that manage data connections.

ASP.NET Control Types

- **ASP.NET controls are a key technology used by the Page class to dynamically generate HTML output.**
- **Four basic types of server controls exist:**
  - **Web Server Controls**: Strongly-typed programmable objects.
  - **HTML Server Controls**: Similar to regular HTML elements but you control them on the server-side.
  - **Validation Controls**: Used to validate Web Form submissions.
  - **User Controls**: Custom controls such as headers, footers and menus.

ASP.NET Control Examples

- **Web Server Controls:**

```
<asp:TextBox id="txtName" runat="server" />
```

- **HTML Server Controls:**

```
<input type="hidden" id="hidVal"
    name="hidVal" runat="server" />
```

- **Validation Controls:**

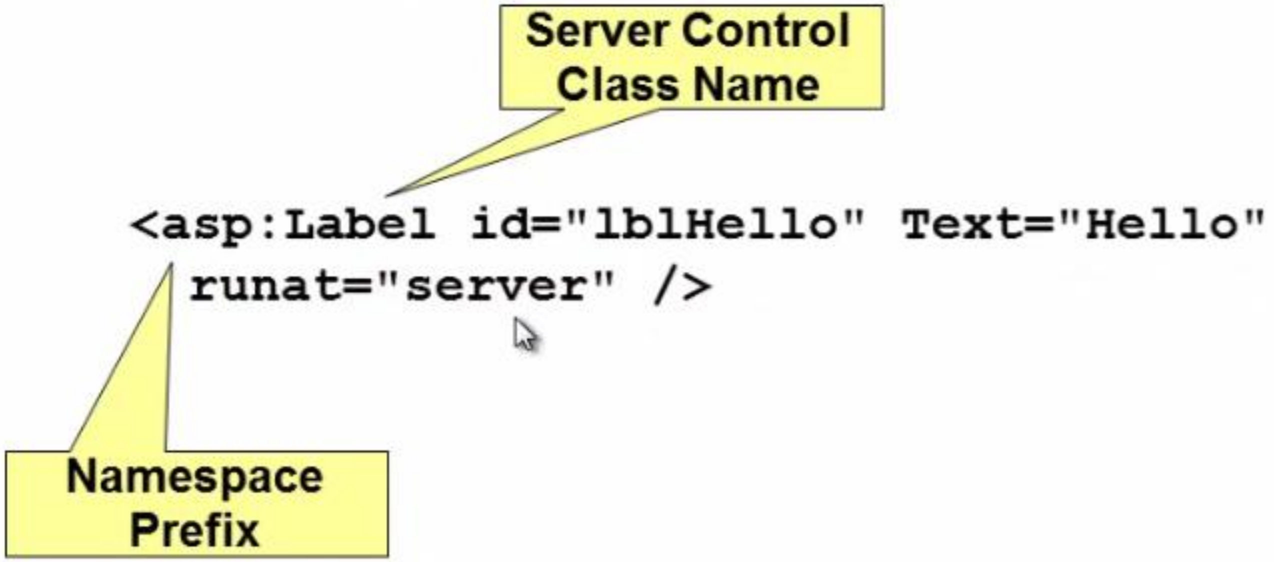
```
<asp:RequiredFieldValidator id="valTxtName"
  runat="server" ControlToValidate="txtName" />
```

- **User Controls:**

```
<acme:Header id="ucHeader" runat="server" />
```

Declaring Server Controls

- Server Controls are used in a Web Form by prefixing the control name with an "asp" namespace prefix:



```
<asp:Label id="lblHello" Text="Hello"
  runat="server" />
```

Server Control Properties

- **Server Control properties can be set declaratively using attributes:**

```
<asp:GridView id="gvRecords" runat="server"
   BorderColor="black"
   BorderWidth="1"
   GridLines="Both"
   CellPadding="3"
   CellSpacing="0"
   Font-Name="Verdana"
   Font-Size="8pt"
/>
```

Property

Hooking up Web Server Control Events

- Server Controls expose different events that can be handled in Web Forms
- The OnClick attribute can be added to hook a Button Web Server control to a Click event handler:

```
public void btnSubmit_Click(object sender, EventArgs e)
{
    lblMessage.Text = "You clicked btnSubmit!";
}
```

```
<asp:Button id="btnSubmit" OnClick="btnSubmit_Click"
  runat="server" Text="Submit" />
```

