WAT2117C

Internet Programming II

Week 1 – Overview of the .NET Framework & Programming with VB.Net

Overview of the ASP.NET Framework

- ASP.NET and the .NET Framework
- Understanding ASP.NET Controls
- Understanding ASP.NET Pages
- Installing the ASP.NET Framework

What is ASP.NET?

- Technology that provides services to allow for the creation, deployment, and execution of Web Applications and Web Services
- Built on .NET Framework: any .NET programming language can be used (C#, VB.Net)
- Object-oriented model
- Separation of code and UI
- Maintains page state
- Session management
- Support for Caching
- Debugging
- Embedding external libraries in VS
- and much more…

ASP.NET Versions

- 1.0 – the initial release (2002)
- 1.1 – included a new version of Visual Studio .NET; no major changes in ASP.NET (2003)
- 2.0 – a wide array of new ASP.NET features and functionality, along with many new features in the .NET Framework; included new versions of Visual Studio and SQL Server (2005)
- 3.0 / 3.5
- 4.0
- 4.5
- 4.5.1 / 4.5.2
- 4.6,
- 4.6.1
- 4.7.2, 4.8
- Versions: https://goo.gl/VpOkJy

Installing VS2013/15/17

- Visual Studio can target: [Compatibility]
  - ASP.NET 2.0
  - ASP.NET 3.0
  - ASP.NET 3.5
  - ASP.NET 4.0
  - ASP.NET 4.5
  - ASP.NET 4.5.1, 4.5.2, etc…

  - To download and install Visual Studio Community

    - https://visualstudio.microsoft.com/downloads/

    Select: ASP.NET and Web Development option

The Good News…

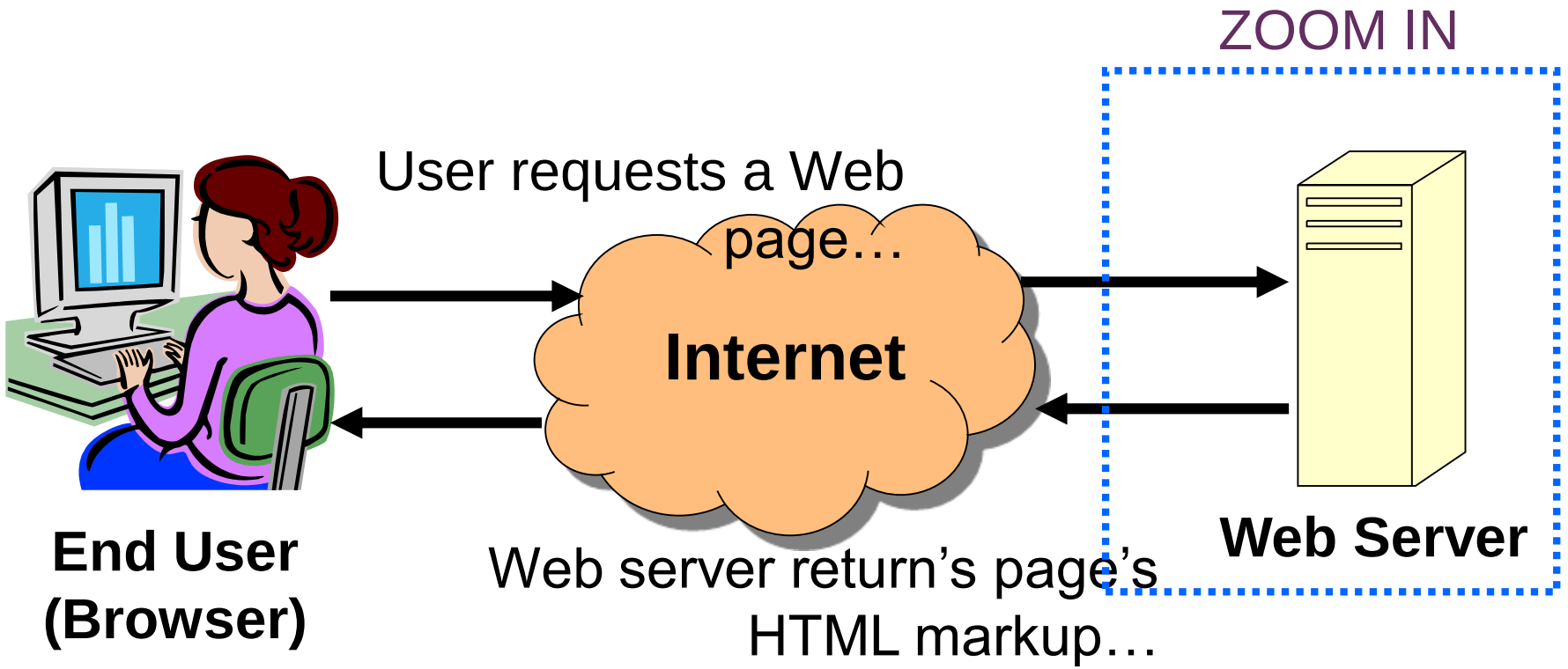
- Different versions of ASP.NET can run side-by-side on a Web server.
- You can install both Visual Studio .NET 2002, 2003, 2005, 2010, 2013, 2015 etc.. on the same machine;
- Along with SQL Server 2000, 2005, 2008, 2012 etc..

