# ASP.Net – Part I

Jim Fawcett

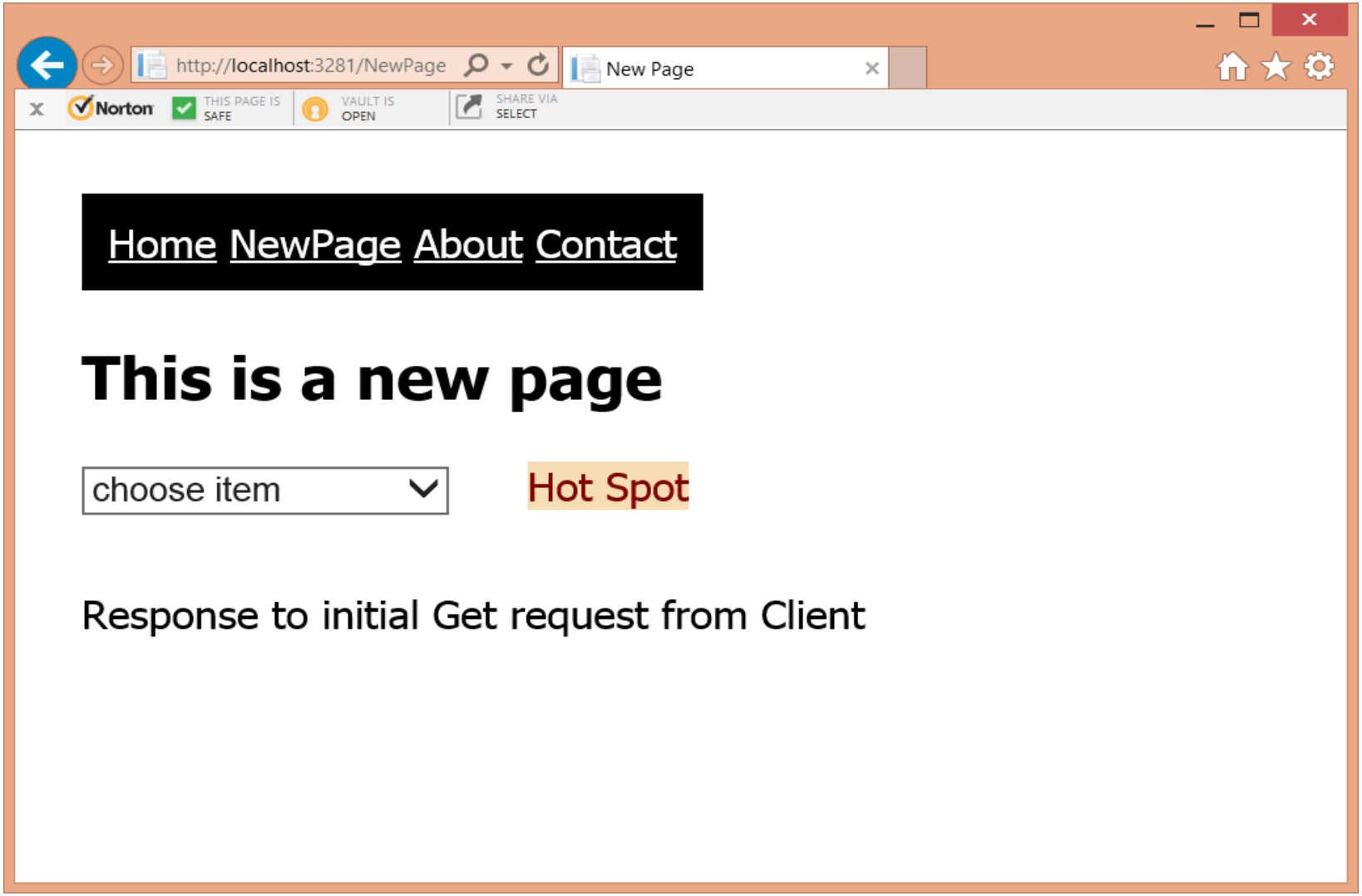
CSE686 – Internet Programming

Spring 2014

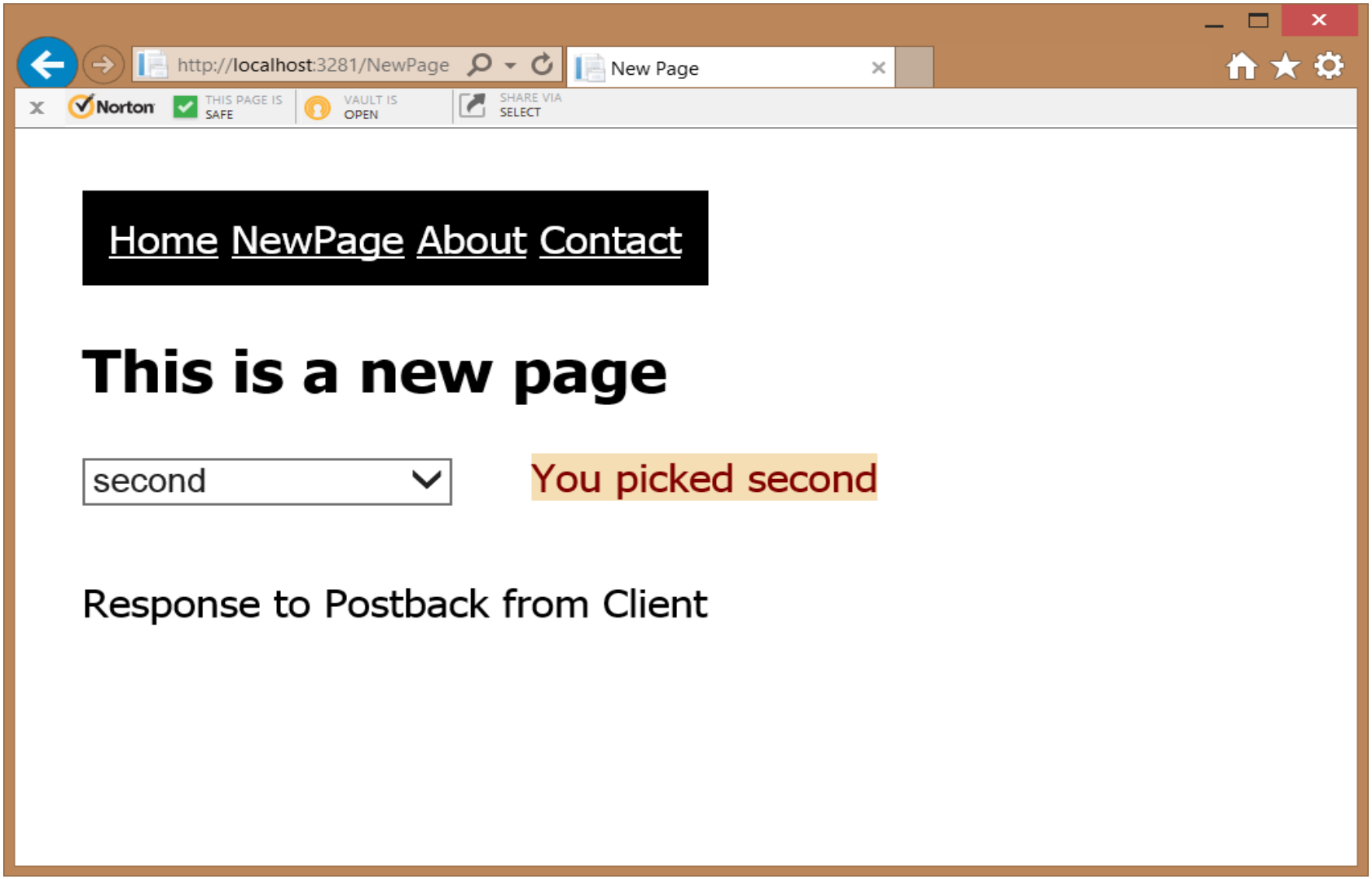
# Building an Asp.Net Application

- Create a Visual Studio 2013 project
  - Visual C# > Web > Asp.Net Web Application > OK
  - Web Forms > OK
- Add a WebForm
  - Add > New Item > Web Form > OK
  - Add controls from toolbox
  - Manually add styles, and text
- Build and run.
- Iterate to add additional capability
  - More forms
  - XML and SQL data
  - More controls

# Example


http://localhost:3281/NewPage
New Page
Norton
THIS PAGE IS SAFE
VAULT IS OPEN
SHARE VIA SELECT
Home NewPage About Contact
This is a new page
choose item
Hot Spot
Response to initial Get request from Client

# Example - Continued



# Server-Side Programs

- You can run any .Net webform that resides in a virtual directory simply by requesting it from a browser:
  - http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE686/code/AspApps/BasicAsp/PickAgain.aspx
  - Provided that directory permissions allow this.
- On Windows platforms most server-side processing takes the form of Active Server Page (ASP), ASP.Net, or ASP.Net MVC applications.

