Introducing ASP.NET

Asst. Prof. Dr. Özgü Can

.NET

- Web-site-building tool
- Introduced to world in 2002
- Powerful
- Flexible
- Simpler
- Software platform for building systems on
  - Windows family of operating systems
  - numerous non-Microsoft OS:
    - Mac OS X
    - various Unix/Linux distributions

The Seven Pillars of ASP.NET

- ASP.NET
  - is integrated with the .NET framework
  - is compiled, not interpreted
  - is multilanguage
  - is hosted by the common language runtime
  - is object-oriented
  - supports all browsers
  - is easy to deploy and configure

#1
ASP.NET is Integrated with the .NET Framework

- Each one of the thousands of classes in the .NET Framework is grouped into a logical, hierarchical container → *namespace*
- Different namespaces provide different features.
- The .NET namespaces offer functionality for nearly every aspect of distributed development from message queuing to security. This massive toolkit is called the *class library*.
- .NET gives the same tools to web developers that it gives to rich client developers.

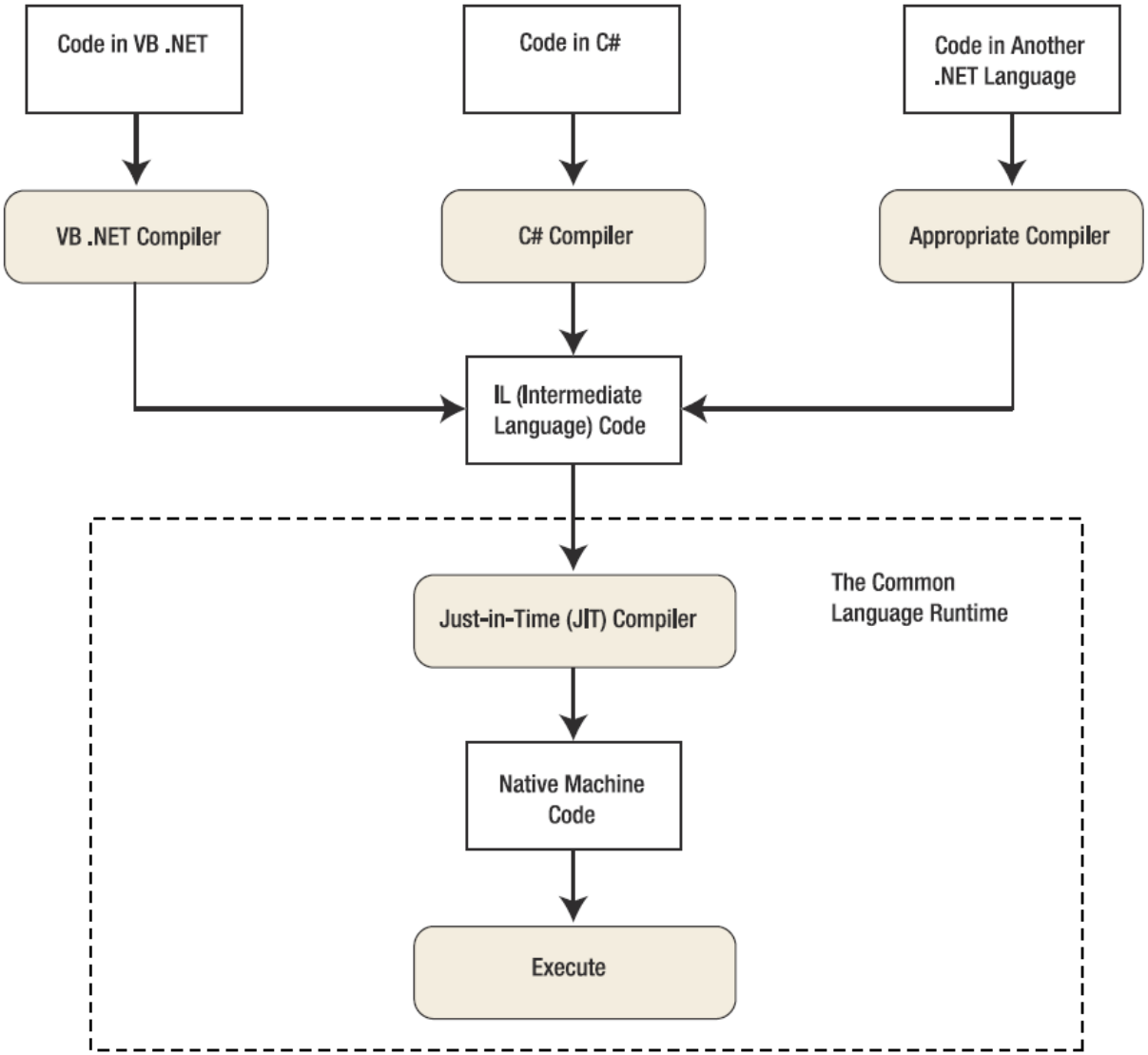
#2
ASP.NET is Compiled, not Interpreted

- ASP.NET applications are always compiled.
- .NET has 2 stages of compilation:
  1. The C# code is compiled into an intermediate language → *Microsoft Intermediate Language (MSIL)* or *(IL)*
     - may happen automatically when the page is first requested, or you can perform it in advance (*precompiling*). The compiled file with IL code is an assembly.
     - Language-interdependent.

#2
ASP.NET is Compiled, not Interpreted

- 2 stages of compilation:
  2. Happens just before the page is actually executed.
     - The IL code is compiled into low-level native machine code. This stage is known as *just-in-time (JIT)* compilation.
     - It takes place in the same way for all .NET applications

Compilation in an ASP.NET web page



#2
ASP.NET is Compiled, not Interpreted

- .NET compilation is decoupled into two steps in order to offer developers the most convenience and the best portability.
  - Before a compiler can create low-level machine code, it needs to know what type of operating system and hardware platform the application will run on (for example, 32-bit or 64-bit Windows).
  - By having two compile stages, you can create a compiled assembly with .NET code and still distribute this to more than one platform.

