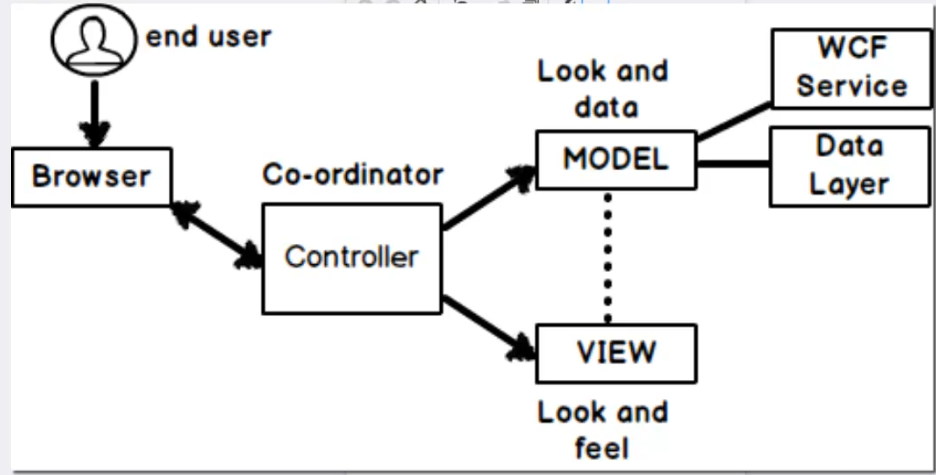
File > new > project> Asp.net application MVC

Architectural patter divided into 3 part

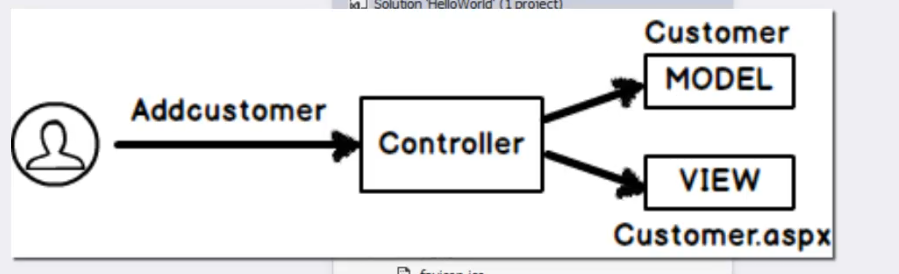
Model: supplied data and business login

View: deals with presentation basically rendering of the data.

Controller: Heart of mvc act as coordinator between model and view.



When user request for resources ultimately request comes to the Controller which basically fetched the data from the model and send t to the view



Each layer is associated with its own functionality so changes done to one layer will not effect the other layer they are completely independent of each other

Don’t delete controller word while making new controller.

For every controller created a folder created inside view folder to make all views related to that controller.

Controller will do some manipulation and return the view to be rendered on the screen.

Inside the view folder we have shared folder

Folder name Specific to the controller

So the view can be present in any of these folder

If the name of action method and view is different then we have use name of the view along with the view returned from the action method.

If the name of vie and controller action is same then just use return statement to return view.

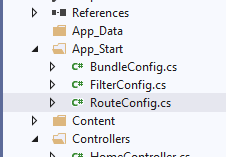
Shared folder in views folder basically contains list of all the view.

**Routing in mVC:**

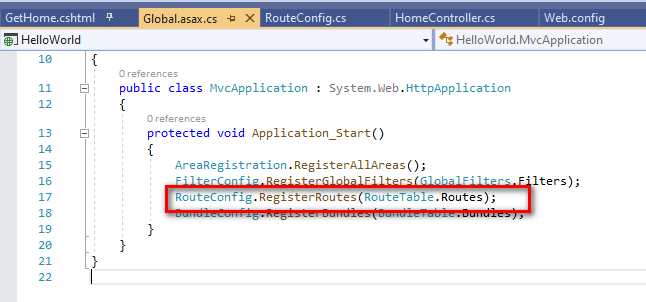
Simplifies the mv curls

Maps view url to our controller and action methods defined

Basically we can define user friendly url for the view







Techniques to pass data to a view:

No code behind

No view states

No server control.

1. Viewdata/viewbag
2. Tempdata
3. Session variable.

View data/view bag: to pass data from the controller to the view or action to view

Scrip tic syntax: ViewData[“key word”]= value;

View bag is syntactic sugar on top of view data

ViewBag.MyTime = DateTime.now;

Viewdata or View bag maintains only data between the action to view

It will not maintain between the controller to controller or action to action.

So to maintain data between controller to controller

Or action to action we use Temdata

Just maintain data between controller to view used viewdata

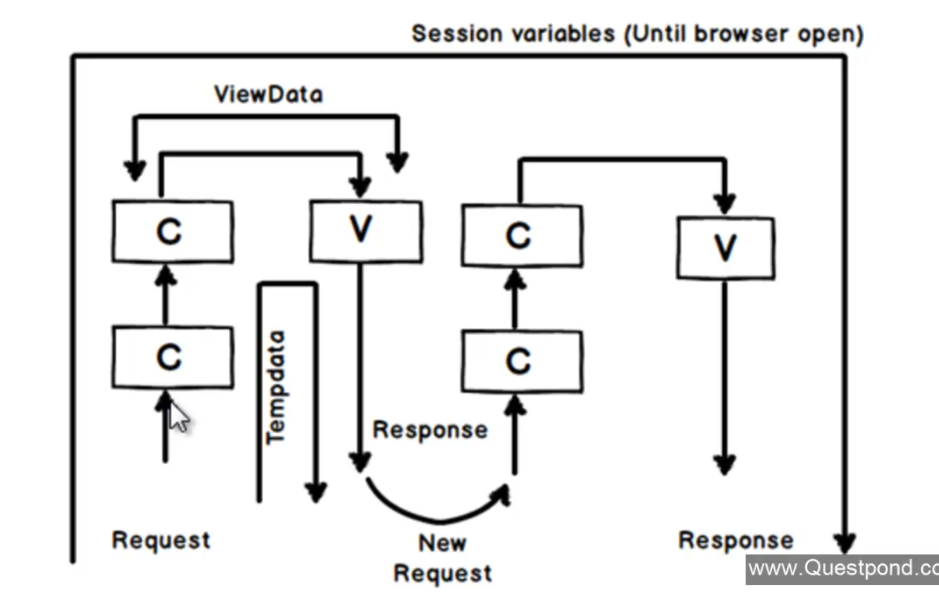
To maintain data to entire request between controller to controller and then from action to view uses temp data.

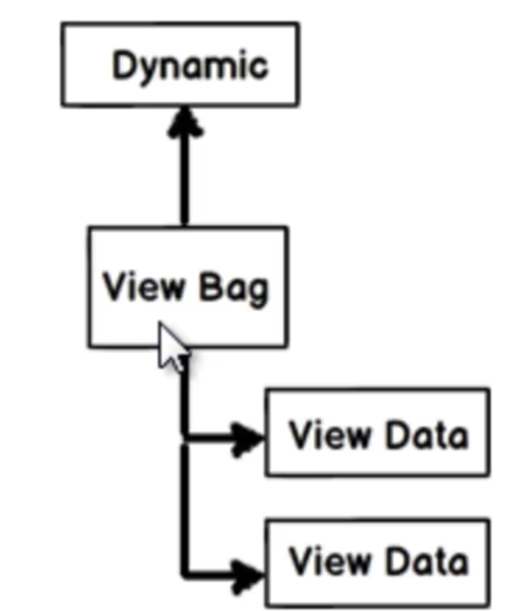
View bag is collection of view data.

View bag scope: between controller and view

Temp data scope : between a full request cycle (c ->c->c----🡪 v)

Session variable scope: value will be maintained until the session is active.





Redirect call from one action to another action:

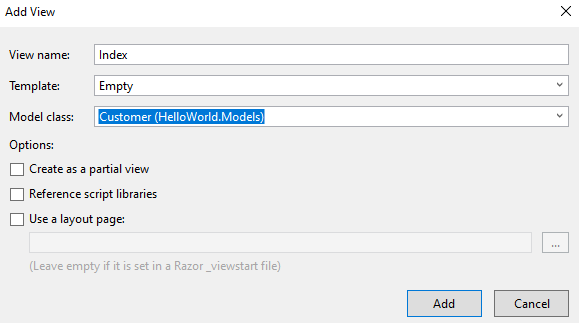
return RedirectToAction(“Action name”,”Controller name”);

Model & strongly typed views:

Model is responsible for the data and business logic associated with the data.

View connected to model with data binding is called the strongly typed view

So while making model



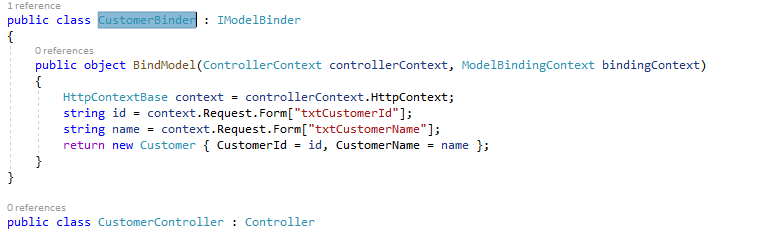
Model binders:

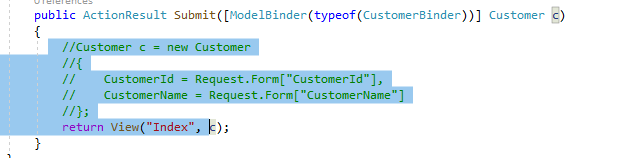
Automatic binding for this model property and form fields name should match with each other.

What if the model field name and the form control names are different?