# Running Example Code

- Asp.Net applications run from a virtual server:

  lcs-vc-fawcett2.syr.edu

  for which I have administrator privileges, needed to set virtual directory properties.
  - To run the examples from your own machine, just right click on a zip file in the college server and select "save target as".
  - If you are running Win7 or Win8 you need to use IIS Manager to create a virtual directory and application. Use Asp.Net 4.5 Application Pool.
  - Now you can open the site with http://localhost/VirtualDirectoryName/formName.aspx
  - You can also run applications in Visual Studio by opening the sln if the application has one.

# Traditional ASP

- Traditional (pre .Net) ASP provides interpreted application scripts running in the memory space of the IIS web server.
  - A traditional ASP page consists of a mix of HTML, sent directly to the requesting browser and Javascript or Vbscript executed on the server, usually to generate html for display or interact with a backend database.
  - Traditional ASP uses a set of standard server side COM objects and can use custom COM objects as well.
  - Deploying custom COM objects to remote servers has been a major problem.

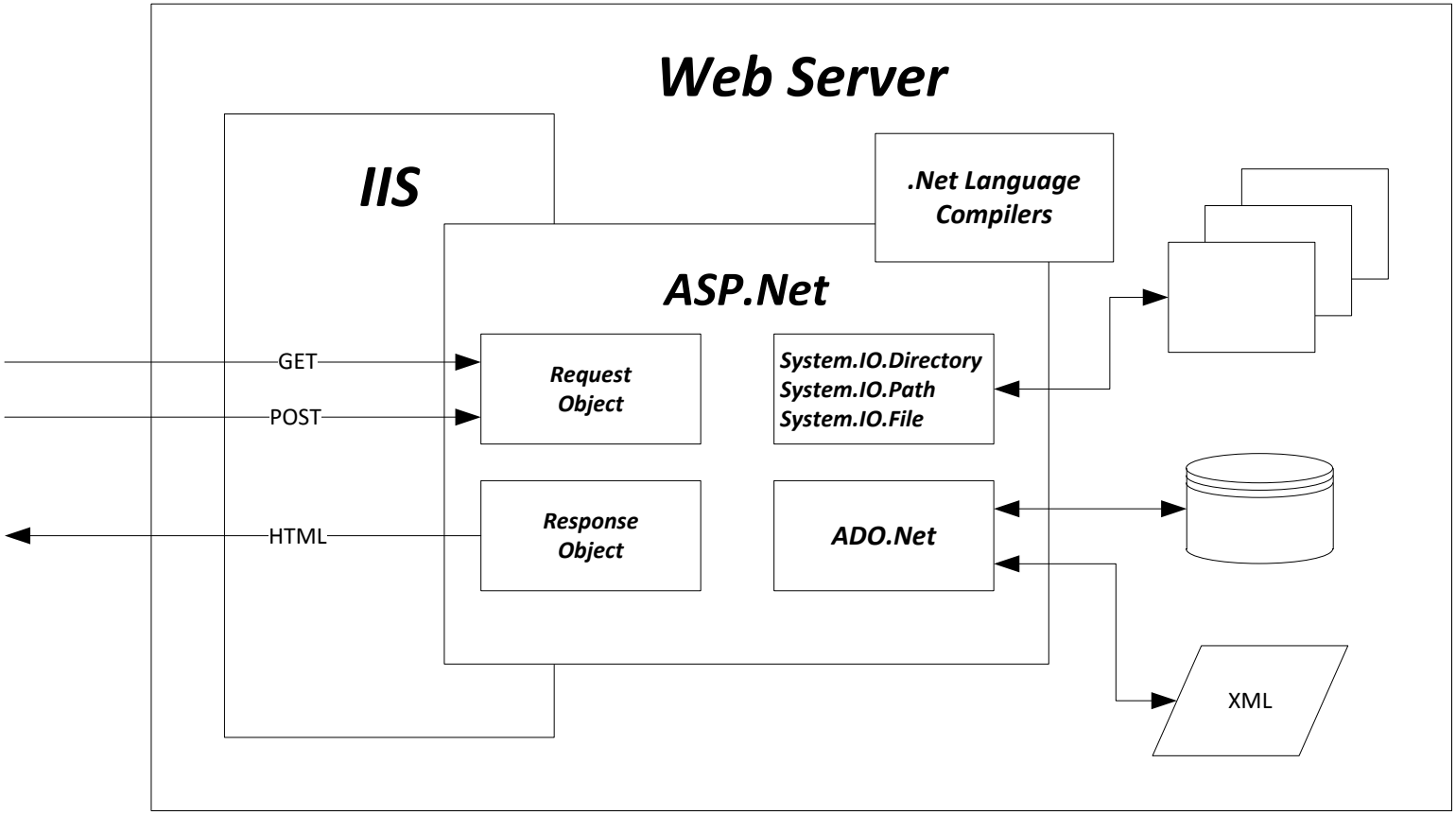
# Server-Side Objects

- Traditional ASP provides seven objects used for server-side programming:
  - Application:
    - starts when IIS starts and runs until IIS shuts down
  - ASPError
    - ASPError object is returned by Server.GetLastError(), and has the properties: Source, Category, File, Line, Description, ASPDescription