ASP.NET Server Controls *(cont.)*

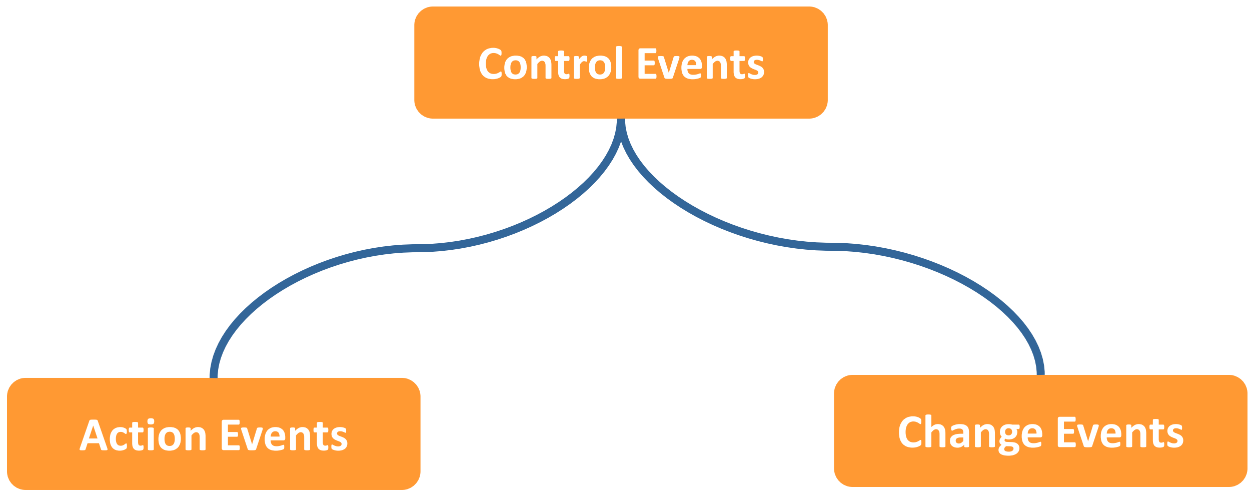
HTML Controls	Web Server Controls
Have no runat="Server"	Have a runat="Server"
Does not maintain State	Maintains State Automatically
Can only write script code like javasript (but no C# or VB)	Handle events in the code-behind (C# or VB)
Have a limited set of properties & events	

Test Demo

- Page.IsPostBack

Test Demo

Controls Events (Change & Action)



Ex: Button

Ex: DropDownDist
TextBox

Test Demo

- Control Event

Test Demo

Setting a Default Button

- Setting the default button when a user hits the "enter" key can be done using the defaultButton attribute:

```
<form defaultButton="btnSearch" runat="server">
```

- The <asp:panel> control can override the defaultButton specified when the panel has focus:

```
<asp:Panel runat="server" defaultButton="btnOK">
 …
</asp:Panel>
```