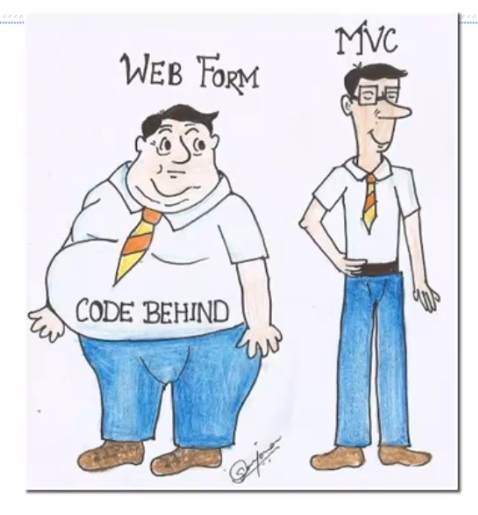
In such scenarios we need to used model binders as attribute on top of the model object bound to the controller action method

Interface iModeBinder > method BindModel





Why MVC:



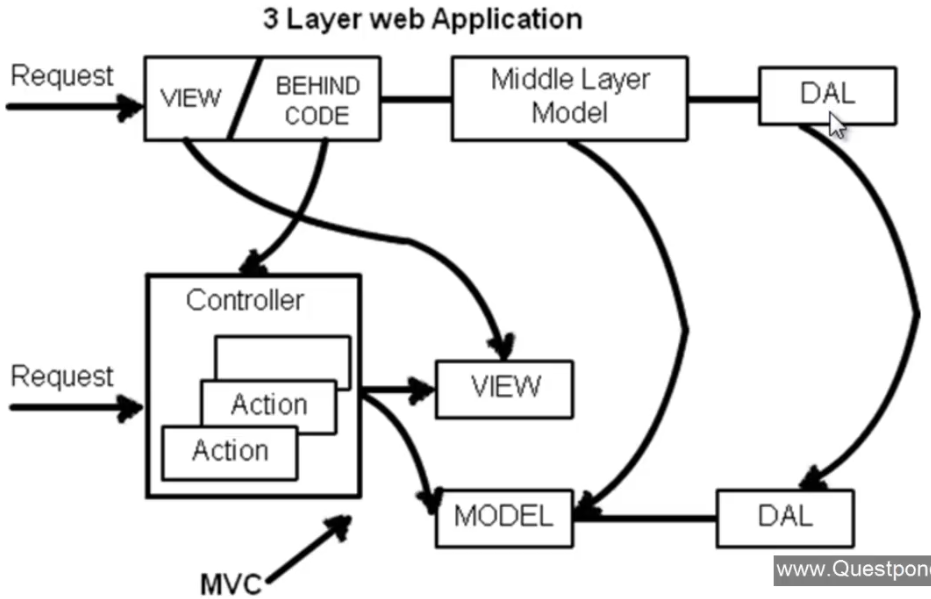
1. View based architecture, web action based architecture

2. Behind code is reusable for different view types (mobile, web etc)

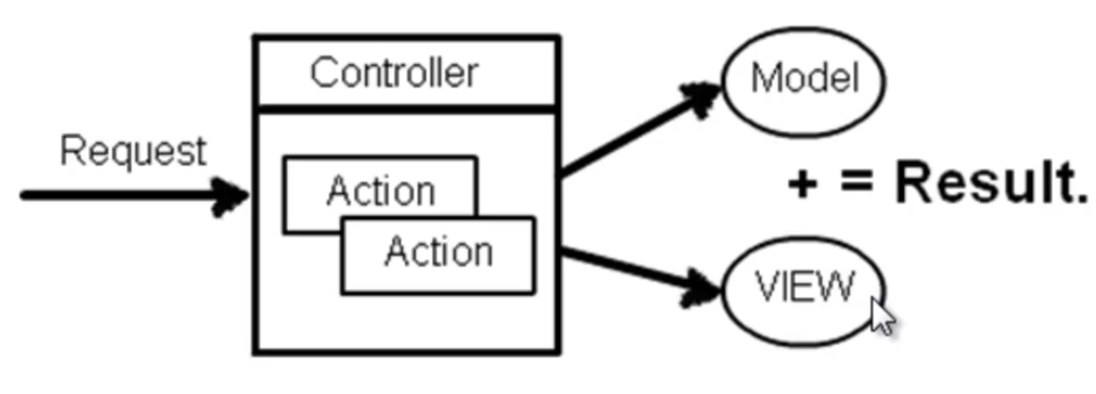
3. Fixed response type in web

3. flexible view and data Control which type of data to be returned or which view can be used for the rendering

4. testing is easier because the things are loosely coupled.

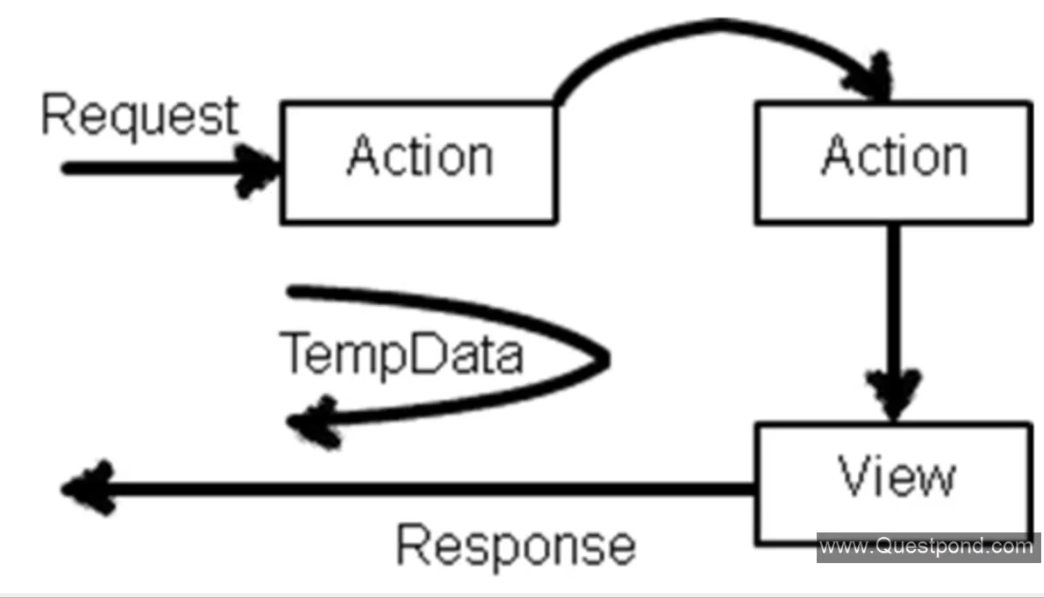


MVC

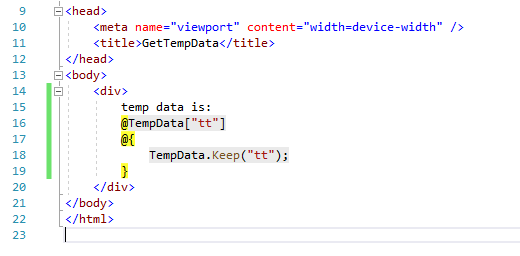


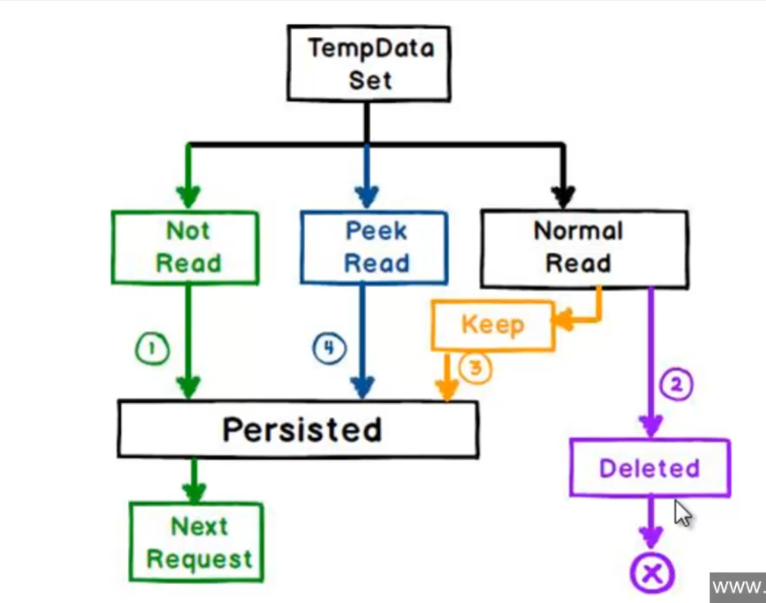
Temp data keep and peek:

Temp data can be preserved between multiple request

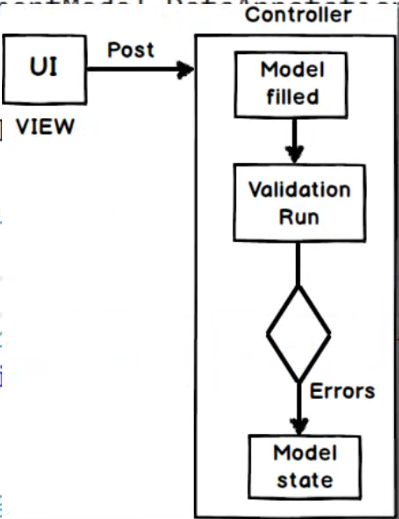


1. If the value is not read from the temp data value will persist
2. If the temp data value is read the value will be cleared or lost
3. If you run keep method on the temp data after reading its value the data will persist.
4. Read value using peek method it will persist the value.



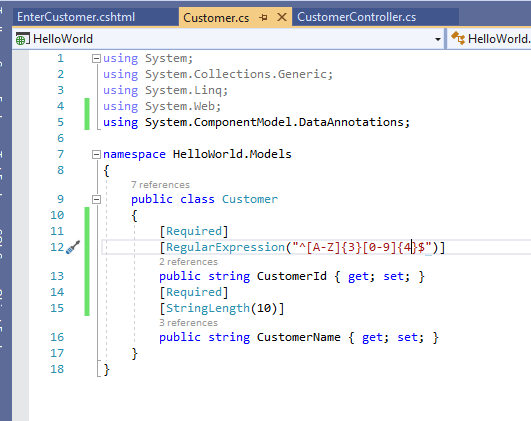


Data annotation and helper class:



With data annotation its possible to setup the attributes and constraints on top of the class fields.

Help class are nothing but the class help in achieving some on the view components

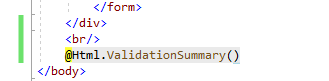


Decorated with constraints

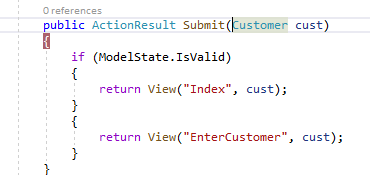
App setting:



Helper class to display validation summery

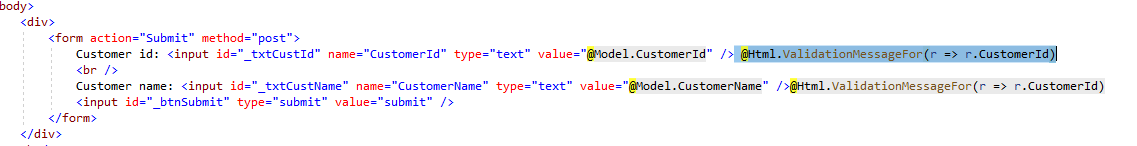


On model binding of the controller action method validation will be done and ModleState class will be set.