  - ObjectContext
    - Access to COM+ objects
  - Request:
    - Provides methods: Form(), QueryString(), Cookies(), ServerVariables()
  - Response:
    - Provides methods: Write(), Clear(), End(), Flush(), Redirect(), Buffer, Expires, IsClientConnected(), PICS()
  - Server:
    - Provides methods: Execute(), Transfer(), MapPath(), URLEncode(), HTMLEncode(), GetLastError()
  - Session:
    - starts when a user requests first page and ends with a timeout

# ASP .Net

- ASP.Net supports the traditional style, but adds processing power of compiled C# and a pervasive object model.
  - We can create user-defined classes in C# and use them on ASP pages.  Any .Net language can be used this way.
  - Web controls are based on CLR objects.  Control state is sent back and forth between client and server in a hidden viewstate.
  - An ASP.Net page can easily be turned into a server control that can be used on any other ASP page.

# ASP.Net Environment



# ASP.Net WebForm

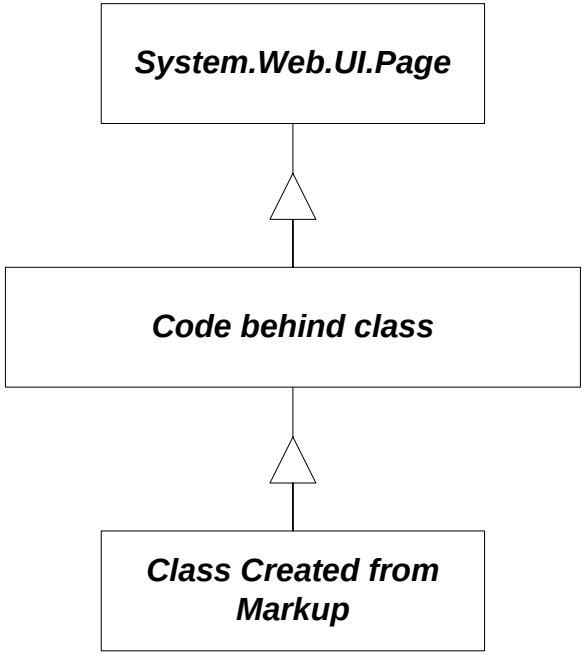
- An ASP.Net Web Form page has:
  - A single form control:
    - <form id="form1" runat="server"> … </form>
  - Zero or more server controls:
    - That render themselves to HTML
    - Have methods, properties, and events
  - Server controls come in two flavors:
    - HTML controls that have html tags
      - <input id="Checkbox1" type="checkbox" runat="server"/>
    - Standard (Web) controls
      - <asp:Button ID="Button1" runat="server" Text="…" />

