Session 1-2

MVC is all about the "C" - the M and the V are optional extras.

Client Webserver M

**C** EF DB

Render

V

"C" is for "Controller"

It is the Centre of MVC operation.

Clients do NOT visit pages, they visit Methods in controllers  
In the following example, the base website is "https://cars.example.com/"

|  |  |
| --- | --- |
| URL Address | Where it really goes |
| https://cars.example.com/  *acts the same as* https://cars.example.com/home  https://cars.example.com/home/index  BUT NOT  https://cars.example.com/index "index" is now a method that needs to be the child of a controller | method "Index" in class "HomeController" |
| https://cars.example.com/modifications  *acts the same as*  https://cars.example.com/modifications/index | method "Index" in class "ModificationsController" |
| https://cars.example.com/modifications/edit | method "Edit" in class "ModificationsController" |
| https://cars.example.com/paintjobs/racing | method "Racing" in class "PaintjobsController" |

A "web page" is the text output from a Method in a Controller

A "View" is a helper template of reusable text to save typing in our C# code

MVC gives you a lot of complex relationships where the Microsoft MVC team have already done a lot of the programming work and we make use of some of that work by following naming conventions and matching names.

For example:

https://cars.example.com/paintjobs/hotrod

The real address is method "Hotrod" in class "PaintjobsController"

The View is "Hotrod.cshtml" in project sub-folder Views\Paintjobs  
This Controller and this View connect to each other and relate   
by matching the keywords "Paintjobs" and "Hotrod".

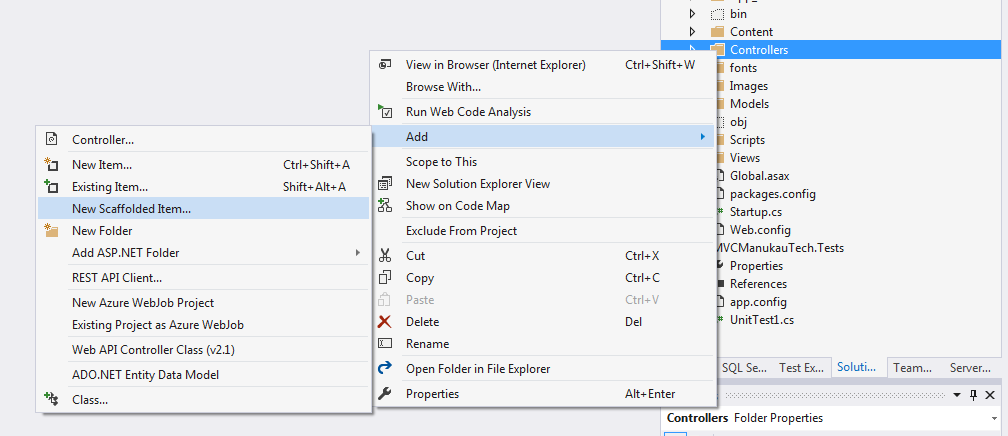
**Starting with C for Controller**

From Canvas "Week01", download and setup "Northwind\_Test01.zip" if you have not already done so.

Under the grandparent folder, double-click on "MVCManukauTech.sln" to start our basic blank website.  
(Lecturer may comment about this website)

Test-run.

In the "Solution Explorer" panel, usually on the right, right-click on the folder "Controllers"



Select "Add" --> "New Scaffolded Item"  
The screenshot above shows going direct to "Controller" as an option, but we only get this when the system remembers we have done this before.  
Then from the Popup Dialog, select **MVC Empty Controller**.  
On the prompt for a name, enter **CalcController**

We will start by adding and testing

public string Add(double a, double b)

{

double c = a + b;

return c.ToString();

}

to test this, run the app and edit the address adding: /Calc/Add?a=4&b=7

http://localhost:44384/Calc/Add?a=4&b=7

we should see ... 11

--------------------------

Try this yourselves: Feet and Inches to mm - start with:

public string MM(double f, double i)

--------------------------

Next example, quadratic equation - practice - lecturer will give you help and a response  
but you need to put in some effort first

x2 - 9x + 20 = 0 2 possible values of x   
 x = 4, x = 5

Needs 2 statements for our coding

x1 = (-b + √(b2 - 4ac))/2a

x2 = (-b - √(b2 - 4ac))/2a

Code to get results for x1 and x2

Then return with a string giving some information

public string Quad(double a, double b, double c)

{

//coding of the supplied formula goes here

string response = "The 2 solutions are " + x1 + " and " + x2;

return response;

}

We can try adding a little HTML like this:

string response =   
"The 2 solutions are <b>" + x1 + "</b> and <b>" + x2 + "</b>";

Try this in ASP.NET CORE 2.0 and HTML rendering does not happen.  
We see the raw tags on the browser screen.