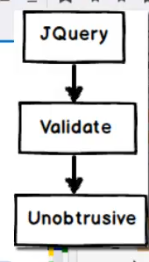
Instead of displaying the error message in on section how about displaying after each and every control.

@Html.ValidationMessageFor(r => r.CustomerId)



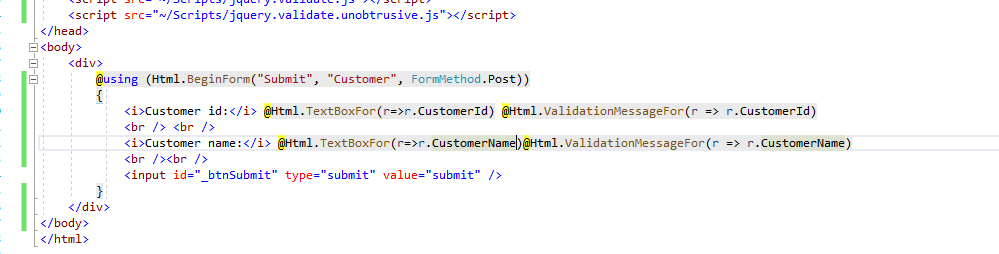
This type of validation will make a round trip to the server in order to perform the validation

Enable jquery to do a validation on the client side.



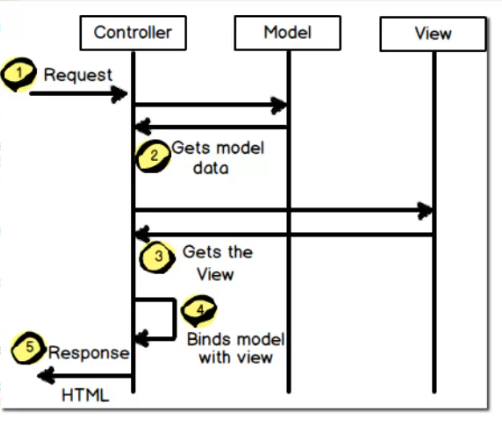
Steps:

1. Include jquery files in he above mentioned order 3 files.
2. Replace form tag with html helper tag for form
3. Replace input controls with equivalent helpers.



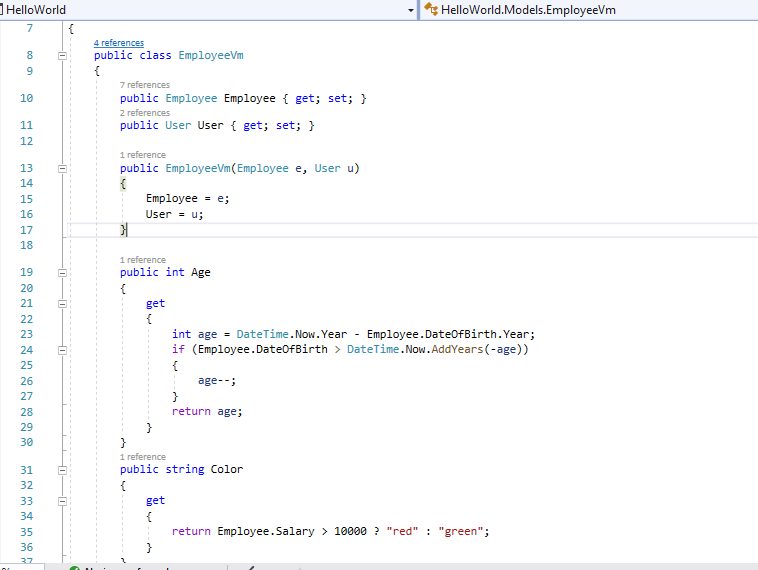
View Model:

Sequence diagram for mvc:

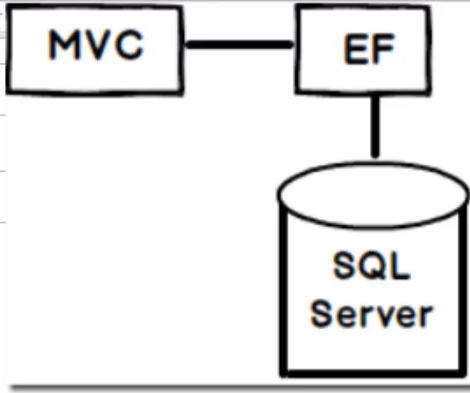


1. Make strongly typed view of multiple models
2. Data transformation logic
3. Presentation logic

Define a custom class which has the logic of presentation and transformation and the combination of multiple entitiy.



Entity frame work in mvc



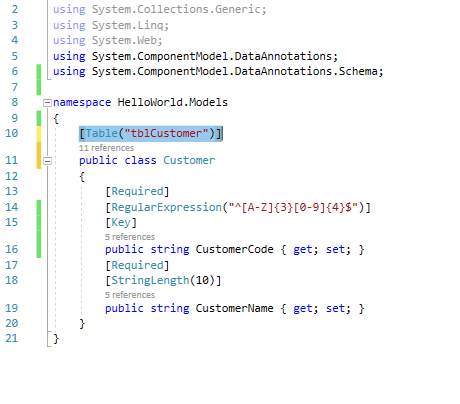
For code first approach:

1.Define db and the entity file

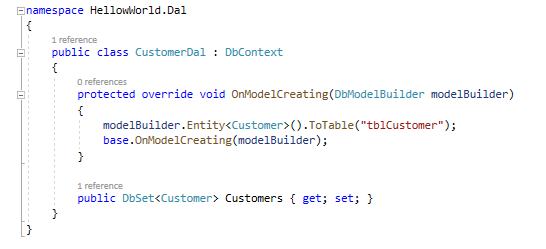
2. Add connection string in the webconfig file

3. mapping between entity and the table done

1. With the help of table attribute



2. Override event in Db context file.



View model & Partial view & Grids

View model is basically used to club multiple entity in a single class file so that binding of the model to the view becomes easier

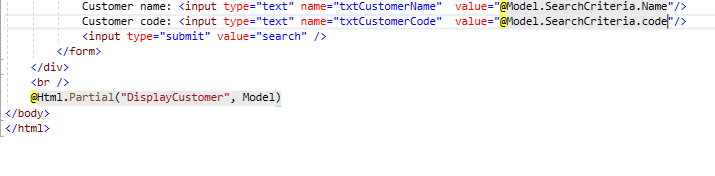
Eg: Customer application when in I have to add new customer display the list and search the customers.

Here we need to add a model view class with list, criteria and single customer object

Partial views:

Reusable view used in the system similar to web used controls.

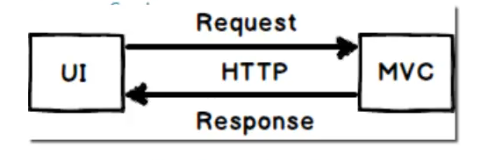
We use html helper class called HTML.Partial(“view name”, model);



Web grid built in control to build grid



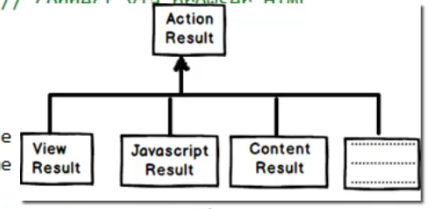
Action results and view results:

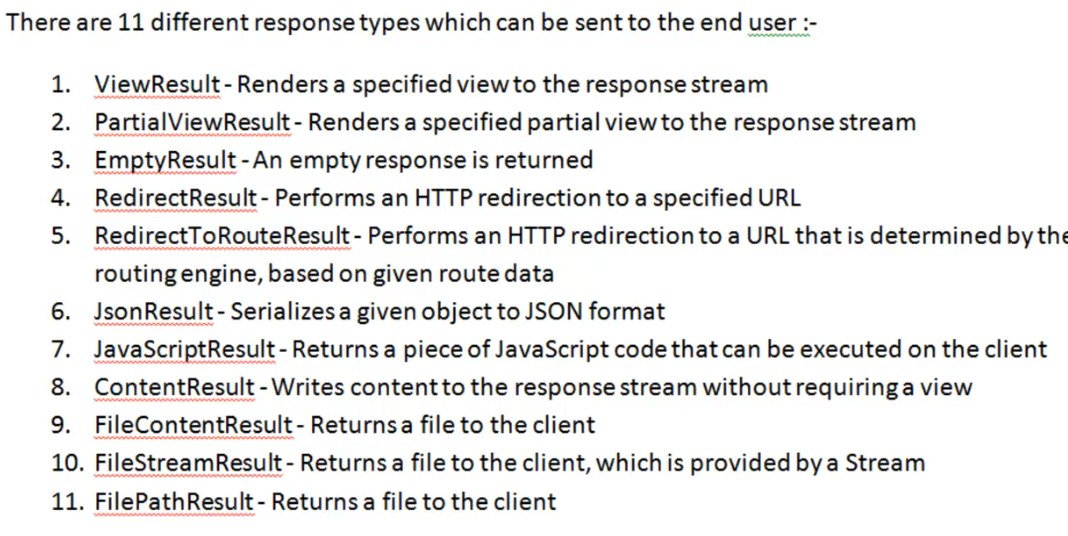


Both are class representing the result nothing but response comes from the controller actions



ActionResult is apparent base class for all the result returned from the action method So we can return any type of the data out of it





Ajax using Json jquery

Asynchronous javascript and XML

It will does asynchronous processing and sends and gets only the necessary data

Ajax call for getting the data

$.get(‘Action name’, null, success method)

Function Success(data)

{

}

Ajax post call:

$.post(‘Action’, $(‘form’).serailize(), success method)

Asynchronous controller in mVC

IIS has thread pool which basically servers the request comes from the user

With this we make a controller aynch so that there wont be any thread starvation problem.

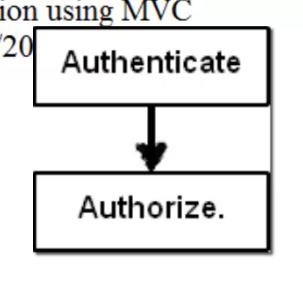
4 steps to do it

1. Your controller must implement AsynController base class
2. Controller action must be mark as Async
3. Controller must return task action result Task<Actionresult>
4. Add await to the long running process.