#2
ASP.NET is Compiled, not Interpreted

- JIT compilation probably wouldn't be that useful if it needed to be performed every time a user requested a web page from the site.
- ASP.NET applications don't need to be compiled every time a web page is requested.
  - The IL code is created once and regenerated only when the source is modified.
  - Similarly, the native machine code files are cached in a system directory that has a path like:

    C:\Windows\Microsoft.NET\Framework\[Version]\Temporary ASP.NET Files

#2

ASP.NET is Compiled, not Interpreted

- The actual point where your code is compiled to IL depends on how you're creating and deploying your web application.
  - If you're building a web project:
    - the code is compiled to IL when you compile your project.
  - If you're building a lighter-weight projectless website:
    - the code for each page is compiled the first time you request that page.
- Either way, the code goes through its second compilation step (*from IL to machine code*) the first time it's executed.

#2
ASP.NET is Compiled, not Interpreted

- ASP.NET includes precompilation tools that you can use to compile your application right down to machine code once you've deployed it to the production web server.
  - This allows you:
    - to avoid the overhead of first-time compilation when you deploy a finished application
    - to prevent other people from tampering with your code

#3
ASP.NET is Multilanguage

- No matter what language you use, the code is compiled into IL.
- Common Language Runtime (CLR)
- IL is the language of .NET, and it's the only language that the CLR recognizes.
- Obfuscator → Dotfuscator
  - recompilation system for .NET applications

```
namespace HelloWorld
{
    public class TestClass
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```

IL Disassembler → *C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin\ildasm.exe*

```
.method private hidebysig static void  Main(string[] args) cil managed
{
    .entrypoint
    // Code size       13 (0xd)
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr    "Hello World"
    IL_0006:  call     void [mscorlib]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
} // end of method TestClass::Main
```

IL code generated from C# version