# Page Rendering Model

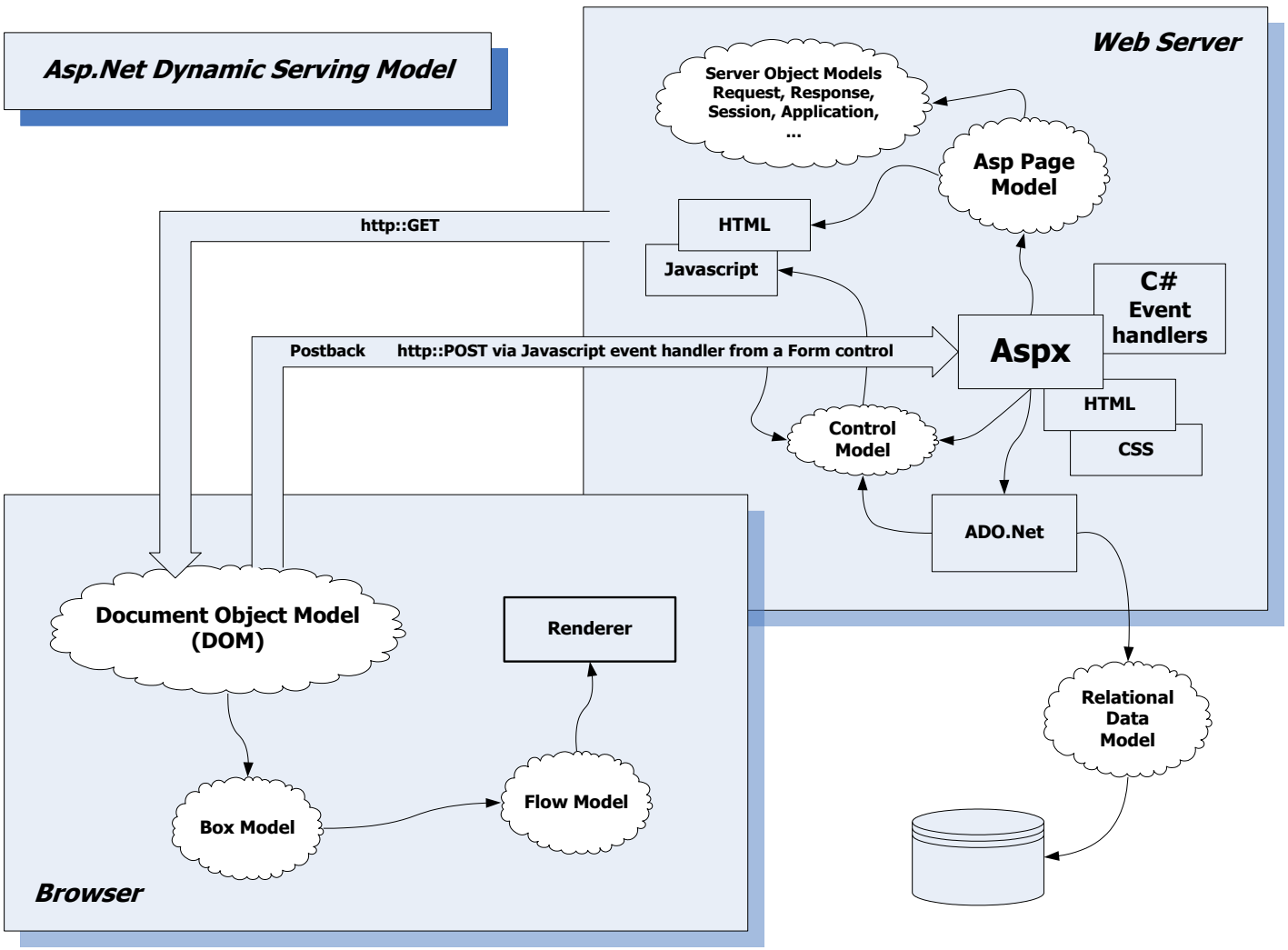
- When an aspx page is requested, a Page object is instantiated, and an object is created for each control on the page.
  - Each of the control objects is added to the page's controls collection.
- When the page renders:
  - It generates HTML representing the form
  - Calls each of its controls to render itself, resulting in:
    - HTML generated for each control.
    - Javascript that generates a postback to the server each time the client takes some action that triggers a client-side HTML-based event.

# The First Call

- The first time a page is loaded after creation or a change to its text:
  - ASP.Net parsers extract code from the codebehind file and build a class, derived from System.Web.UI.Page, that contains a collection for the page's controls, renderning code, events, …
  - The aspx file is parsed, and a class, derived from the code behind-based class, is built to render the page.



