**https://github.com/mosh-hamedani/vidly-mvc-5/commits/master**

**FOLDERS USE**:

* APP\_DATE: Used to store database details.
* APP\_START:Has building of class files that are called when the app is started.
* CONTENT:Store the css files,images and aother client side assets.
* CONTROLLERS:By default it has 3:

Text

Description automatically generated

i)Account contains: has actions for singin,login,logout.

ii)Home: Contains home page.

iii)Manage:Handling request along user profile (enabling password,2 factor authentication,social logins).

* MODELS: All the domain classes be here.
* SCRIPTS: We have java script files
* VIEWS:View name should be same as controllers,shared folder can contain views that can b used across the difrnt controllers.
* GLOBAL.asxs:Tradional file,class that provides hooks for various events in the app lifecycle. When the app is started the function inside global.asx file will b called.
* Web.config:is an xml includes configuration

1. **“ACTION RESULT “ IN Controller:**

**Can use action result or view result :**

DIFFERENCE B/W BOTH:

**ViewResult** is a subclass of **ActionResult**. The View method returns a **ViewResult**. ... The only **difference** is that with the **ActionResult** one, your controller isn't promising to return a view - you could change the method body to conditionally return a RedirectResult or something else without changing the method definition.

**TYPES OF ACTION RESULT:**

|  |  |
| --- | --- |
| ViewResult | View() |
| PartialViewResult | PartialView() |
| ContentResult | Content() |
| RedirectResult | Redirect() |
| RedirectToRouteResult | RedirecttoAction() |
| Json Result | Json() |
| FileResult | File() |
| HttpNotFoundResult | HttpnotFound() |

1. ACTION PARAMETERS:

Parameter can be in the form of

Query string eg:

/moviese/edit?id=1

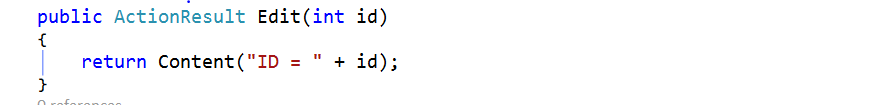
In the form of url eg:

/movie/edit/1

Parameter eg:

Id=1

EG:1



Graphical user interface, text, application, email

Description automatically generated

EG:2

Graphical user interface, text, application

Description automatically generated

In here while passing variable (“?”) symbol is to denote that the variable can be made to “nullable”

Graphical user interface, text, application, email

Description automatically generated

1. CONVENTION -BASED ROUTING:

While we are creating custom routing,,the path we should mention from “**most specific to most generic**” orelse more generic will apply.

Text

Description automatically generated

The highlighted part is for validation the year should be in 4 digits and month in 2 digits

Graphical user interface, text, application, email

Description automatically generated

If not giving the values ciorrectly will display the above messages

1. **ATTRIBUTE ROUTING:**

It is more powerful than conventional routing,it is mostly used now.

Need to declare in routingfile then directly used in contrioller.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Also has (min,max,minlength,maxlength,int,float,guid)

1. PASSING DATA TO VIEWS:

\*ViewBag

\*ViewData

Both are old ones and hard due to casting ,it is even difficult to use so use view(“obj”).In view file use model.variable.

1. VIEW MODEL:

It is specifically build to display a view,if we need to dsipalya the movies as well as the customers who selects the movie

Steps:

We need to create a customer class with id and name

Need to create a viewmodel class with

Graphical user interface, text

Description automatically generated

In controller

Graphical user interface, text, application

Description automatically generated

In Randomlayout:

Graphical user interface, text, application, email

Description automatically generated

1. RAZOR VIEW:

Can use c# code inside Random.cs.html page

Graphical user interface, text, application, email

Description automatically generated

1. PARTIAL VIEW:

Can place the piece of code by creating the partial view and refer the file name in the main view

1. DATABASE FIRST vs CODE FIRST

Code first is better tan database first

**Database**

**Entity Frame Work**

**Code**

1. Code-first:

In Package manager console;

\*enable-migrations---so u will get a folder called migration

Graphical user interface, text, application, email

Description automatically generated

* add-migration ‘any name’-----for creating
* ad-migration samename -force ----for updating or overwriting

Table

Description automatically generated with low confidence

Identity Db context is the gate way to DB

* In application db contect function add “required table” to be created in the database.

Graphical user interface, text, application

Description automatically generated

* Then overwrite
* Last update db using ” update-database” –wil get below dbs

A picture containing text, screenshot, computer, computer

Description automatically generated

1. Seeding Db:

We should not go to db and manually insert value,everything should b done in the form of migration.In case we need to add the particular value through code first add a migration ,then will find a up method,there u should insert the value

Graphical user interface, text, application, email

Description automatically generated

Then simple update database.

1. Convention change:

By default the variable we declared as string will taken as “varchar” in db and it will be nullable,inthis case we can change that

Graphical user interface, text, application, email

Description automatically generated

Then migrate and update db

Graphical user interface, application, table, Excel

Description automatically generated

1. **BINDING FORMS:**
   1. **MARKUP:**

* **@HTML.BeginForm**-like <form> when we post this page the function inside the begin form will be called eg: **@HTML.BeginForm(“create”,Customer)**

**Here create is an action and customer is the controller.**

* If we use **@using(HTML.Beginform){}** then after the braces the dispose method is called which act as </form>
* **@HTMLLabelFor()-**inside we should use a variable we declared in model.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* For checkbox us the below

Graphical user interface, text, application

Description automatically generated

* For DropDown -usually it returns the list

<div class="form-group">

        @Html.LabelFor(m => m.Customers.MembershipTypeId)

        @Html.DropDownListFor(m => m.Customers.MembershipTypeId,new SelectList(Model.MembershipType,"Id","Name"),"Select MemebershipType" , new { @class = "form-control" })

    </div>

**Definition of Dropdownlistfor(“modelclassfromcustomertable”,new selectList(“List”),”Id”,”Value”),”Any string to display first in dropdown”,”styleclass”)**

**REFERENCE:**

Graphical user interface, text, application

Description automatically generated

* RadioButton

**REFERENCE:**

Graphical user interface, text, application

Description automatically generated

**WEB API:**

**Web API** is a programming interface/application type that provides communication or interaction between software applications. **Web API** is often used to provide an interface for **web** sites and client applications to have data access. **Web** APIs can be used to access data from a database and save data back to the database.

JQUERY-LEARN

https://www.impressivewebs.com/jquery-tutorial-for-beginners/