Using HTTP in Web Applications

- This text-based, request-response protocol defines how web browsers and web servers communicate with each other:
  - HTTP Requests
  - HTTP Response

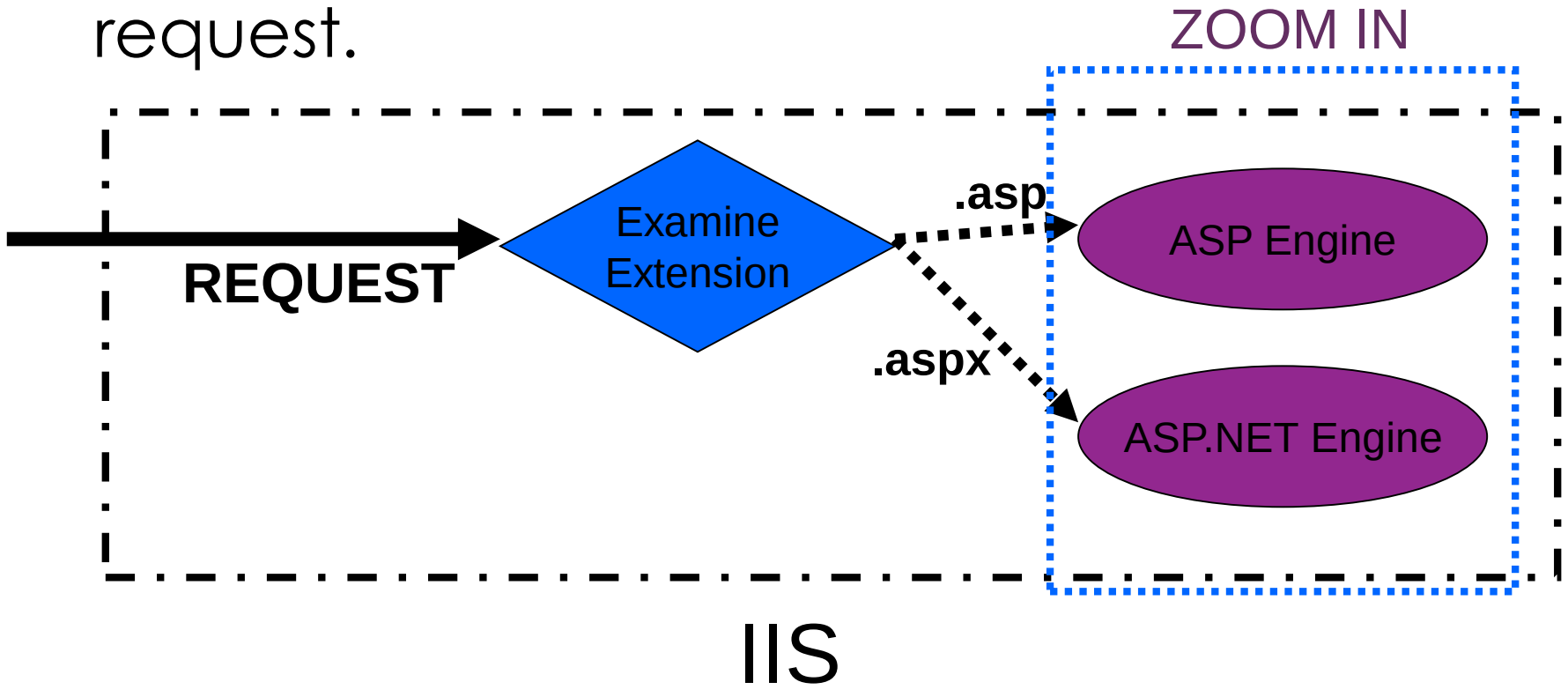
Client/Server Architecture for Web Applications