Authentication and authorization.

Authentication: if the user is exist in data base or not

Authorization : check if the user has access to use then section of the page.

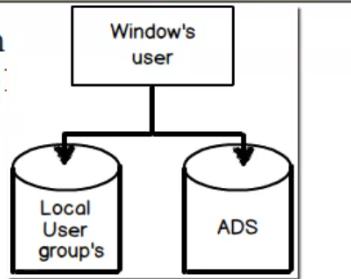


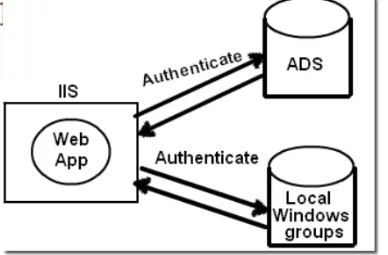
Authentication:



For windows authentication users need to be saved in the local windows machine or ADS active directory services

Normally used for the internet based application. Where all the credentials saved in the server and access using windows services.

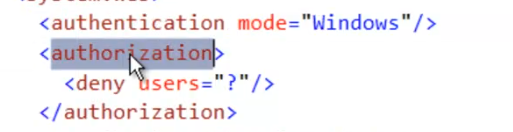




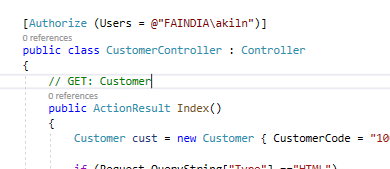
Windows authentication:

Steps:

1. Enable windows authentication from Web config
2. Authorization to deny anonymous user.



Add authorize attribute on top of the controller class

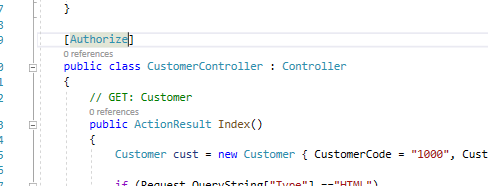


Forms Authentication:

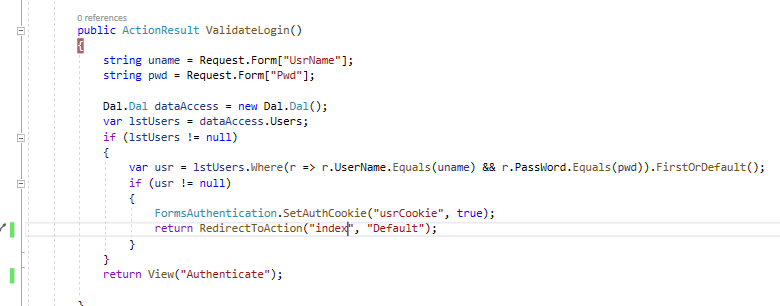
1. Add user table
2. Web config setting.



1. Decorate controller with

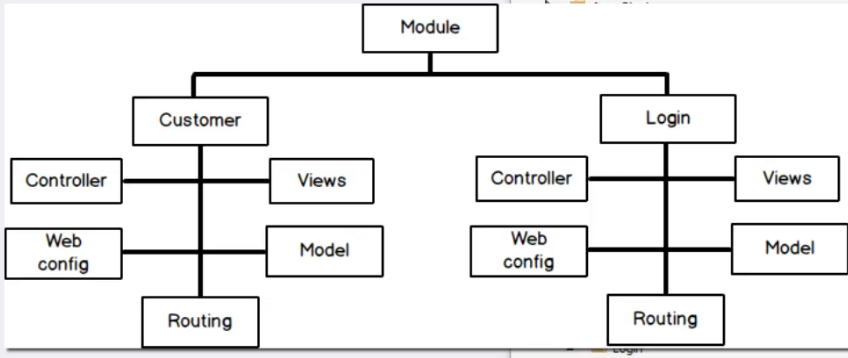


1. Provide authentication details.
2. Validate
3. Set authentication cookie



Areas in mVc

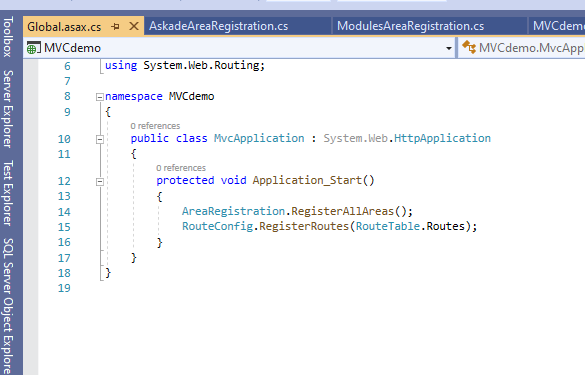
Modular development



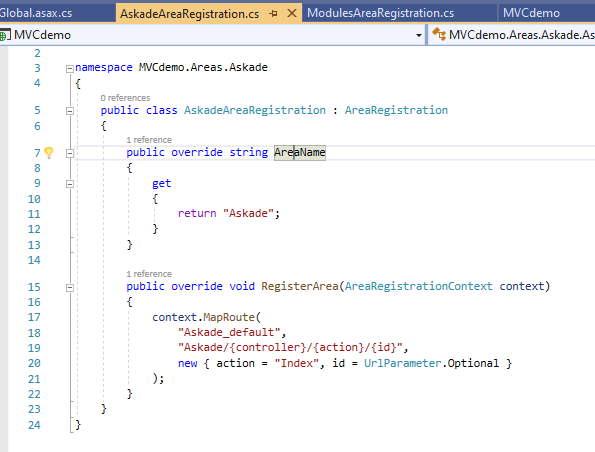
For creating new area

Right click on the project and create area

Area has its own views controller, model, configuration file and route registration



Area registration:



Angular and MVC 5

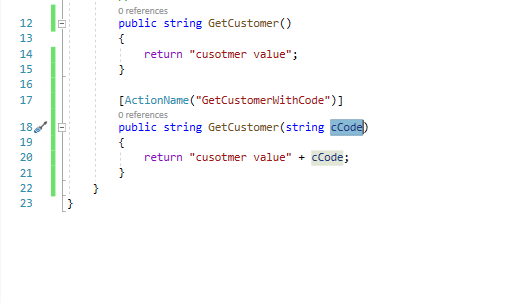
Can we Overload the controller action methods:

Yes and partially no

Because C# supports overloading of methods

But in MVC controller action method forma the end point for the url and HTTP expect every url to be unique

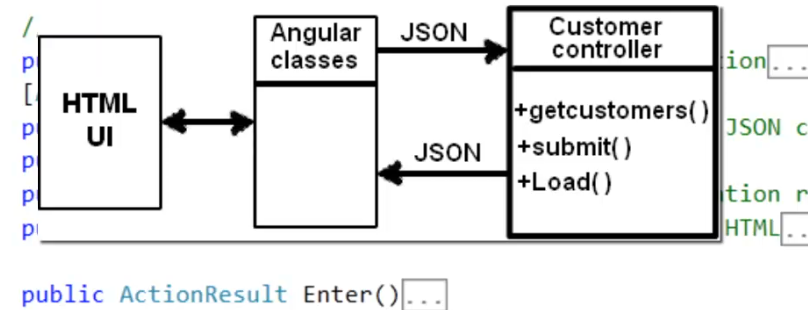
So this can be achieved with the help of the ActionName attribute on top of the controls action method.

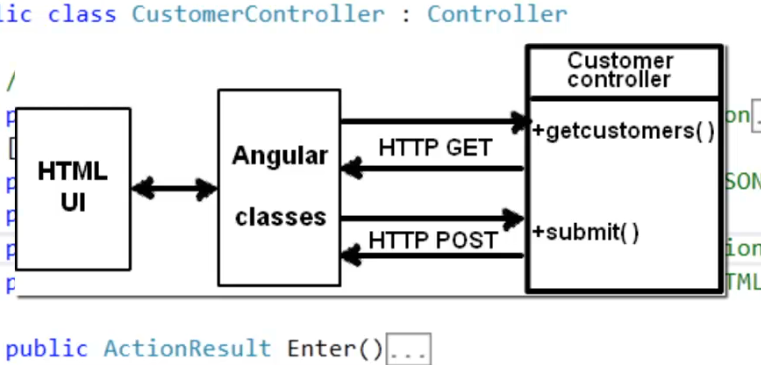


Web Api

Respects HTTp protocol and uses the rest architecture.

Current architecture.



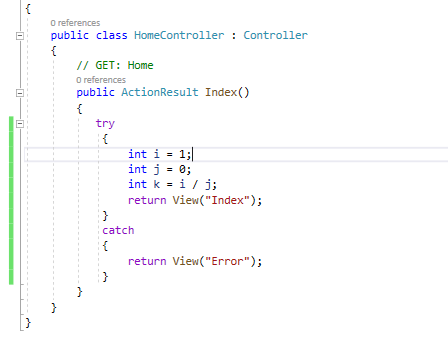


Rest:

Both client and server respects the HTTP way of communication so it simplifies the implementation between them this stand for representation state transfer

Exception handling in MVC

1. Use try catch block

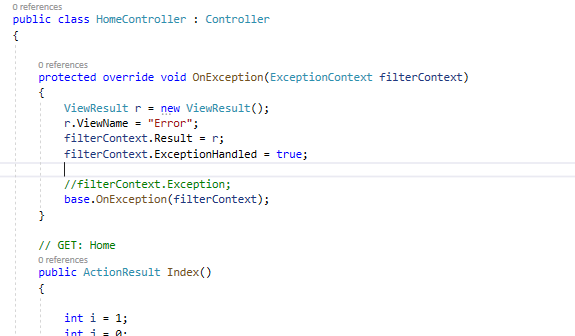


Problem violation on dry principle (do not repeat yourself) . This implementation will repeated the exception handling logic in all the controller action methods.

1. OnException method from the controller class

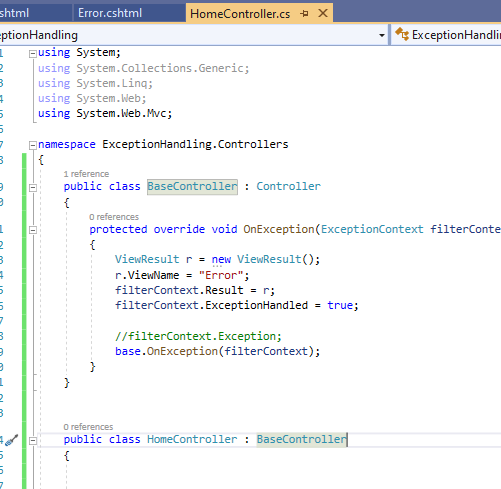
Here you need to create view result and assign it to exception action result

And set filter exception handled property to true to indicate exception is handled indicating that no default behaviour in needed



Problem specific to the controller exception handling login cannot be shared with other controllers.

