BTI420 - Web Programming on Windows

Lecture 1

Agenda

- BTI420 course overview
- Introducing the development environment:
  - Visual Studio 2015
  - Web browsers
  - Browser developer tools
  - HTTP inspector (e.g. web-sniffer.net, Fiddler app)
- Web app development overview and context
- Introduction to ASP.NET MVC
- Get started with the C# language and the .NET Framework
- Writing code, writing classes
- Modelling data entities with classes
- Getting started with string-number conversions

Textbook coverage

Read these before today's session:

- Foreword
- Introduction
- Chapter 1
  - Optional – 'history' info on pages 4, 5, and 6
  - Optional – "ASP.NET Web API" info on pages 7, 8, and 9; we'll revisit this later in the course
  - Note – on page 16, in the "Installing ASP.NET MVC 5" section, it states that "MVC 5 is included with Visual Studio 2013, so there's nothing to install". It's the same situation in Visual Studio 2015 – it's included and built in.

Review – web app

- A 'web app' is a program that runs on a web server.
- A web server uses the HTTP application protocol to communicate with users.
- A 'web browser' is typically used to access the web app.
- As a result, responses from a web app are typically (composed as) HTML. However, other internet media types can be in a request or response (image, CSS, JavaScript, etc.).
- Static vs dynamic web apps:
  - static web app: delivered without any server-side alteration of the HTML, CSS, or JavaScript content.
  - dynamic web app: using server side scripts as well as data access technologies: ASP, ASP.NET, PHP, JSP, ...

Web app development overview

- In the last decade or so, two web app dev frameworks have dominated:
  - ASP.NET
  - PHP
- Other than these two, you'll see some Ruby on Rails and Java solutions (e.g. Spring) out there. What about the future? Who knows… but the trend is towards full-stack JavaScript.
- The Model-View-Controller (MVC) software design pattern is now the de facto development approach for web apps.
- Each dominant web app dev framework – ASP.NET and PHP – offer the ability to use the MVC pattern.

Introduction to ASP.NET MVC

- **ASP.NET** is a server-side web application framework and runtime environment for dynamic web apps that run on the Microsoft Web Platform.
- **ASP.NET MVC** is a framework for building web apps that conform to the model-view-controller (MVC) design pattern.
- The two main things that new devs need to know about ASP.NET MVC:
  1. The front controller pattern, in that a URL does not reference a file system resource. e.g.
  2. The app dev process relies on programming conventions
- If you have trouble learning both of these, you will not be successful in this course.

How does the front controller pattern work?

1. When you request a resource in an ASP.NET MVC app, the ASP.NET runtime receives the request
2. The URL is inspected, because it will determine what happens next
3. The request is routed to a specific method in a specific "**controller**" class
4. The method generates some data, and passes it to a "**view**" for rendering

- Regarding programming conventions, in a default "file > new" ASP.NET MVC project, the HomeController is the web app's entry point. The URI to methods in HomeController use the /home/path.

What is an ASP.NET application?

The following was adapted from this MSDN reference document:

- ***An ASP.NET application*** is the collection of all source code files and resources (e.g. images) within a specific directory, and its subdirectories, on a single web server.
  - Initially, a browser/client requests any URL in an ASP.NET application,
  - the web server creates an instance of an HttpApplicationState class. A reference to this instance is available within the app via the ApplicationInstance property.
  - The lifetime of the instance is twenty (20) minutes.
  - The lifetime is reset with each request to a URL in the app.

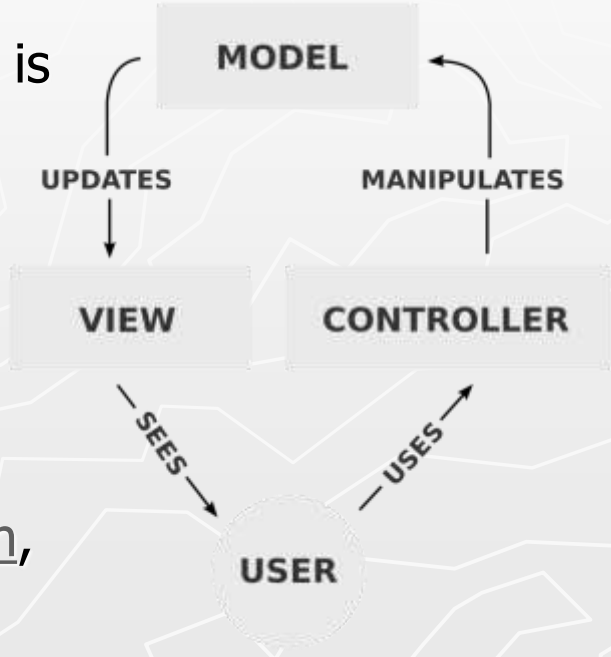
What is an ASP.NET application?