- Let's "zoom in" on the sequence of a Web request



Client/Server Architecture for Web Applications

- IIS receives the request, examines the extension, and decides who should handle the request.



Role of global.asax in Life Cycle Events

- Global.asax file is created at the root directory of the application.
- This file contains the code to handle application events.

```
Sub Application_Start()
        …
End Sub
```

Project Types for Web Application

- File-based website
- Local IIS HTTP
- Remote IIS HTTP
- FTP

ASP.NET and the .NET Framework

- To build ASP.NET pages, you need to take advantage of the features of the .NET Framework.

- The .NET Framework consists of two parts:
  - the FCL and
  - the CLR.

Understanding the Framework Class Library

- The .NET Framework contains thousands of classes that you can use when building an application.
- The Framework Class Library was designed to make it easier to perform the most common programming tasks.
- Here are just a few examples of the classes:
  - Random class
  - Graphics class
  - SmtpClient class
  - File class

Understanding the Framework Class Library

- The .NET Framework includes around 18,619 types; 12,909 classes; 401,759 public methods; 93,105 public properties; and 30,546 public events.

- Each class in the Framework can include properties, methods, and events. For example, here is a partial list of the members of the **SmtpClient class**:

- **Properties**
  - Host: The name or IP address of your email server
  - Port: number of the port to use when sending email message
- **Methods**
  - Send: Enables you to send an email message synchronously
  - SendAsync: send an email message asynchronously
- **Events**
  - SendCompleted: Raised when an asynchronous send operation completes

Understanding Namespaces

- A namespace is simply a category. For example, all the classes related to working with the file system are located in the **System.IO** namespace.
- All the classes for working a Microsoft SQL Server database are located in the **System.Data.SqlClient** namespace.
- Before you can use a class in a page, you must indicate the namespace associated with the class. There are multiple ways of doing this.

Understanding Namespaces

- First, you can fully qualify a class name with its namespace.
- For example, because the File class is contained in the System.IO namespace, you can use the following statement to check whether a file exists:

System.IO.File.Exists("SomeFile.txt")

Understanding Namespaces

- Specifying a namespace each and every time you use a class can quickly become tedious. A second option is to import a namespace.
- You can add an <%@ Import %> directive to a page to import a particular namespace:

<%@ Import Namespace="System.Net.Mail" %>

or

Imports System.Net.Mail

Understanding Namespaces

- If you discover that you are using a namespace in multiple pages in your application, then you can configure all the pages to recognize the namespace only once.

- A web configuration file is a special type of file that you can add to your application to configure your application.

Understanding Assemblies

- An assembly is the primary unit for version control, deployment and security permissions in the .NET Framework.
- Each assembly is stored as an .exe or .dll file.
- There are two types of assemblies:

  private and

  shared.

Understanding the Common Language Runtime

- The second part of the .NET Framework is the Common Language Runtime (CLR).
- The Common Language Runtime is responsible for **executing the application code**.
- When you write an application for the .NET Framework with a language such as Visual Basic .NET or C#, your source code is never compiled directly into machine code.
- Instead, the Visual Basic or C# compiler converts your code into a special language named CIL.

Understanding the Common Language Runtime

- When your application actually executes, the CIL code is "just-in-time" compiled into machine code.

- Normally, your entire application is not compiled from CIL into machine code. Instead, **only the methods** that are actually called during execution are compiled.

More about CLR

- The CLR manages execution of .NET programs
- It coordinates code execution, security, debugging and other aspects of the execution process such as memory management
- The CLR includes the Common Type System
  - all .NET applications use the same data types regardless of the programming language used.