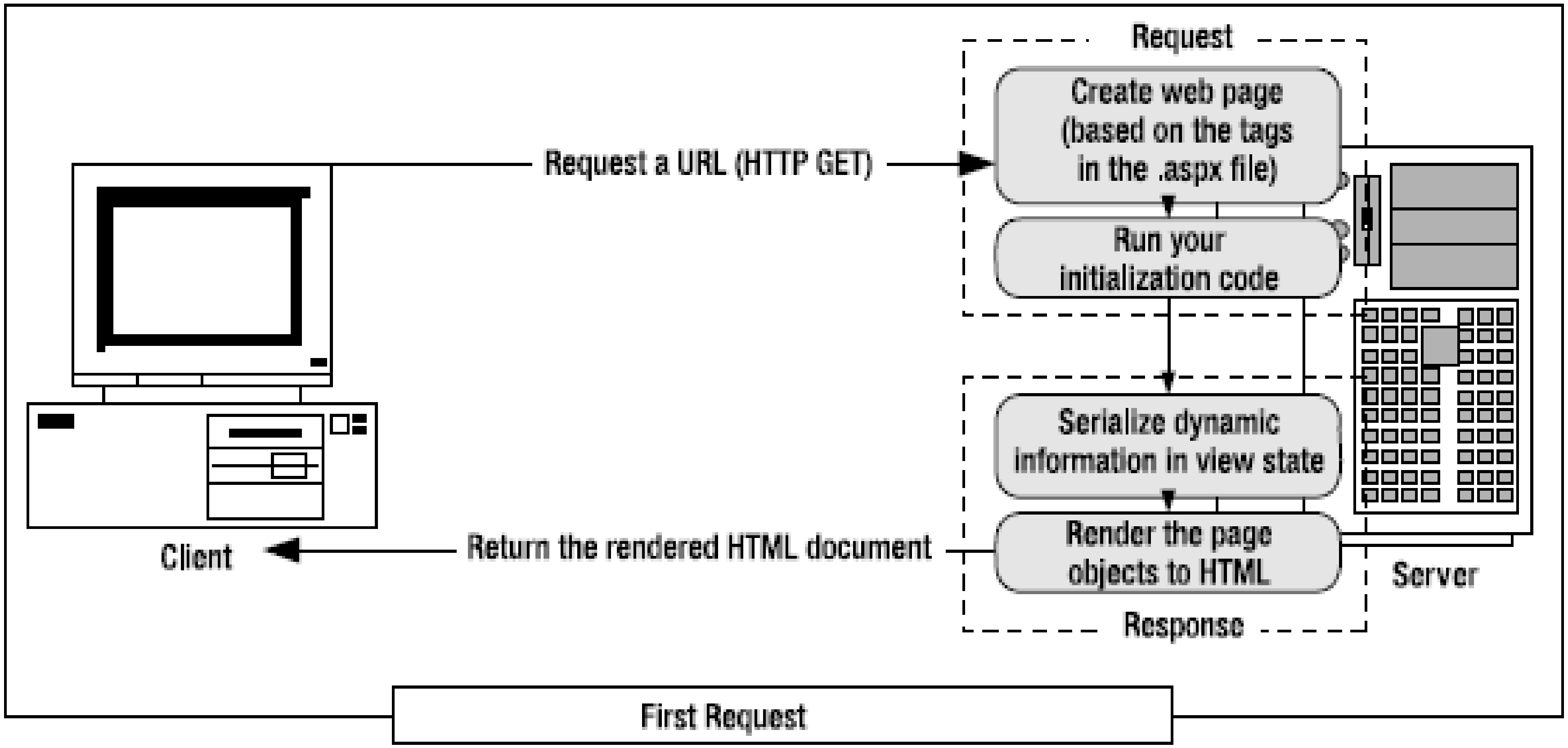
Setting the Default Focus

- **Setting the default focus for a page can be done using the defaultFocus attribute:**