- The Global.asax.cs source code file defines the "MvcApplication" class, a subclass of System.Web.HttpApplication.
- You can add code to this class, to (for example) initialize software components and data for the app, and to customize behaviour when servicing a request. While in this class:
  - The ApplicationInstance property refers to the app's code and execution environment
  - The Application property refers to the state of the app, including its data
- You will typically use the Application property most often. In other source code files, you must get a reference to the application, using System.Web.HttpContext.Application. (The System.Web namespace is always added in a using directive, so we can simply use "HttpContext.Application".)

What is an application domain (aka problem domain)?

- We will begin using the phrase "application domain" (often shortened to "app domain") in this course. An app domain is the area of knowledge that is implemented in a software application.

- In an ASP.NET MVC application, an app domain is materialized in some standard software components, such as models, views, and controllers. In addition, it will include software that implements business logic and rules, workflow, and persistence management.



- Wikipedia has useful articles on problem domain, domain model, and conceptual model.

MVC – what is a model?

- A model holds and manages your app's data.
  - Textbook definition: A set of classes that describes the data you're working with, as well as the business rules for how the data can be changed and manipulated.
- We write classes that model the data:
  - Data that's persisted in a store is modelled by design **model classes**.
  - Data that is delivered to the user, or gathered from the user, is modelled by **view model classes**.
- You will learn much more about view model classes in the next few sessions.
- In this course, we use view model classes extensively.
- The textbook? Not so much. This is a key difference between this course and the textbook.

MVC – what is a view?

- A view is a source code file that contains user interface code.
  - Textbook definition: Defines how the application's UI will be displayed.

- The user interface target is a web browser, so a view contains HTML markup and code expressions that place data into the markup.

MVC – what is a controller?

- A controller is a class that services requests from users.
  - Textbook definition: A set of classes that handles communication from the user, overall application flow, and application-specific logic.
- An incoming request from a user is routed to a specific method in the controller.
- The method generates some data (from the model), and passes the data to the view for rendering.

Introduction to the .NET Framework

What is the .NET Framework?

- The .NET Framework (.NET is pronounced "dot net") is a set of technologies for developing and running software.
- As a result, its purpose is to enable software developers to do great things, and it is therefore targeted to software developers. However, the result of using the .NET Framework – software – has almost universal application in life today. Please keep this in mind:
  - Software developers will care about things like the .NET Framework, but other information technology industry professionals may care less about it. People from other parts of life and society may not care at all.
- Work on what became the .NET Framework began in 1998, when Brad Abrams led a diverse group of Microsoft programmers to create what was then known as "Next Generation Windows Services".
  - Like ASP.NET above, "version 1" was released in January 2002.
  - The current "version" of the .NET Framework is known as "version 4.6".

.NET Framework

.NET Framework components

- There are two main components of the .NET Framework:
  1) The .NET Framework Class Library, and
  2) the Common Language Runtime.
- The .NET Framework Class Library (aka the FCL) is a comprehensive,object-oriented collection of reusable types that developers can use to create software.
- The Common Language Runtime (aka the CLR) is an execution environment, providing memory, process, and thread management.

How do I get the .NET Framework?

- The .NET Framework, which is free (from cost), is included in all modern versions of Windows (back as far as – ugh – Windows 98).
  - Typically, you don't have to do anything to get the .NET Framework, other than making sure Windows Update is turned on.
- Interesting side note:
  1. Through the efforts of Scott Guthrie, source code for the .NET Framework libraries is available.
  2. The CLR is Microsoft's implementation of the Common Language Infrastructure (CLI) ECMA and ISO standards, which define an execution environment for program code.

Intro to the C# programming language

What is C#, and where did it come from?

- The C# programming language is an easy-to-use, powerful, object-oriented language created in 1999 and 2000 by Microsoft, and standardized worldwide through ECMA and ISO. Anders Hejlsberg is the lead architect and principal designer of C#.
- It may surprise you to learn that it was basically Anders and three others who designed and created the language. Its design was intended to be close to C++, but include deep object-orientation, and component orientation.

Note:

- C# is the name of the programming language
- The .NET Framework is the name of the class library and execution runtime. The classes are organized into namespaces. Your app will link to one or more namespaces.
- So in summary, you write code in the C# language, and use .NET Framework classes.

What's the origin of the name, C#?

What's the origin of the name, C#?

- Of geeky interest, your professor offers the following collection of facts and educated guesses about the origin of how C# got its name:
- Around 1960, the programming language Algol 60 was created (you can consider this to be the language "A" for our purposes here)
- In the UK, an extended version of Algol 60 was created, and was referred to as the Combined Programming Langauge (CPL)
- This terminology was refined to "basic CPL", or, BCPL
- Dennis Ritchie, while at Bell Labs in the late 1960's, transformed BCPL into a programming language named B
- Dennis and Brian Kernighan later (early 1970's) enhanced its ideas into the C programming language
- Around 1980, Bjarne Stroustrup, also with Bell Labs, created C++
- Taking hints from the ++ iterator, C++ was "one up" from C
- Around the year 2000, Anders Hejlsberg of Microsoft created C#
- Taking hints from music, C# was "one semitone higher" [than C].