# Page Serving Model



# ASP.Net Page Contents

- An ASP.Net page contains:
  - Directives for the compiler, which must include a <@Page … > directive.
  - Literal HTML content, intended for the client
  - Code in C#, VB, Jscript.Net. The Code will:
    - Respond to client-side events that result in a PostBack to the server

    and may also:
    - generate HTML for client
    - get or send data to a database on this or a remote server
    - interact in some way with the server's file system
  - Traditional script, e.g.: Javascript or Vbscript
  - Embedded ASP.Net server controls
    - Means to collect information from, and present information to, clients
    - Control state is preserved in transactions between client and server using a hidden viewstate.
  - Server HTML controls, based on the traditional HTML controls
    - Also manages information between client and server. Preserving state requires more work on programmer's part.

# Page GET Life Cycle

- Browser issues an HTTP GET request for an aspx page.
- The IHttpHandler::ProcessRequest method is called.
- Handler creates a Page-derived class by loading the aspx page specified in the request, and loading any requred dynamic link libraries (dlls), residing in the application's bin directory.
- Server calls Page's ProcessRequest, which results in a recursive call to __Render__control for the page and each of its child controls.
  - Each control's __Render__control call is responsible for constructing html for it's own part of the page display.

# Page POST Life Cycle

- Any event triggered by a user action in the client browser generates a submit request and subsequent HTTP POST message. The body of the message contains data from the form to be processed on the server.
- POSTed data is captured by the server's request object and processed by event handlers in the original aspx page's C# Codebehind. This processing almost always results in more rendering and the page is then sent back to the client.
- Complete cycle:
  - GET ➔ reply ➔ user action ➔ POST ➔ reply ➔ user action …
  - The Page.IsPostback property tells server code whether processing is in response to a GET or a POST command.

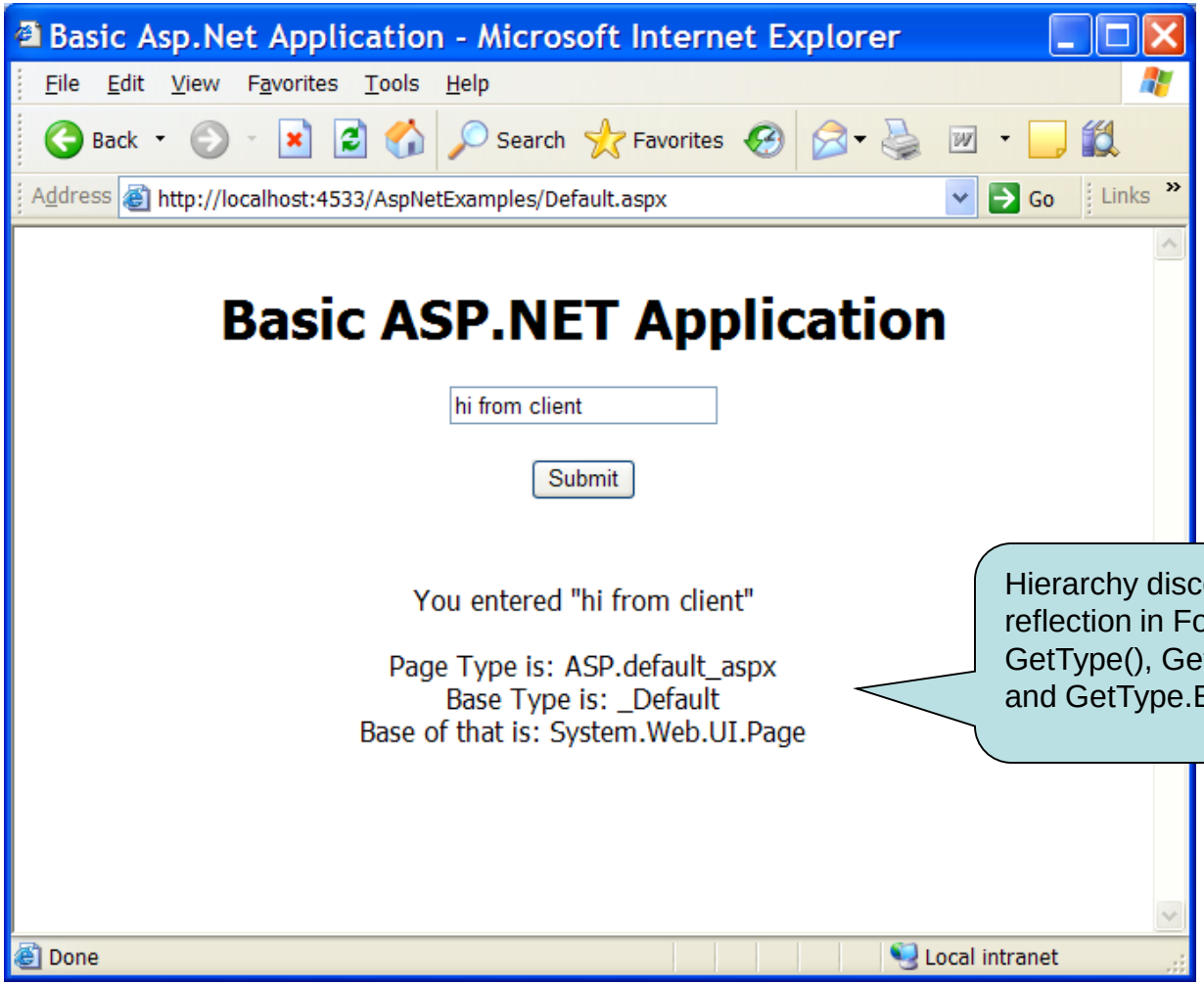
# What's So Great about Asp.Net?