```
Imports System

Namespace HelloWorld
    Public Class TestClass
        Shared Sub Main(args() As String)
            Console.WriteLine("Hello World")
        End Sub
    End Class
End Namespace
```

Console application in Visual Basic code

#3
ASP.NET is Multilanguage

- The CLR expects all objects to adhere to a specific set of rules so that they can interact.
- The Common Language Specification (CLS) is this set of rules.
  - It defines many laws that all languages must follow, such as primitive types, method overloading, and so on.
  - The CLS gives developers, vendors, and software manufacturers the opportunity to work within a common set of specifications for languages, compilers, and data types.
- Any compiler that generates IL code to be executed in the CLR must adhere to all rules governed within the CLS.

#4
ASP.NET is Hosted by the CLR

- ASP.NET engine runs inside the runtime environment of the CLR.
- Benefits of CLR:
  - Automatic memory management and garbage collection
    - GC runs periodically inside the CLR.
  - Type safety
    - When you compile an application, .NET adds information to your assembly that indicates details such as the available classes, their members, their data types, and so on.
  - Extensible metadata
    - Metadata describes your code and allows you to provide additional information to the runtime or other services.
  - Structured error handling
    - Allows you to organize your error-handling code logically and concisely.
    - You can create separate blocks to deal with different types of errors.
  - Multithreading
    - The CLR provides a pool of threads that various classes can use.
      - call methods, read files, or communicate with web services asynchronously, without needing to explicitly create new threads.

#5
ASP.NET is Object-Oriented

- Not only does your code have full access to all objects in the .NET Framework, but you can also exploit all the conventions of an OOP environment.
  - You can create reusable classes, standardize code with interfaces, extend existing classes with inheritance, and bundle useful functionality in a distributable, compiled component.
- ASP.NET offers server controls as a way to abstract the low-level details of HTML and HTTP programming.
  - Instead of forcing the developer to write raw HTML manually, the control objects render themselves to HTML just before the web server sends the page to the client.

#5
ASP.NET is Object-Oriented

```
<input type="text" id="myText" runat="server" />
```

With the addition of the runat="server" attribute, this static piece of HTML becomes a fully functional server-side control that you can manipulate in C# code.

```
void Page_Load(object sender, EventArgs e)
{
    myText.Value = "Hello World!";
}
```

ASP.NET Web Controls

- ASP.NET web control tags always start with the prefix *asp:* followed by the class name.

```
<asp:TextBox id="myASPText" Text="Hello ASP.NET TextBox" runat="server" />
<asp:CheckBox id="myASPCheck" Text="My CheckBox" runat="server" />
```

- You can interact with these controls as:

```
myASPText.Text = "New text";
myASPCheck.Text = "Check me!";
```

#6
ASP.NET Supports all Browsers

- Different browsers, versions, and configurations differ in their support of XHTML, CSS, and JavaScript.
- ASP.NET encourages developers to ignore considerations and use a rich suite of web server controls.
- You don’t need any extra coding work to support both types of client.

#7
ASP.NET is Easy to Deploy and Configure

- Deploying a completed application to a production server.
  - Not only do the web-page files, databases, and components need to be transferred, but components need to be registered and a slew of configuration settings need to be re-created.
- ASP.NET simplifies this process considerably.
  - You simply need to copy all the files to a virtual directory on a production server.
  - As long as the host machine has the .NET Framework, there are no time-consuming registration steps.

#7
ASP.NET is Easy to Deploy and Configure

- Distributing the components:
  - All you need to do is copy the component assemblies along with your website files when you deploy your web application.
  - Because all the information about your component is stored directly in the assembly file metadata, there's no need to launch a registration program or modify the Windows registry.
  - As long as you place these components in the correct place (*the Bin subdirectory of the web application directory*), the ASP.NET engine automatically detects them and makes them available to your web-page code.

#7
ASP.NET is Easy to Deploy and Configure

- Configuration:
  - If you need to transfer security information such as user accounts and user privileges:
    - ASP.NET makes this deployment process easier by minimizing the dependence on settings in IIS(Internet Information Services).
    - Most ASP.NET settings are stored in a dedicated web.config file.
      - The web.config file is placed in the same directory as your web pages.
      - It contains a hierarchical grouping of application settings stored in an easily readable XML format that you can edit using nothing more than a text editor such as Notepad.
    - When you modify an application setting, ASP.NET notices that change and smoothly restarts the application in a new application domain (keeping the existing application domain alive long enough to finish processing any outstanding requests).
    - The web.config file is never locked, so it can be updated at any time.

LINQ

- Language Integrated Query
- A set of extensions for the C# and Visual Basic languages.
- It allows you to write C# or Visual Basic code that manipulates in-memory data in much the same way you query a database.
- Defines 40 query operators.
  - Select, from, in, where, orderby, etc..

LINQ

- LINQ to Objects
  - allows you to take a collection of objects and perform a query that extracts some of the details from some of the objects.
  - isn’t ASP.NET-specific, you can use it in a web page in exactly the same way that you use it in any other type of .NET application.

LINQ

- LINQ to DataSet
  - querying an in-memory DataSet object
- LINQ to XML
  - works on XML data
- LINQ to Entities
  - allows you to use the LINQ syntax to execute a query against a relational database