1. Inheritance make base class implementing controller and implement the OnException method in base class and the implement all controller from the base controller.



Problem customization where in different controller want to customer the exception handling logic.

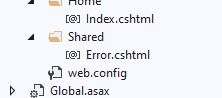
1. Exception filter for the controller or action

Steps:

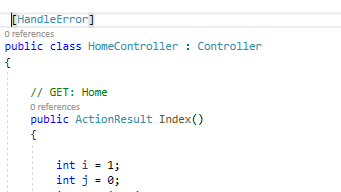
1. Enable custom error from webconfig



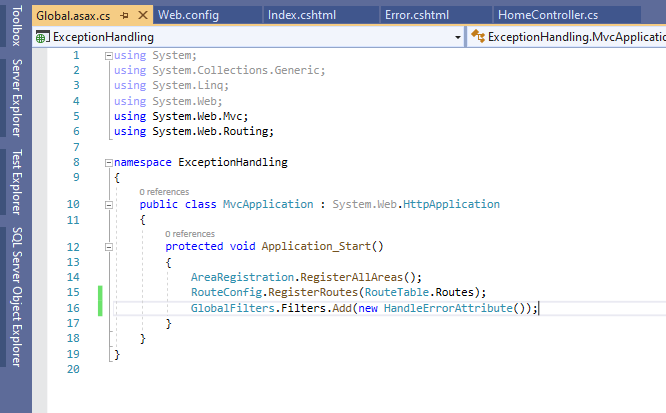
1. Create Error.cshtml view in shared or controller view folder



1. Decorate controller or action method with HandelError Attribute

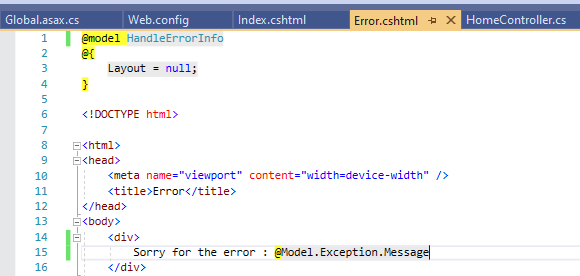


But for this we need to add this attribute for each and every controller we can avoid this by registering the error handling at the global level



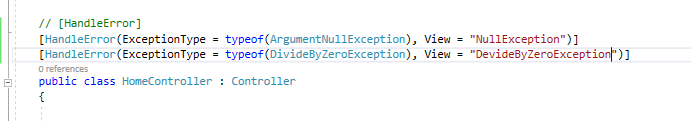
Say you need to display the information about he exception

Then make your view as strongly typed view for the class type ‘HandleErrorInfo’



Say you want t0 render a different view for different error





In above approach if you want to perform the logging on exception the it is not possible so to support this we need to have some custom exception filter class.

WebGrid: helper function present inside the MVC library

Make view model

The model specific to the view

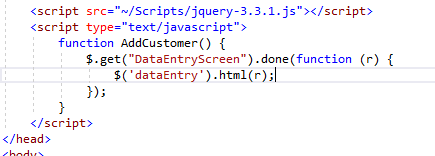


Single page application WITH MVC

Building application using one page

Here im using a promising object $.get() to get the action result\

Promising objects basically tells on completing of the execution it will return something for the use.

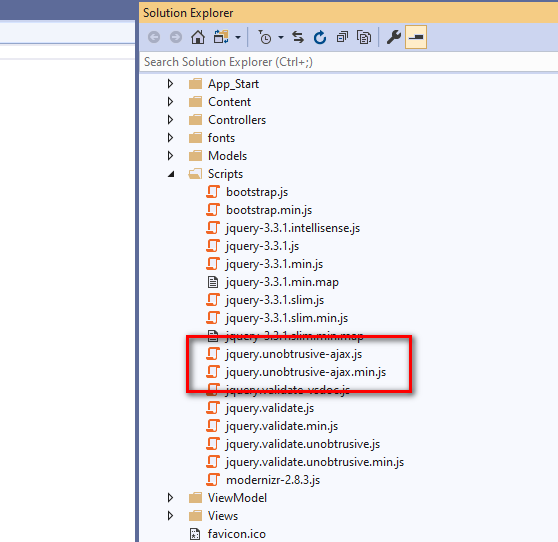


Mvc Ajax for the ajax call

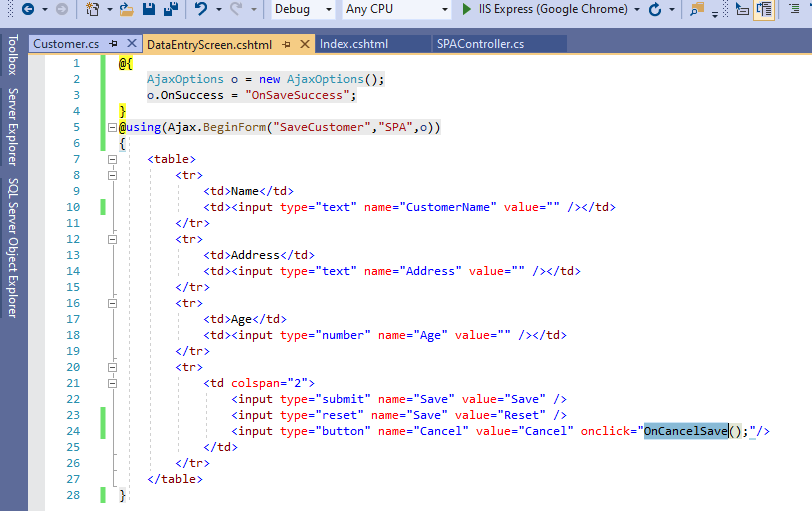
Install this package to the solution



Includes this jQuery file



Then use the Ajax for tag to perform MVC ajax call



Organize MVC project and understanding cross domain issue.

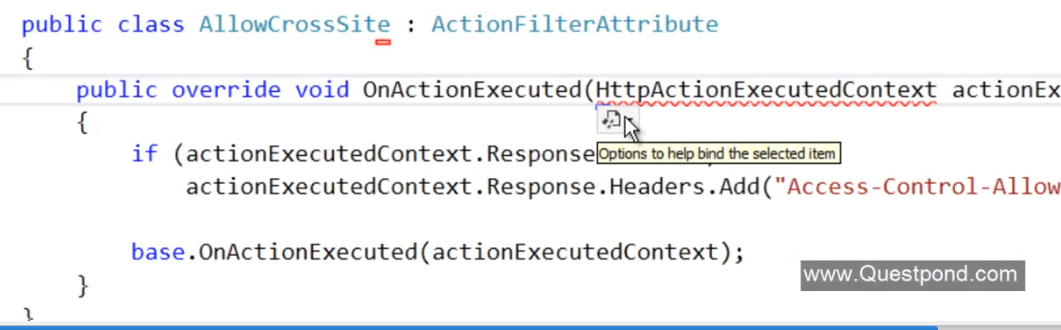
Add separate projects for the

Model

Data access layer

Ui

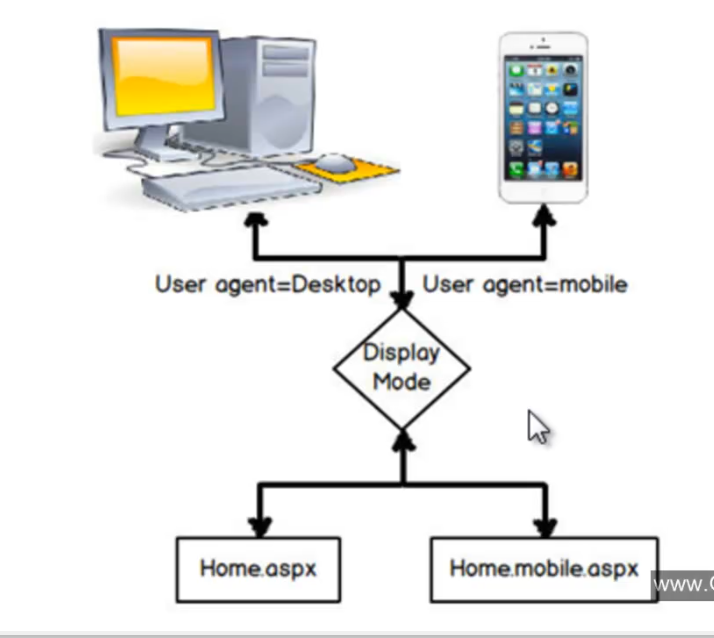
Web Api



When web api and the ui hosted on different domain we get a cross domain issue. So to fix this we need to add the Action filter attribute on top of the controller to allow access to the Api.

Access-control-Allow-Origin header value to be sent with reponse

Display modes:



Based on the device its possible to have the separate view for the device

For that we need to create separate views

Index.cshtml

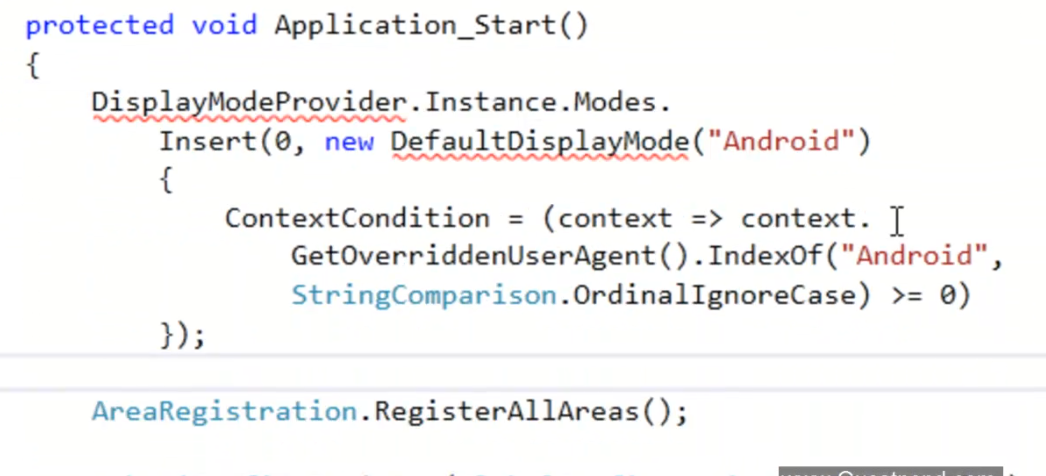
Index.Mobile.cshtml

We can also have the custom views based on the os. Say for android I want to have a separate views.

