**INT422 – Lecture 1**

Welcome to our first session together in int422.

**Reminder: What do I expect from you?**

Before a Wednesday class, I expect you to prepare for the class. This means:

* Read and study the class notes page
* Read and study the linked documents
* Make your own notes, including questions that you have

In other words, do not come into the classroom expecting somehow to soak up knowledge like a sponge. You need to prepare before class, so that you understand the topics and their context.

Also, at the beginning of each Wednesday class, beginning in Week 2, there will be a test at the beginning of the timeslot. The questions are based on the new topics this week.

In the Wednesday class, I expect you to be an engaged and actively-learning participant. This means:

* Listening effectively
* Asking and answering questions
* Writing notes
* Doing the in-class exercises and activities

.

**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.

.

**Topic coverage today**

INT422 course overview

Introducing the development environment:

* Visual Studio 2015
* Web browsers
* Browser developer tools
* HTTP inspector (e.g. web-sniffer.net, Fiddler app)

Learning resources (web page)

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

.

**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.).

.

**Web app development overview and context**

In the last decade or so, two web app dev frameworks have dominated:

* ASP.NET
* [PHP](http://en.wikipedia.org/wiki/Php)

Other than these two, you’ll see some [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) and Java solutions (e.g. [Spring](http://en.wikipedia.org/wiki/Spring_MVC)) out there. What about the future? The trend is towards full-stack JavaScript.

The [Model-View-Controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) software design pattern is now the de facto development approach for web apps.

Each dominant web app dev framework – ASP.NET and PHP – offers the ability to use the MVC pattern.

.

**Introduction to ASP.NET MVC**

ASP.NET is a programming and runtime environment for 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. It uses the [front controller pattern](http://en.wikipedia.org/wiki/Front_Controller_pattern), in that a URL is parsed and processed, and typically does not reference a file system resource
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. A routing module inspects the URL, and will determine what happens next
3. The request is typically routed to a specific method (i.e. a function) in a specific “**controller**” class (which may use data in the “**model**“)
4. The method generates some data, and passes it to a “**view**” for rendering by a view engine

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](https://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate%28v=vs.110%29.aspx) 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.

The first time a browser/client requests any URL in an ASP.NET application, the web server creates an instance of an [HttpApplicationState class](https://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate%28v=vs.110%29.aspx). A reference to this instance is available within the app via the ApplicationInstance property. The lifetime of the instance is twenty (20) minutes, by default (but it can be changed). The lifetime is reset with each request to a URL in the app.

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

Sometimes, you will use the Application property. 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](http://en.wikipedia.org/wiki/Problem_domain), [domain model](http://en.wikipedia.org/wiki/Domain_model), and [conceptual model](http://en.wikipedia.org/wiki/Conceptual_model_%28computer_science%29).

.

**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.

View model classes are very important, and the best thing that you will learn in this course.

.

**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.

.

**Get started with the C# language and the .NET Framework**

Please read these brief documents:

[Introduction to the .NET Framework](https://scs.senecac.on.ca/~wei.song/int422/common/IntroductionToThe.NET.html)

[Introduction to C#](https://scs.senecac.on.ca/~wei.song/int422/common/IntroductionToTheCS.html)

[Learning C# – The top ten list](https://scs.senecac.on.ca/~wei.song/int422/common/LearningCSharp-theTopTenList.html)

Also scan this course’s “[Resources](https://scs.senecac.on.ca/~wei.song/int422/int422.html#resources)” page to learn about the Microsoft Developer Network.

**Get started with Visual Studio for web app development**

We will use Visual Studio Enterprise 2015 in this course.

Your professor will demonstrate a “file > new” ASP.NET MVC project creation workflow.

Then, we will get a tour of your project’s source code items, and Visual Studio features for web app development.

**Writing code, writing classes**

You will write many classes in a web app.

The [class](http://msdn.microsoft.com/en-us/library/ms173109.aspx) 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](http://msdn.microsoft.com/en-us/library/0d941h9d.aspx). A class will include [members](http://msdn.microsoft.com/en-us/library/ms173113.aspx). Common members include:

* Private [fields](http://msdn.microsoft.com/en-us/library/ms173118.aspx), not visible outside the class
* [Constructors](http://msdn.microsoft.com/en-us/library/ace5hbzh.aspx), for creating a new instance of the class
* [Properties](http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx), for holding publicly-available data
* [Methods](http://msdn.microsoft.com/en-us/library/ms173114.aspx), 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.