Understanding the ASP.NET Page Structure

An ASP.NET page has:

- Directives
- Server-side Code [optional]
- Layout

Differentiating In-Line Coding and Code-Behind Programming

- In-Line coding contains all the code and markup in a single file.
- The code-behind programming model provides clean separation between the client-side and the server-side code.
- In addition, the code-behind programming model adds another file to the web page called the code-behind page, which contains the server-side code.

Dynamic Pages: Stateless

- A dynamic page is generated each time it is called
- The same page may be posted back to the server for processing.
- Nevertheless, the page itself is stateless, i.e., it will not maintain the value of a variable between each loading of the page

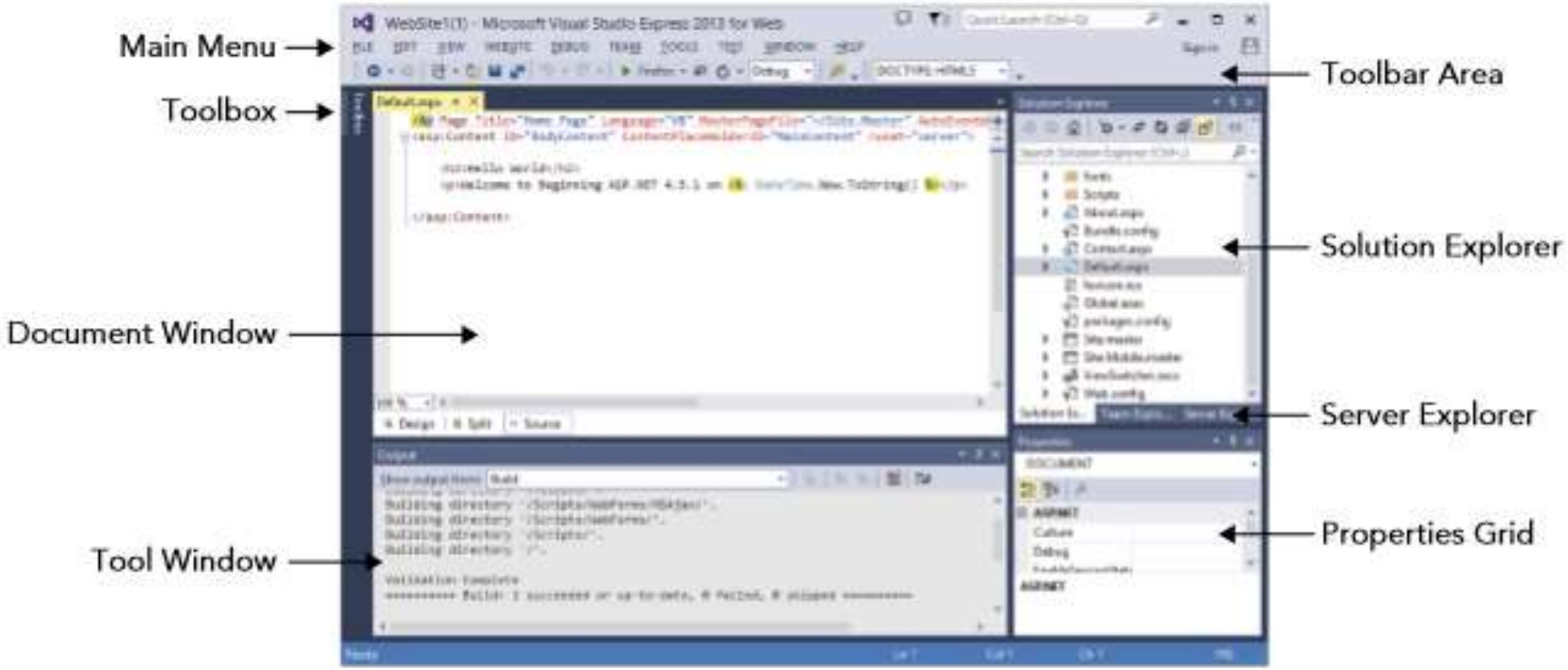
More About State

- "State" refers to the values of a variable or object
- Variables in a web page is stateless meaning that they do not "remember" their previous values
- Methods used to maintain a variable's state include
  - view state
  - session or application state object
  - cookie

Working with Intrinsic Objects in ASP.NET

- The `HttpApplication` Object
- The `HttpRequest` Object
- The `HttpResponse` Object
- The Session Object
- The `Server` Object

Visual Studio IDE



Demo: welcome.aspx

- Let's create our first Web page using Visual Studio
  1. Modify title of the page
  2. Add a heading <h2>
  3. Look at the page in Design and Split modes
  4. To add controls to the page, you can drag and drop them from the **Toolbox** onto the Web Form in **Design** mode.