Then

For this first we need to add display mode providers which is defined under

System.web.webpage;

Then add

Index.Android.Cshtml add this view

Unit testing in MVC

Testing the code (every unit of the code we test) in automated and isolated mode is repetitive manner

In MVC we can test models

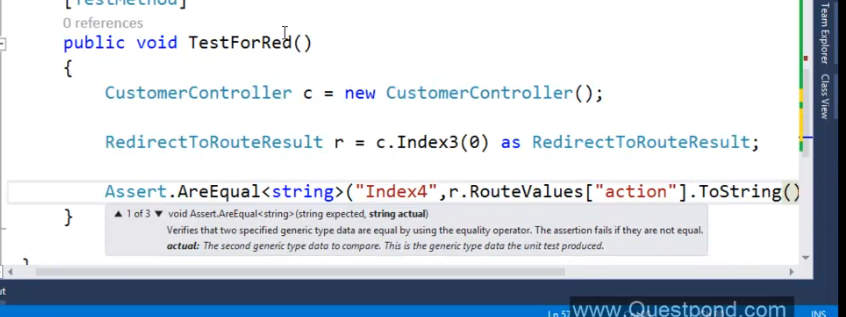
Data access layer

Left out Is controller. That mainly perform this job received the request from the view. Generate data and send it back to the view.

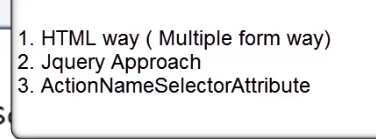
1. Test if the proper view or not
2. Proper model/viewbag/viedata
3. Proper redirect to view

Testing the redirection result:



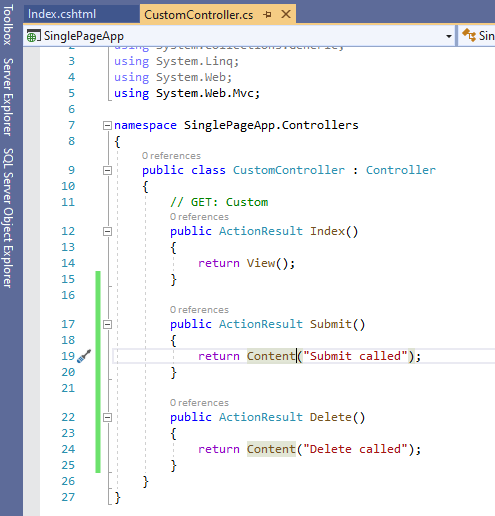


Multiple submit in form:



1. **HTML way of solving problem add separate form tab for each buttons.**



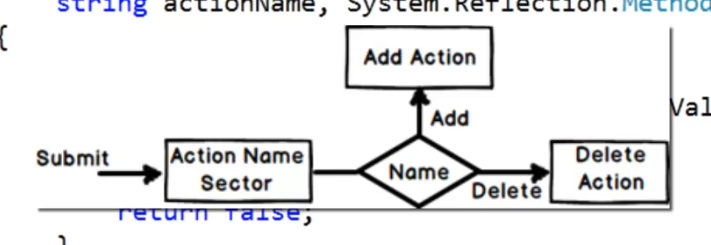


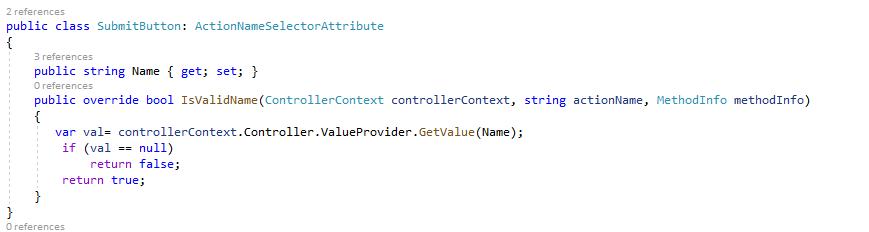
**2.JQuery solutions:**

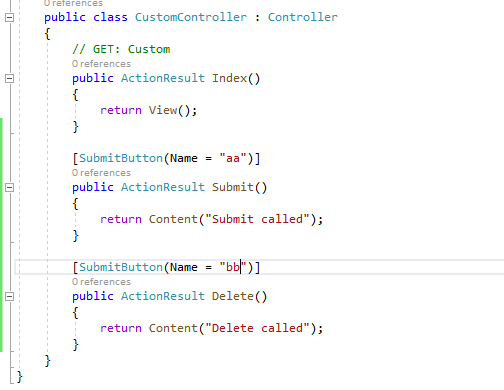


1. Use Action Name Selector Attribute









Antiforgery token:

Basically used to avoid the deadly attack called CSRF (Cross Site Request forgery). Method of hacking system where in he acts like trusted system.



CSRF: cross site request forgery:

Hacking method by which the requesting client forges to be the trusted client and get access to the system

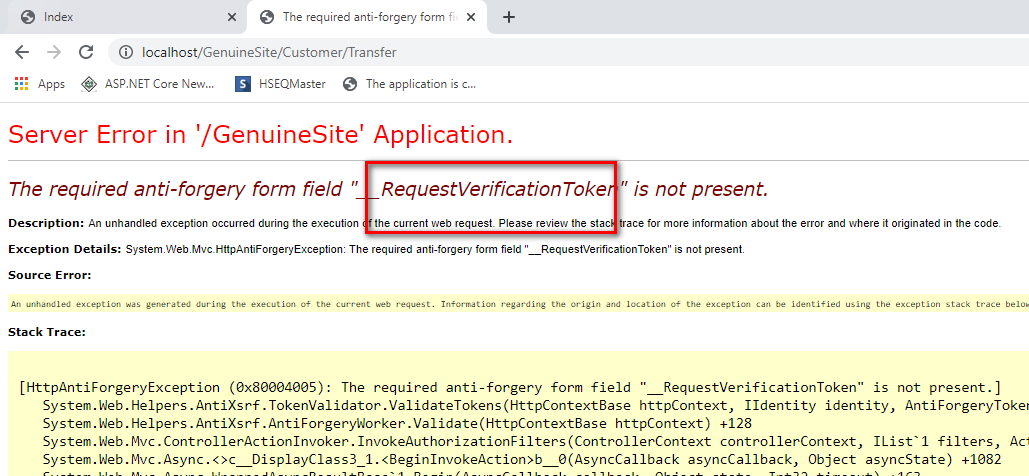
This can be eliminated with the request token sent as a part of the request generated by the server called antiforgery token

Html helper used:

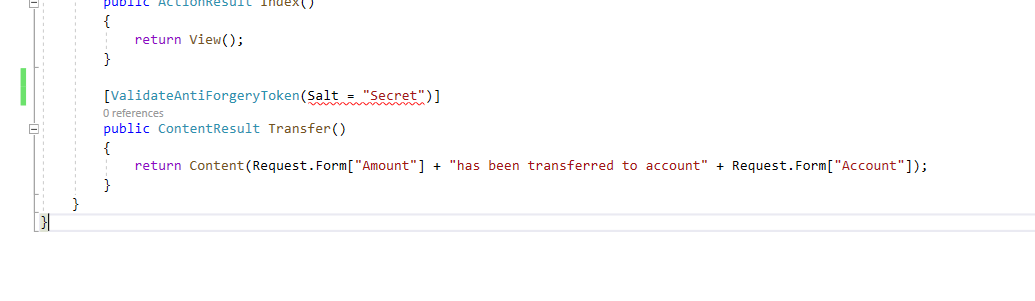




From the forged site you get



Forged client my also generate the antiforgery token so use salt value to eliminate.



Input validation:

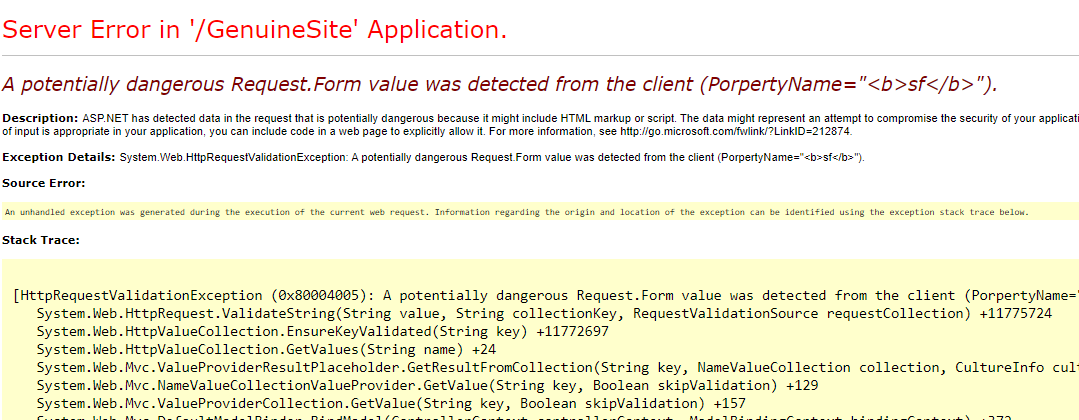
Validate input allow Html

XSS – cross site scripting issue. Script attacking to the application where In the hacker will inject malicious script to the application as a part of input.

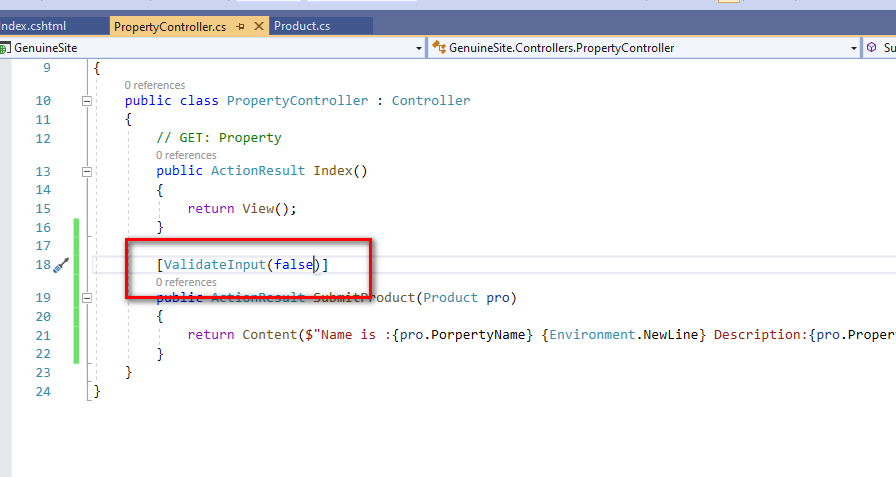
Validate input on the action method level

AllowHtml is on the property level.