- The object model, with its Page class that supports Asp.Net pages, is extremely helpful in building effective websites:
  - We can build a Page derived class that will serve as a base class for all our web pages that contains all the code common to pages in the site:
    - styles
    - controls (navigation bar and user access control for example)
    - Headers and Footers
  - User defined controls are easy to define and reuse.
  - All of the power of the .Net framework is available for our server-side processing, e.g.:
    - directory and file manipulation
    - Regular expression analysis
    - XML processing
    - Web services
    - Advanced data management classes

# ASP .Net Applications

- You can build an ASP application using notepad to create an aspx page, a C# code page, and, optionally, a web.config file.
- Here's what is required to do that:
  - Create an aspx file that has:
    - Page directive that contains an Inherits attribute that specifies a class from the code page, e.g., Inherits="_Default"
    - HTML including a form and one or more controls
  - Create a codebehind cs file that contains:
    - Event handlers for each of the aspx control events you want to handle
    - Helper code
    - Make each of these members of a class derived from System.Web.UI.Page
    - Declare protected fields with names the same as the IDs of the controls on the aspx page, e.g., TextBox UserName;
    - Compile the cs file into a library dll and place the dll in a bin subdirectory immediately below the aspx application.
- Of course, you can do that quickly by running the website wizard.

# Page Hierarchy using Reflection




Hierarchy discovered by using reflection in Form_Load, e.g., GetType(), GetType().BaseType, and GetType.BaseType.BaseType

# HTMLServer Controls

- Input: Button, Reset, Submit, Text, File, Password, Checkbox, Radio, Hidden
- Textarea, Table, Image, Select, Horizontal Rule, Div
- <INPUT id="UserID" style="…" type="text" runat="server">
- Allows you to take any valid html page and change its extension to aspx and have it run as an ASP.Net application.
  - This makes migration from older sites somewhat easier.

# WebServer Controls

- Label, TextBox, Button, LinkButton, ImageButton, HyperLink, DropDownList, ListBox, CheckBox, CheckBoxList, RadioButton, Image, ImageMap, Table, BulletedList, HiddenField, Literal, Calendar, AdRotator, FileUpLoad, Xml, MultView, Panel, PlaceHolder, View, Substitution, Localize

- <asp:Label id="Label1" runat="server"
  BorderColor="maroon">default text</asp:Label>

- Richer behavior, styles, and configurations than HTML controls.

# Server Controls

- WebServer Controls
  - These controls have state which is marshalled between client and server in a hidden ViewState variable.
  - Events, like button clicks, that happen on the client side, are marshalled back to the server to trigger event handlers in C#, processed on the server.
    - This event model is based on encapsulated Javascript processing that posts page data back to the server when a specific event occurs.

# References


- Pro ASP.Net 4.0 in C# 2010, MacDonald, Freeman & Szpuszta, Apress, 2010
- Programming Microsoft .Net, Jef Prosise, Microsoft Press, 2002
- http://Asp.net/ has some interesting tutorial material and videos on ASP.Net.