Notes on ASP .NET Routing. <https://msdn.microsoft.com/en-us/library/cc668201.aspx>

* The advantage for ASP .NET routing seems to be that you do not need to create a query string in order to pass information.
  + Instead the information is passed via variables within the URL. {controller} {action} {category}. This helps to keep the URL clean
  + It also allows use to specify what is suppose to happen at this url.

General Notes:

* **M**odels: a collection of classes which represent the data of your application.
* **V**iews: Template files that your application uses to dynamically create HTML responses.
* **C**ontrollers: A collection of classes that handle the incoming browser requests, retrieve model data and then specify the view templates that return a response to the browser.

Notes on passing data from the controller to the View

* Best practice: The view should never perform business logic or interact with db. It should only work with the data that is provided to it by the controller.
  + This maintains code separation, allowing for a clean and manageable code base.
* Controller classes are used to process incoming URL requests.
  + It uses this information to decide what type of response to send back to the browser.
    - This could be in the form of a view change, data processing, or something else.
  + Controllers use the **ViewBag** function as a way of **passing processed data from the URL request** to the **View template** where the View can then decide how to respond to that information.
  + The **ViewBag** will automatically be **bound** and **mapped** to the named parameters passed to it by the **ASP.NET MVC model binding system.**
    - The code ViewBag.Message or ViewBag.NumTimes = numTimes; is dynamically created. We can really name Message or NumTimes whatever we want because the ViewBag is a dynamic object.

Notes on Models

* We will be using the .NET Frameworks data-access technology known as the Entity Framework to define and work with model classes.
  + This is also known as EF.
  + Supports the **Code First dev paradigm!**
    - The Code First paradigm allows you to code simple classes up that will be the model objects.
      * These classes will then be created on the fly to make up your database.
      * These classes are known as POCO, or plain-old CLR objects.
      * These classes are made up in C#.
      * To use these classes we will need to specify *using System.Data.Entity;*
  + The classes created will be the information in the db.
    - Each instance of the class will map to a row in the db
    - Each property within the class will map to a column in the db.
  + We will be using the DbContext class which:
    - Represents the Entity Framework
    - Allows us to fetch, store, and update the Movie class instances in the db.
    - In order to use this, under the class in our Model we will do “YourDbClass”DBContext : DbContext { public DbSet<”YourClass”> varName {get; set; } }
    - *The name highlighted must match the name in the connectionStrings section of the root Web.config file.*

More on Databases in VS

* We will be using the LocalDB natively installed with VS.
  + LocalDB is a lightweight version of SQL Server Express Database Engine.
  + This DB allows you to work with .mdf files
    - These files are typically kept in the App\_Data folder of our web project.

Model Data from the Controller.

* After creating a new controller from the Controller w/ view from Entity Framework option in the add section of the controllers folder.
  + Specify the model class and the data context we will be working off of. (should be in a drop down menu.)
  + It even created a good Controller name for you! Wow.
* This now adds a few files.
  + New Controller file, new View File and a CRUD set up within the view file. (Create Read Update Delete)
* If take a look at the Details.cshtml page we can see that the “**@model** syntax is used to specify what type of object this class is expecting.
  + We also see this in the Index.cshtml file with **@IEnumerable**. Compared with the MoviesController.cs file which when **Index()** is called the **db.Movies.toList()** method is called and passes in the list to the Index.cshtml file.