By default input security is enabled. It will throw and error if in valid html is entered.



Disable input validation for entire action method. Which will open up security issue. And application is more prone for the XSS.

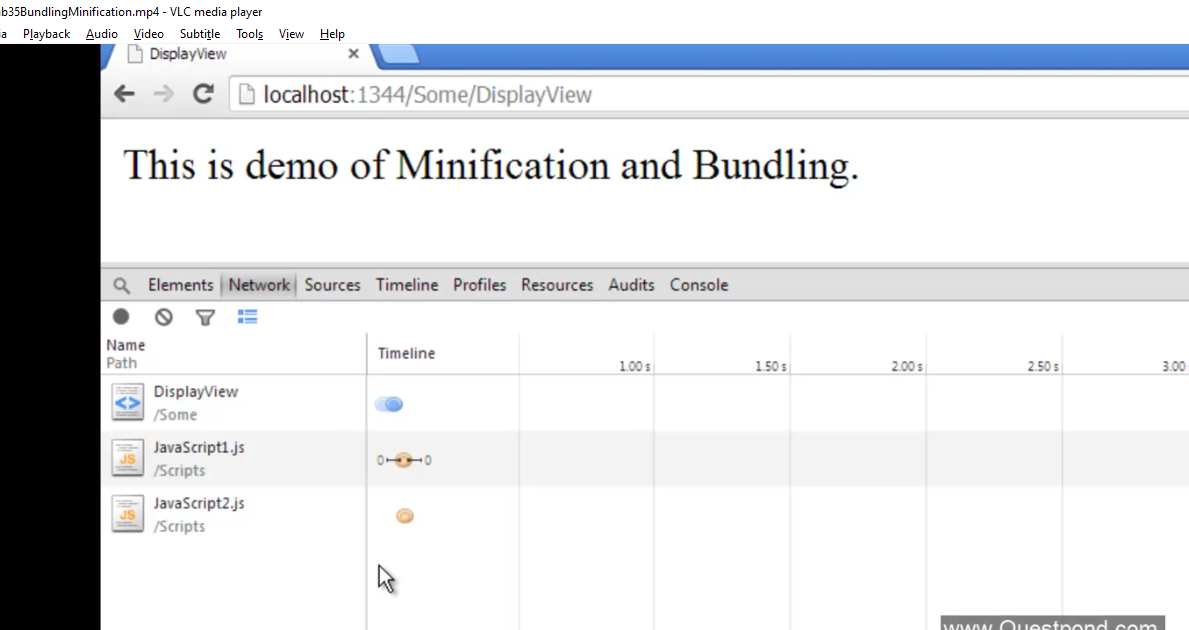


Suppose you want to allow html only for the particular field and not for entire form.



Bundling and minification:

Increase the performance of the application

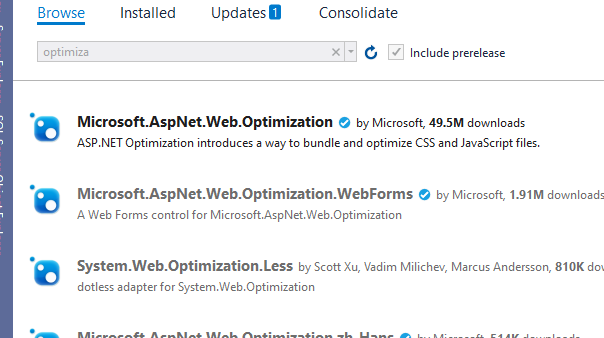


Say application has multiple css files and javascript file . we make a call to a page that uses this css and JavaSript. So depending on number of css files and jquery file our browser will make so many calls to the server or so many get requests to the server.

SO with bundling we make number of CSS and javascript files into single entity and during runtime we use single call to download this single entity.

So all multiple request will be eliminated and with single call we can get all the files.

System.web.optimization namespace to do this



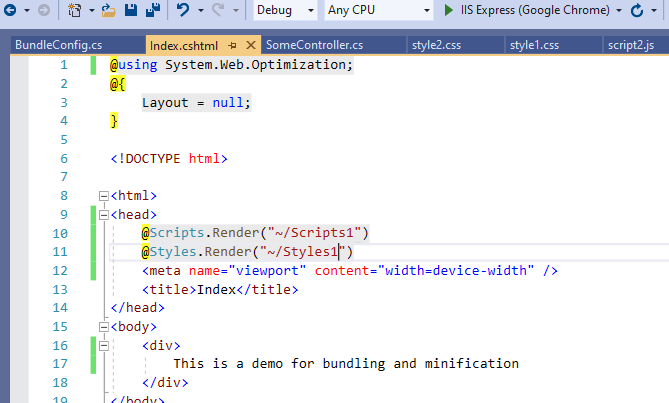
Above name space is responsible for the bundling in .net

Then we need to implement the BundleConfig.cs file in the project

Minification

Will reduce the size of the Css, script file by removing all the comment, white space, new line characters.





Layout pages in MVC:

With this we can achieve consistent look and feel across the application. Header, footer, logo etc

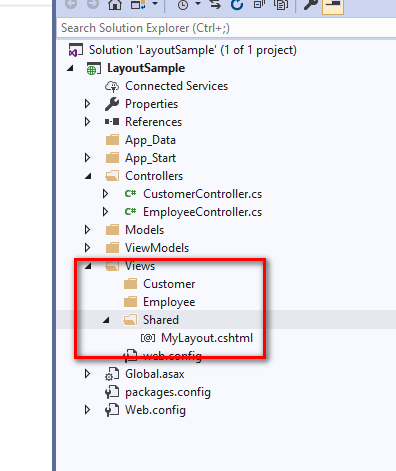
Its also type of view

Similar to what we have in master page

With this we can have the shared design or html for our application

Define Html layout at one and use it across all the views

Add shared folder inside views



Things put inside the shared folder can be shared by all the controllers

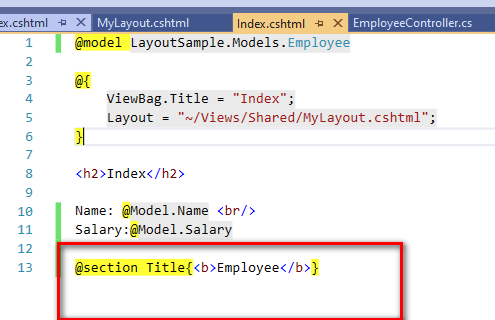
Then Add layout related html common for all page

Then call body render method @RenderBody() this will render the body content from the

Suppose we want to render any section of the content from view to master screen then we can have name section. And define that section in view

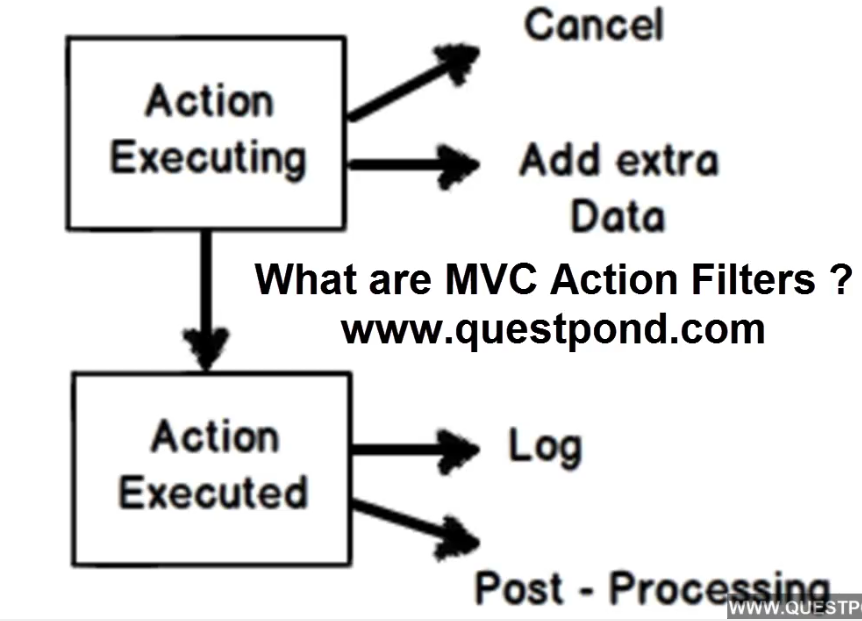
@RenderSection(“Section name”);

@section desction name{define content to be rendered}



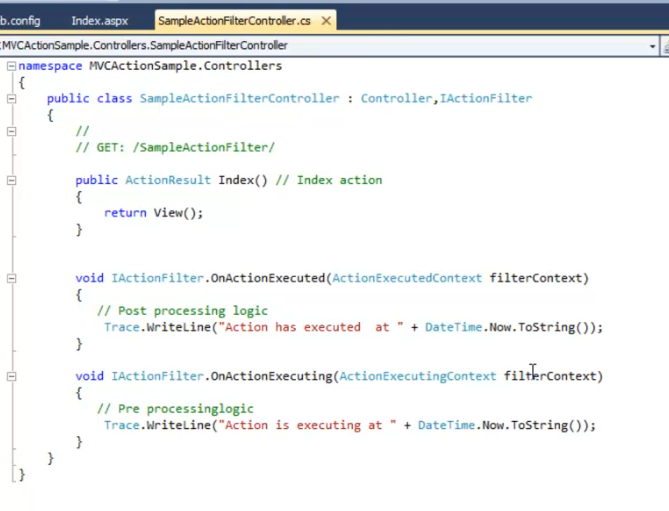


Action Filter:



With this we can inject a logic before and after the execution of the action methos logic

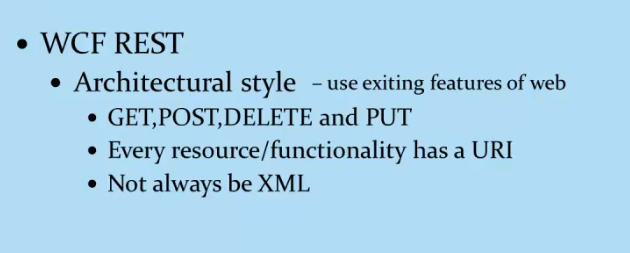
Inline action filter:



Asp.net Web API

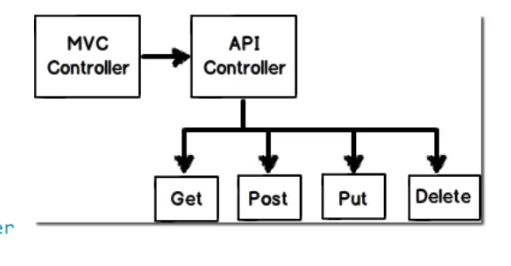
Old : WCF and web services use soap object to tunnel the data using HTTP as protocol in their transport layer.

transfer data in XML format



Technology to implement the restfull Apis.