.

**Modelling data entities with classes**

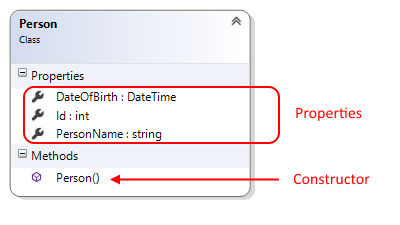
Modelling data entities with classes is a simple task, as it is in other languages and frameworks.

A model class needs properties, maybe one or more methods, and one or more constructors. It should always have a default constructor.

For example, assume that you want to model a “person”. Here is some sample code for a class that will model a person:

|  |  |
| --- | --- |
|  | **public class Person**  **{**  **// Constructor**  **public Person()**  **{**  **DateOfBirth = DateTime.Now.AddYears(-25);**  **}**    **// Properties**  **public int Id { get; set; }**  **public string PersonName { get; set; }**  **public DateTime DateOfBirth { get; set; }**  **}** |

This is how the class appears in a nifty Visual Studio feature, a [class diagram](https://msdn.microsoft.com/en-us/library/33864ckt.aspx):



Let’s look at another example. Assume that we want to model a “manufacturer”, which makes and sells “products”. This is a classic example of classes that are associated or related with each other. The association could be to-many (e.g. a manufacturer has many products), or to-one (e.g. a product is from one manufacturer). The association is coded with a property, known as a navigation property. Both associated classes have the navigation property, so that each class “points to” the other.

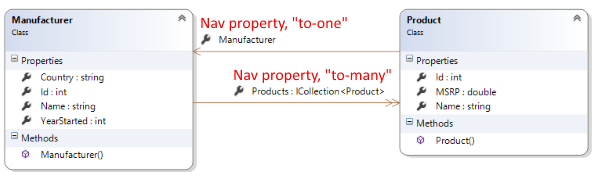
❝Definition: Navigation property

A navigation property has a data type of another class in the model.

Here is some sample code for classes that model the manufacturer-product concept:

|  |  |
| --- | --- |
|  | **public class Manufacturer**  **{**  **// Constructor**  **public Manufacturer()**  **{**  **YearStarted = 1980;**  **Products = new List<Product>();**  **}**    **// Properties**  **public int Id { get; set; }**  **public string Name { get; set; }**  **public string Country { get; set; }**  **public int YearStarted { get; set; }**    **// Navigation property**  **public ICollection<Product> Products { get; set; }**  **}**    **public class Product**  **{**  **// Constructor**  **public Product()**  **{**  **MSRP = 1.00;**  **}**    **// Properties**  **public int Id { get; set; }**  **public string Name { get; set; }**  **public double MSRP { get; set; }**    **// Navigation property**  **public Manufacturer Manufacturer { get; set; }**  **}** |

This is how the classes appear in a class diagram:



**Getting started with string-number conversions**

The String class is often used. It is a [reference type](http://msdn.microsoft.com/en-us/library/zcx1eb1e%28v=VS.100%29.aspx), and is immutable. However, 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 in the .NET Framework are typically [value types](http://msdn.microsoft.com/en-us/library/zcx1eb1e%28v=VS.100%29.aspx). Many numeric types are available, but we typically use int and double in C#.

.

**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, as data, to and from a web page, you must convert them to and from strings.

You can “convert” a number to a string with 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.

If a string contains or holds a numeric value, there is a [System.Convert](http://msdn.microsoft.com/en-us/library/system.convert.aspx) class, which has methods that help. For example:

|  |  |
| --- | --- |
|  | **// The "Convert" class is in the local scope, because "using System;"**  **// is always present at the top of the source code file**  **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).

A better approach is to use the [Int32.TryParse()](http://msdn.microsoft.com/en-us/library/f02979c7%28v=vs.110%29.aspx) 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:

1. The return result is “true”, and
2. The variable “foo” holds the converted string-to-number

If it’s not successful, the return result is false, and foo is zero.

There is a [Double.TryParse()](http://msdn.microsoft.com/en-us/library/994c0zb1%28v=vs.110%29.aspx) method that you can use for double values.

Back to strings for a moment… 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);** |