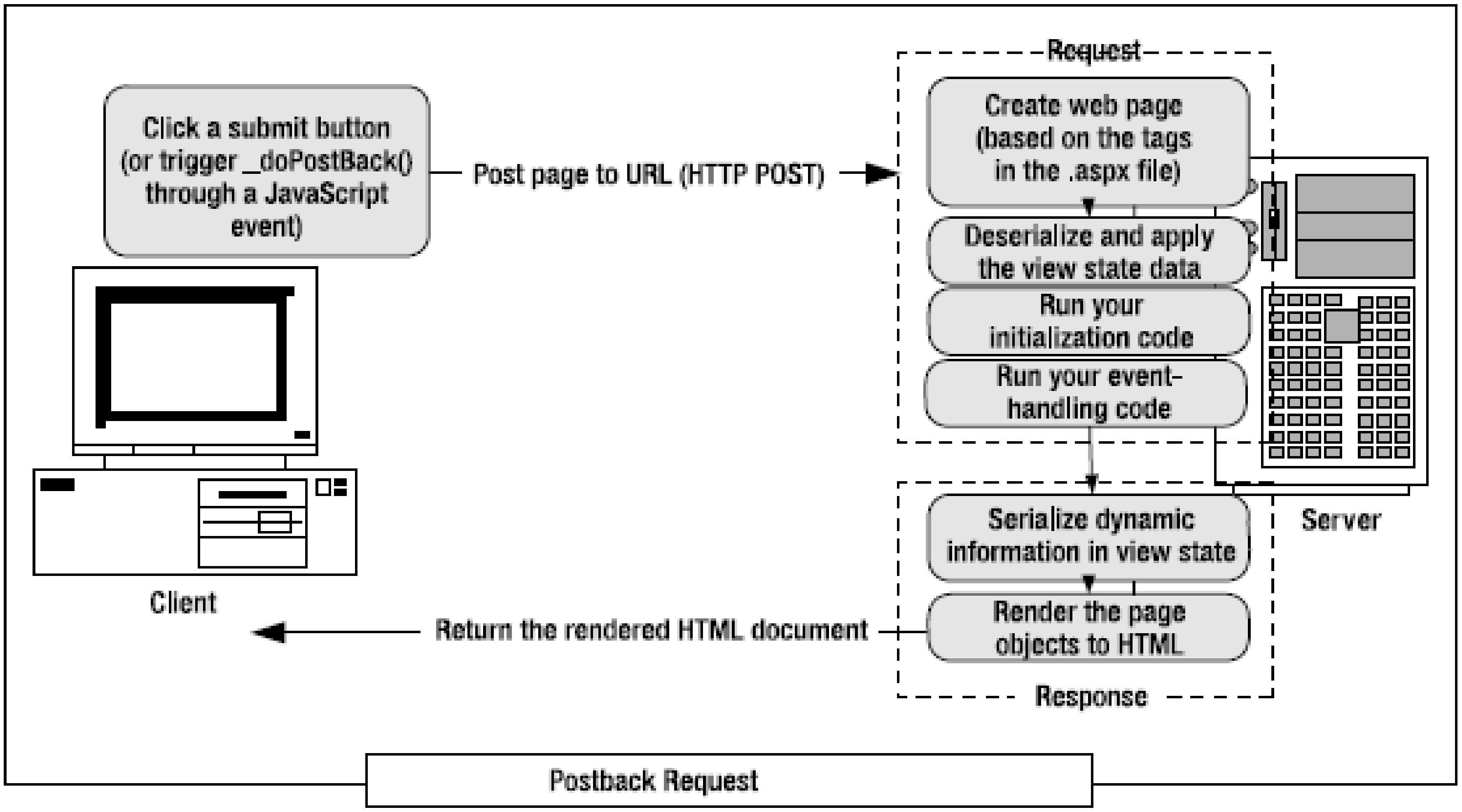
```
<form defaultFocus="txtName" runat="server">
```

- **Programmatic support for validating groups:**
  - Page.SetFocus(control)
  - Page.SetFocus("ClientID")
  - txtName.Focus()