To get this working we need to bring in methods and objects that send the protocol "headers" to the browser.  
These are like instructions to the browser on the actions to take with this data.

public IActionResult Quad(double a, double b, double c)

{

Response.ContentType = "text/html";

//coding of the supplied formula goes here

string response =   
 "The 2 solutions are <b>" + x1 + "</b> and <b>" + x2 + "</b>";

return Content(response);

}

--------------------------

Your exercises - with a little help including on whiteboard in class. Do responses including info like the previous example.  
NOTE - Normally I give methods long names to be "self documenting" eg "ConvertToMPG" rather than "Mpg"  
For this exercise we need to type the identifiers a lot therefore I am allowing shorter names but this is only for learning purposes and you should give longer and more detailed identifier names in other situations.

--------------------------

**Fahrenheit to Celcius - student exercise**  
Work from this formula only. You will need to add value to code this. Name your method: FtoC

c = (f - 32) \* 100 / 180

--------------------------

**Volume of a Cylinder - student exercise**  
Equation in Maths style - you need to code as a C# statement. Name your method: Cyl

--------------------------

**Car fuel economy** **- student exercise**  
- Litres/100km to miles/gallon  
Do in steps with a number of statements  
Input e (economy) which for a typical car will be 6  
  
public string Mpg(double e)

Then change litres/100km to litres/km

double eL = e / 100;

Then change litres/km to km/litre

double eKm = 1 / eL;

Then change km/litre to miles/litre

double eMilesPerLitre = eKm \* 5 / 8;

Then change miles/litre to miles/gallon

double mpg = eMilesPerLitre \* 4.5;

----------------------------------------------------------------

**CSV input to mean - student exercise**

Input would look like:

http://localhost:7151/Calc/Mean?csv=8,3,4,7,20

Note that there are no spaces in these "querystrings" where we send data added on to the URL

Hint - use the Split method to split this csv string into individual strings.  
You can then convert and work out the mean from there.

----------------------------------------------------------------

**JSON quick intro (lecturer-led)**

JSON gives more detail and we consider it to be more modern than CSV.

Begin with the introduction from W3Schools. Read the first 4 pages:  
<https://www.w3schools.com/js/js_json_intro.asp>

With JSON we use {} for objects - often these are 1 row of data from a database table.  
We use square brackets for arrays.  
The above "Mean" example done again as "JSONMean" with JSON has this test input

http://localhost:7151/Calc/JSONMean?json=[8,3,4,7,20]

To work with JSON in C#  
At the top of your controller add this statement to the other "using" statements:  
using Newtonsoft.Json;

Ref is

<http://www.newtonsoft.com/json>

<http://www.newtonsoft.com/json/help/html/SerializingJSON.htm>

**Useful methods are**

To convert from JSON string to an array, list, collection or other object  
JsonConvert.DeserializeObject<T>()

eg

Product deserializedProduct = JsonConvert.DeserializeObject<Product>(JSONString);

where T is the type of object you want to convert to. You will need this for your "Mean" exercise.

The opposite is "SerializeObject" for converting from your object, eg database table data, to JSON to send across the internet to another business

by example of "product" as the object

string output = JsonConvert.SerializeObject(product);

**We can also work with JSON in JavaScript - with different method names.**

From string to object

var myJSON = '{ "name":"John", "age":31, "city":"New York" }';  
var myObj = JSON.parse(myJSON);

From object to string

var myObj = { "name":"John", "age":31, "city":"New York" };  
var myJSON = JSON.stringify(myObj);

**Summary of JSON methods**

|  |  |  |
| --- | --- | --- |
| **Action** | **C#** | **JavaScript** |
| string to object | JsonConvert.DeserializeObject<T>() | JSON.parse() |
| object to string | JsonConvert.SerializeObject(); | JSON.stringify(); |