     Add a `Label` control from the *Toolbox*
  5. Change ID of the `Label` control
  6. Change some physical properties of the `Label` control
  7. Add a `Page_Load` event to set the `Text` property of the `Label` control to "Welcome to Visual Studio" (to add in code-behind)

Demo: welcome.aspx

- Modify the **Title** property in the **Properties** window (F4)
- Like the Web Form itself, each control is an object that has properties, methods and events: change backcolor (black) and forecolor (yellow) of the Label
- Page Directive <%@ … %>
- AutoEventWireup attribute
- The Label generates a tag

Demo: ***Running the Program***

- You can view the Web Form several ways.
  - You can select **Debug > Start Without Debugging**, which runs the application by opening it in a browser window.
  - To debug your application, you can select **Debug > Start Debugging** (**F5**) You cannot debug a web application unless debugging is explicitly enabled by the **web.config** file.
  - To view a specific ASPX file, you can right click either the Web Forms Designer or the ASPX file name and select **View In Browser**.
  - Finally, you can run your application by opening a browser window and typing the web page's URL in the **Address** field.

Demo: Event Handling [datetime.aspx]

- Let's create another Web page using Visual Studio
  1. Add a `Button (btnDateTime)` and a `Label` control (lblresult)
  2. To create this click event handler, double click the Button on the Form.
  3. Note: an empty event handler is created
  4. Set the `Text` property of the `Label` control with the current date time.

- To add an event handler, alternatively in markup (aspx) file: (tedious process though)
  1. Add a `onclick="btnDateTime"` property to the `Button` control.
  2. Add a function `btnDateTime` to the page class in the code behind.

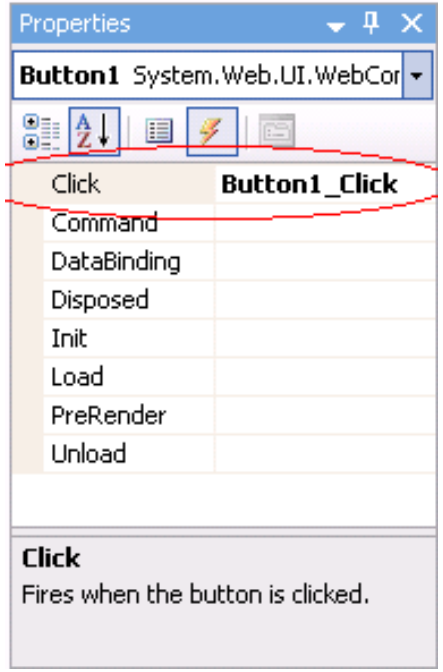
Event Handling

- By convention, the event-handler method name is set to *controlid_eventName* (e.g., `btnDateTime_Click`).
- Each event handler receives two parameters when it is called:
  - An object reference named `sender`—a reference to the object that generated the event.
  - A reference to an object of type `EventArgs`, which contains additional information about the event.

```
Sub btnDateTime_Click(ByVal sender As Object, ByVal e As EventArgs)
        lblmessage.Text = DateAndTime.Now.ToString()
End Sub
```

Other Ways to Create Event Handlers

- Typically, controls can generate many different types of events.
- Clicking the **Events** icon (the lightning-bolt icon) in the **Properties** window, displays all the events for the selected control.



Recap

- List and explain two types of assemblies?
- What are the three important parts of an ASP.NET web page?
- As compared to in-line coding, what are the advantages of code-behind programming model?
- Give three ways how to invoke namespaces.
- How CLR works?
- What is FCL?
- What do you understand by the term “Stateless”?
- Explain the importance of ViewState in a Web Page and how it works?
- Name the file used to add configuration in a website.