  - creates a properly parameterized SQL query based on your code
  - executes the query when you attempt to access the query results

ASP.NET AJAX

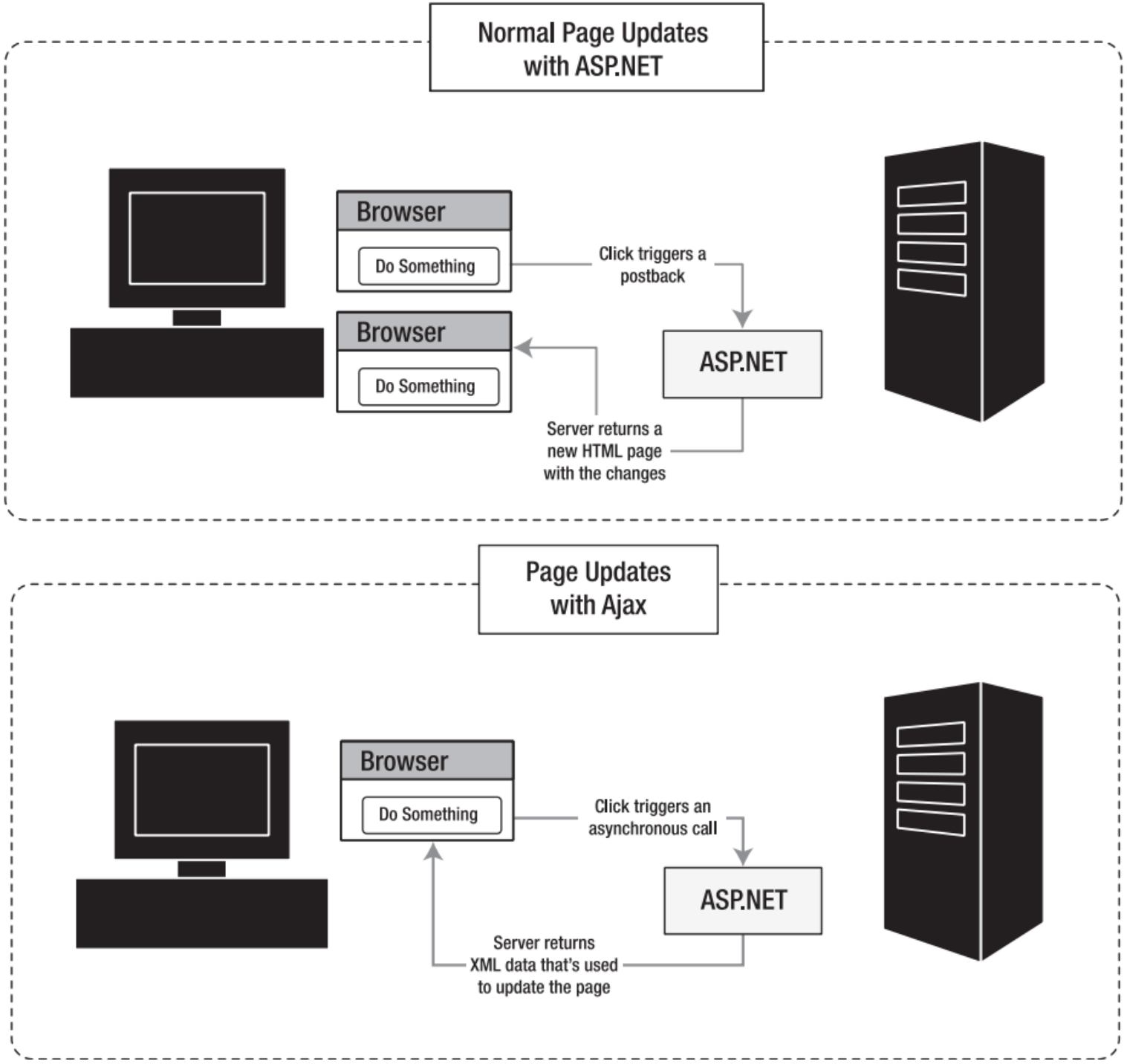
- Traditional ASP.NET code does all its work on the web server.
  - Every time an action occurs in the page the browser needs
    - to post some data to the server
    - get a new copy of the page
    - refresh the display
  - This process, though fast, introduces a noticeable flicker.

ASP.NET AJAX

- In ASP.NET, many of the most powerful controls use a healthy bit of *JavaScript* to support client-side scripting language.
- Developers use server controls.
- The solution works, but it isn’t seamless.
- Asynchronous JavaScript and XML (AJAX)

ASP.NET AJAX

- Client-side tecnique
  1. Allows your page to call the server and update its content without triggering a complete postback.
  2. An Ajax page uses client-side script code to fire an asynchronous request behind the scenes.
  3. The server receives this request, runs some code, and then returns the data your page needs (often as a block of XML markup).
  4. Finally, the client-side code receives the new data and uses it to perform another action, such as refreshing part of the page.
- AJAX allows you to create pages that work more like seamless, continuously running applications.


Normal Page Updates with ASP.NET
Browser
Do Something
Click triggers a postback
Browser
Do Something
ASP.NET
Server returns a new HTML page with the changes
Page Updates with Ajax
Browser
Do Something
Click triggers an asynchronous call
ASP.NET
Server returns XML data that's used to update the page

ASP.NET MVC

- Model View Controller
- Application is separated into three logical parts:
  - The *model* includes the application-specific business code
    - data-access logic and validation rules
  - The *view* creates a suitable representation of the model by rendering it to HTML pages.
  - The *controller* coordinates the whole show
    - handling user interactions, updating the model, and passing the information to the view

ASP.NET MVC

- Suits to the Web??
- Extra effort with no clear payoff??

MVC

- Test-driven development
- Control over HTML markup
- Control over URLs

Web Forms

- Rapid application design
- A high level model that manages state for you
- A range of rich web controls

Silverlight

- Allows a variety of browsers on a variety of operating systems to run true .NET code.
  - Works through a browser plug-in.
  - Provides a subset of the .NET Framework class library.
- Silverlight is all about client code.
  - It allows you to create richer pages than you could with HTML, DHTML, and JavaScript alone.
    - draw sophisticated 2D graphics
    - animate a scene
    - play video and other media files

Silverlight

- is a good choice:
  - creating a mini-applet, like a browser-hosted game
  - adding interactive media and animation to a website
- isn't a good choice:
  - for tasks that require server-side code, such as:
    - performing a secure checkout in an e-commerce shop
    - verifying user input
    - interacting with a server-side database
  - to replace basic ingredients in a website with Silverlight content
    - Users without the Silverlight plug-in won't be able to see your button or interact with it.