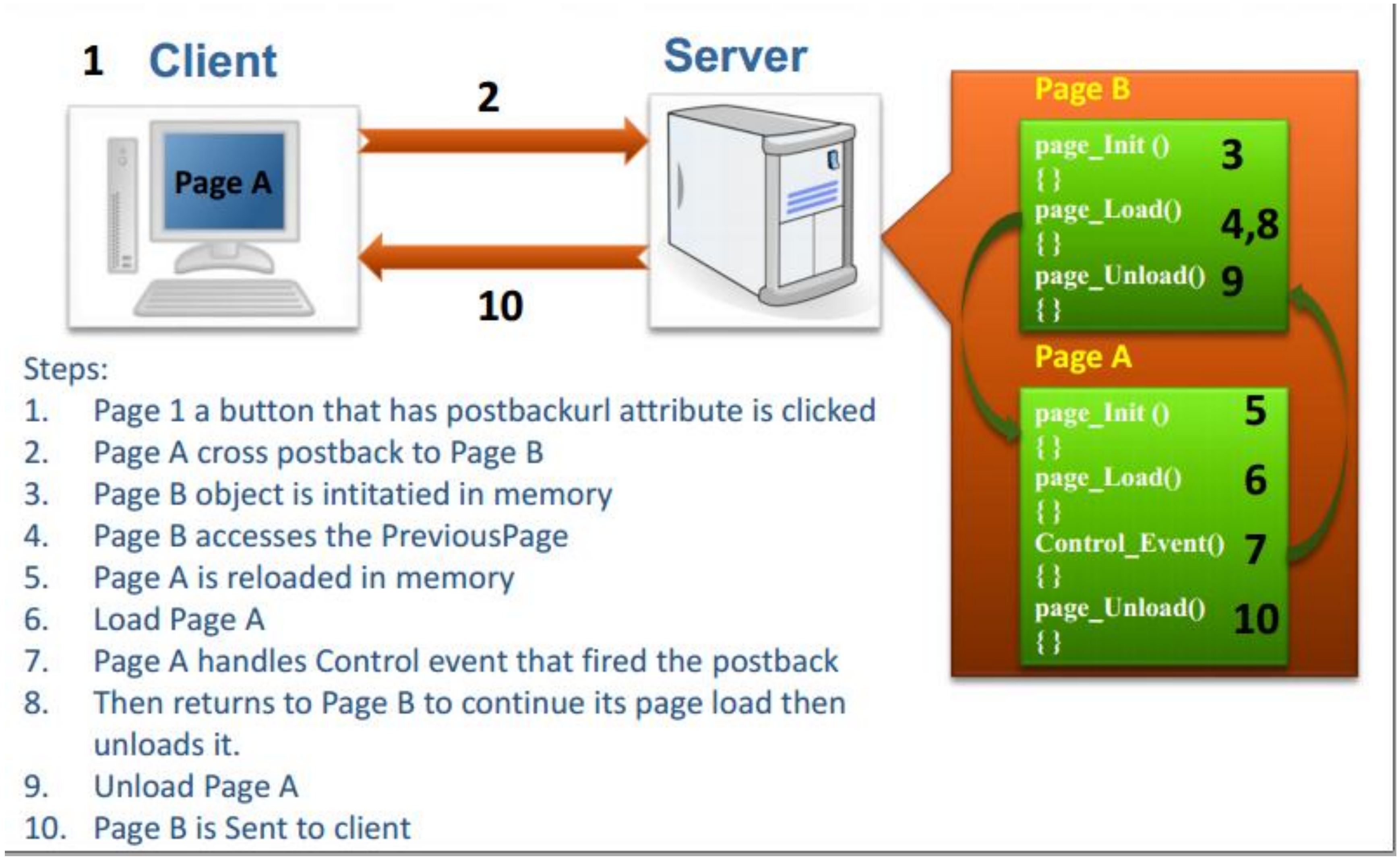
Web Page Life cycle --First Request


Request
Create web page (based on the tags in the .aspx file)
Run your initialization code
Request a URL (HTTP GET)
Serialize dynamic information in view state
Render the page objects to HTML
Return the rendered HTML document
Client
Server
Response
First Request