What can you expect?

- In this BTI420 course, if you pass, you will become proficient at an entry level in C#. Rest assured that you can develop your concepts and skills further to create software that scales down to the tiniest of devices, all the way up to the most powerful computing systems.

Learning C# – The top ten list

Here is the "top ten" list of questions to ask when learning C#:

1. How do I create/edit/save the source code? What editor do I use? Where do I store the source code? What is the source code file naming convention?
2. How do I build/compile and then execute/run my program? Is there a compiler? Is there a host execution (aka "runtime") environment?
3. What is the program's entry point? Syntactically, what coding convention must I follow?
4. What data types can I use? Are they categorized (e.g. value/stack, reference/heap), and if so, what do I need to know about their characteristics (e.g. size, initial value, precision, convertibility, etc.)?

… **…**

* **Please check this link for all questions and answers.**

* Also scan this course's "Resources" page to learn about the Microsoft Developer Network.

Writing code, writing classes

- You will write many classes in a web app.
- The class is the most fundamental container, or building block, for your code.
- You have already seen that a controller is a class.
- You will also use classes to define **data** objects (i.e. classes that are things), as well as objects with **behaviour** (i.e. classes that do things), or both data and behaviour.
- Code for a class is placed within a namespace. A class will include members. Common members include:
  - Private fields, not visible outside the class
  - Constructors, for creating a new instance of the class
  - Properties, for holding publicly-available data
  - Methods, public or private, for behaviour
- And, there are a few other kinds of members that can be added.
- **Do not nest** a class inside another class.

A Class Example

```
public class CalendarEntry
{
    public string Day { get; set; } // public property (shorthand).

    //public DateTime Date { get; set; } // use this line or the next 6 lines.
    private DateTime date;  // private field
    public DateTime Date    // Public property
    {
        get { return date; }
        set { date = value; }  // pass in: DataTime
    }

    // Public method also exposes date field safely. // pass in: string
    public void SetDate(string dateString)
    {
        DateTime dt = Convert.ToDateTime(dateString);

        // Set some reasonable boundaries for likely birth dates.
        if (dt.Year > 1900 && dt.Year <= DateTime.Today.Year) { date = dt;  }
        else
            throw new ArgumentOutOfRangeException();
    }
}
```

Modelling data entities with classes

- Modelling data entities with classes is a simple task, as it is in other languages and frameworks.

- A model class needs properties, maybe one or more methods, and maybe one or more constructors.

Getting started with string-number conversions

String/string

- The String class is often used. It is a reference type, and is **immutable**. The editor, compiler, and runtime make it easy to work with strings.
- Declaring a string is simple, and does not require the "new" keyword (unlike the declarations for other reference types):

```
string myName = "Peter";
```

- And, although a string is immutable after it has been created, the compiler and runtime **enable you to mutate (change)** an existing string's value by using a natural and comfortable syntax. Using the myName instance from above, let's change it:

```
myName = "Prof. Peter McIntyre";
```

Numbers

- Numbers in the .NET Framework are typically value types.
- Many numeric types are available, but we typically use **int** and **double** in C#. All numeric types are aliases of .NET type classes.

Conversions

Why do you care about converting between strings and numbers?

- Well, think about it: What data type is HTML, which appears in a browser?

  Text (string).
- Therefore, if you are "round-tripping" numbers to and from a web page, you must convert them to and from strings.

Converting a number to a string: the number's "ToString()" method:

```
string foo = 123.ToString();
```

- Some number types have ToString() overloads, which permit you to specify a preset (common) or custom number format string.

Converting a string to a number: using the methods in System.Convert class, which has methods that help. For example:

```
// The "using System;" is always present at the top of the source code file, so
int bar = Convert.ToInt32("123");
```

- This is an unsafe conversion, because it's possible that the string cannot be converted. If that's the situation, this statement will cause an exception (a runtime error).

"Safe" conversion

- Using the Int32.TryParse() method:

```
// Create a temporary variable
int foo;
// Attempt to convert…
bool isNumber = Int32.TryParse("123", out foo);
```

  - The return result of TryParse() is a bool.
  - If the conversion is successful:
    - The return result is "true", and
    - The variable "foo" holds the converted string-to-number
  - If it's not successful, the return result is false, and foo is zero.
- Using Double.TryParse() method - for double values.

… …

- How can you determine whether a string variable is null or empty? Use this test:

```
// Assume that "stringVariable" may or may not hold some content…
bool isProblem = string.IsNullOrEmpty(stringVariable);
```

Resourceful Links

- How to create and run a console application?

  https://msdn.microsoft.com/en-us/library/k1sx6ed2.aspx

- How to create a simple ASP.NET web app?

  https://docs.asp.net/projects/mvc/en/latest/getting-started/first-mvc-app/start-mvc.html#install-visual-studio-and-asp-net

Thank You!