Web Page Life cycle --PostBack Request

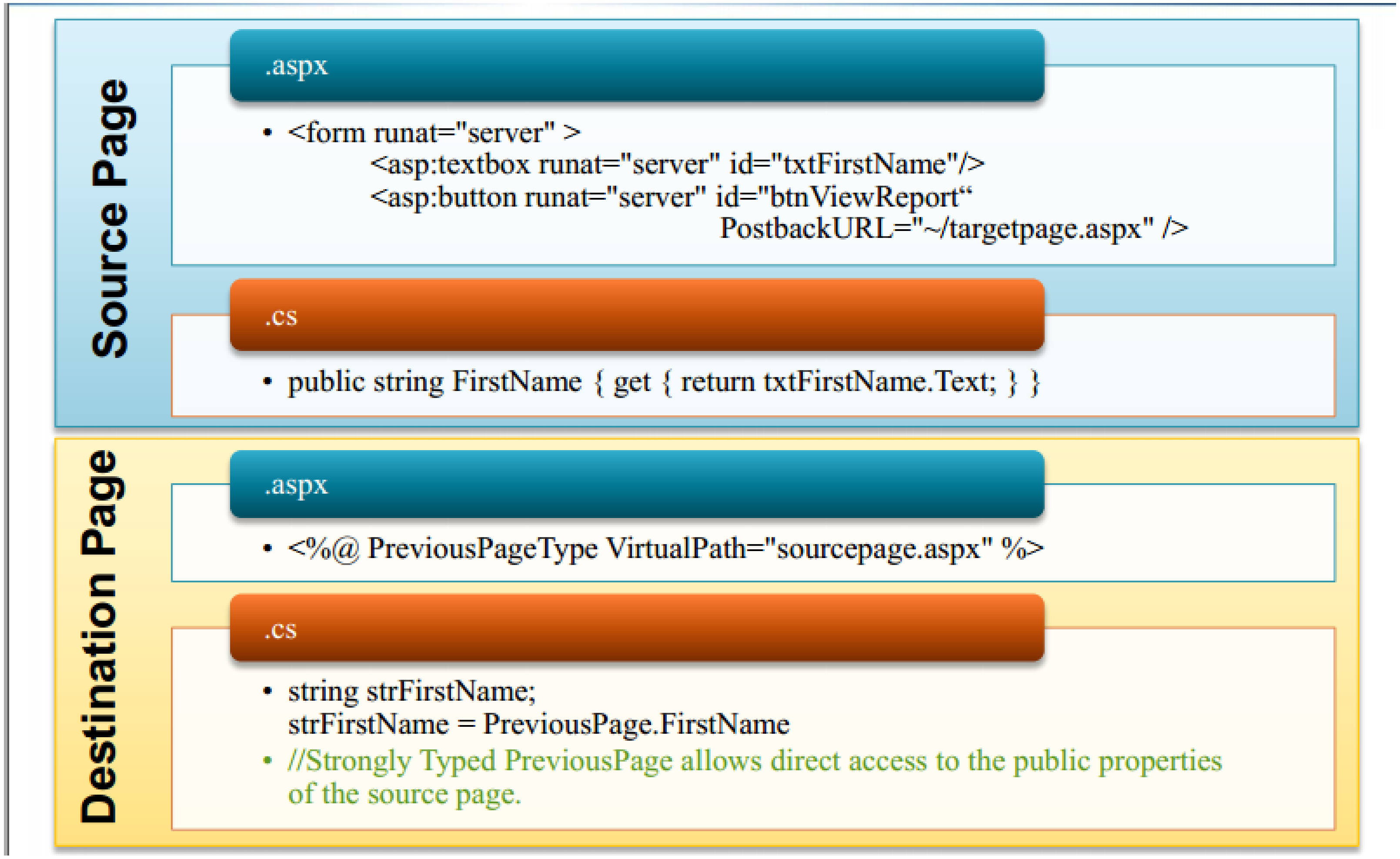

Click a submit button (or trigger _doPostBack() through a JavaScript event)
Post page to URL (HTTP POST)
Request
Create web page (based on the tags in the .aspx file)
Deserialize and apply the view state data
Run your initialization code
Run your event-handling code
Serialize dynamic information in view state
Render the page objects to HTML
Response
Server
Client
Return the rendered HTML document
Postback Request

Cross Page Postback



Steps:

1. Page 1 a button that has postbackurl attribute is clicked
2. Page A cross postback to Page B
3. Page B object is intitatied in memory
4. Page B accesses the PreviousPage
5. Page A is reloaded in memory
6. Load Page A
7. Page A handles Control event that fired the postback
8. Then returns to Page B to continue its page load then unloads it.
9. Unload Page A
10. Page B is Sent to client



Source Page

.aspx

```
• <form runat="server" >
            <asp:textbox runat="server" id="txtFirstName"/>
            <asp:button runat="server" id="btnViewReport“
                                                      PostbackURL="~/targetpage.aspx" />
```

.cs

```
• public string FirstName { get { return txtFirstName.Text; } }
```

Destination Page

.aspx

```
• <%@ PreviousPageType VirtualPath="sourcepage.aspx" %>
```

.cs

```
• string strFirstName;
  strFirstName = PreviousPage.FirstName
• //Strongly Typed PreviousPage allows direct access to the public properties
  of the source page.
```

- https://www.asp.net/web-forms/what-is-web-forms
- What’s new in asp.net 4.5 webforms
  - http://linqto.me/ASPnet45P1