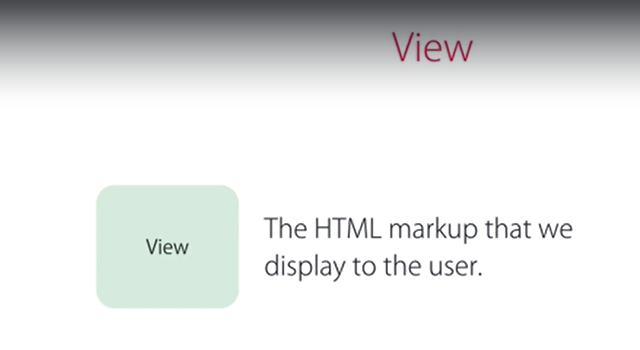
Here we have APi controller and methods inside it are HTTP Action methods that are HTTP methods

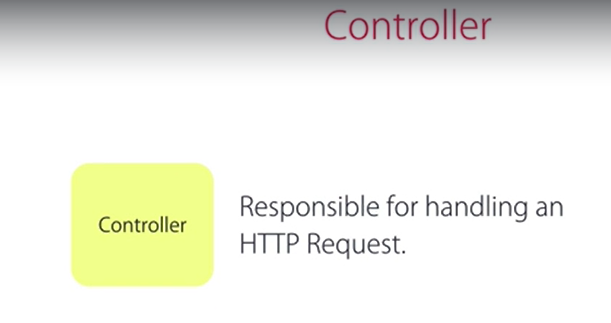


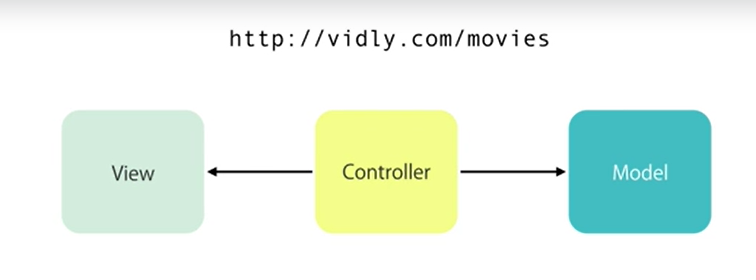
We use Api prefix in url this is because we want to segregate MVC routs from webApi routs

BY Mosh Hamadani:

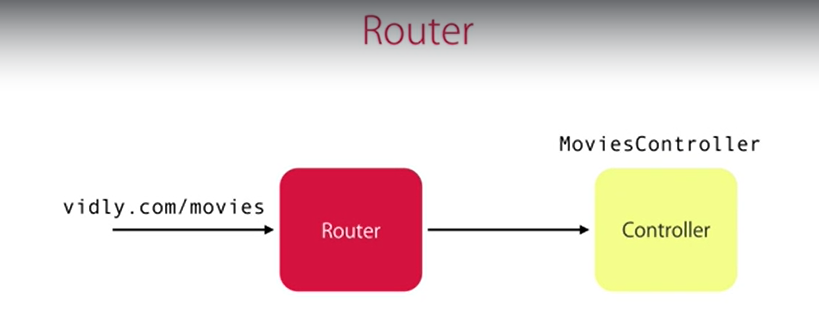








Responsible for selecting right controller and action

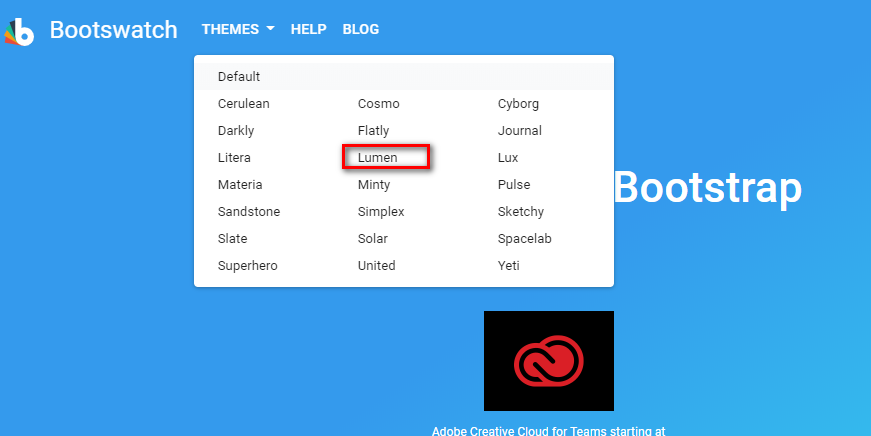


Asp.net MVC by default uses the Boot strap as UI framework

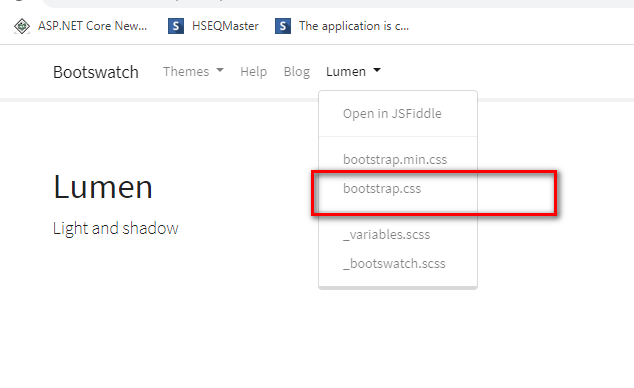
So we can change the application them as follows.

Load:

<https://bootswatch.com/>



Download

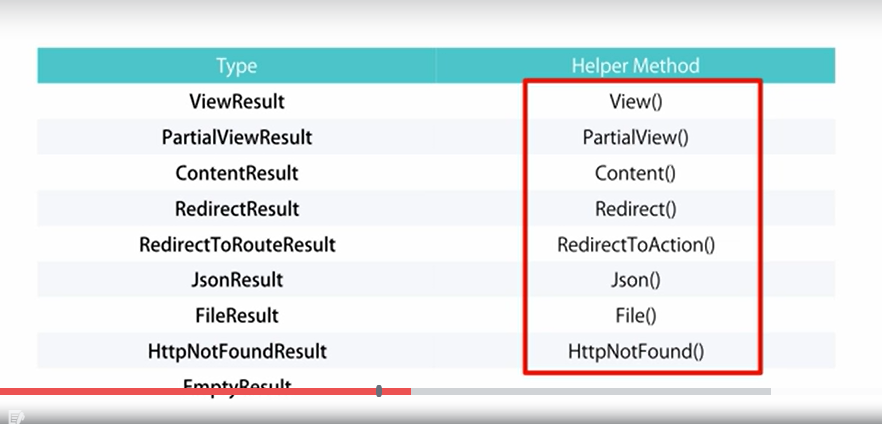


Include the downloaded Css in project with theme name

Add this in bundle.cs file

Acton result:

Base class for all result returned front eh action method





Action parameter:

Basically passing parameters from the url to action methods

Either we can integrate it as a part of url or as a query string parameter to the URL

Integrated in URL

<http://Appname/Controller/ActionName/parameter>

As query string parameter

<Http://Appname/Controller/Action?prameter1&Parameter2>

**Convention based routing:**

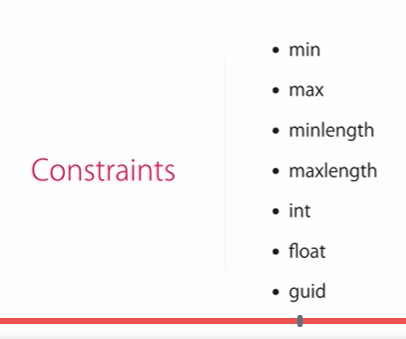
Custom routes can be defined in the rout config files where in I have to add the custom route in the beginning and the put the generic routs.

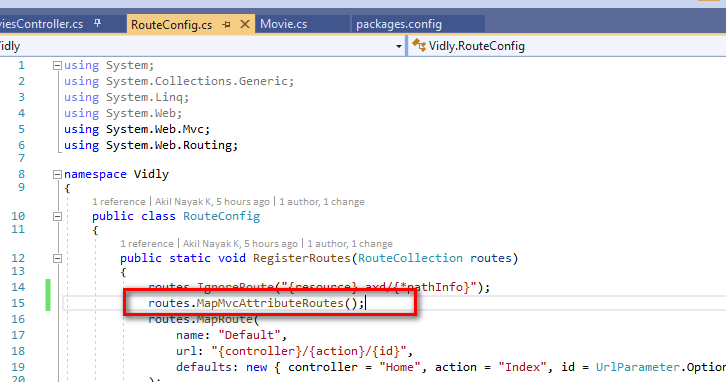
So the ordering should be always from custom to more generic.

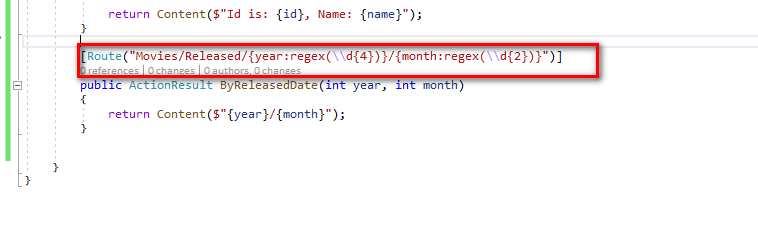
In url patter {} indicates the place holders

Attribute routing:

Add rout on top action method and enable attribute routing in rout config







Passing data to the view:

ViewData dictionary

ViewBags

View models: Models built specifically for the view basically contains all the data specific to that view. Gets data from multiple models and bound to the view.

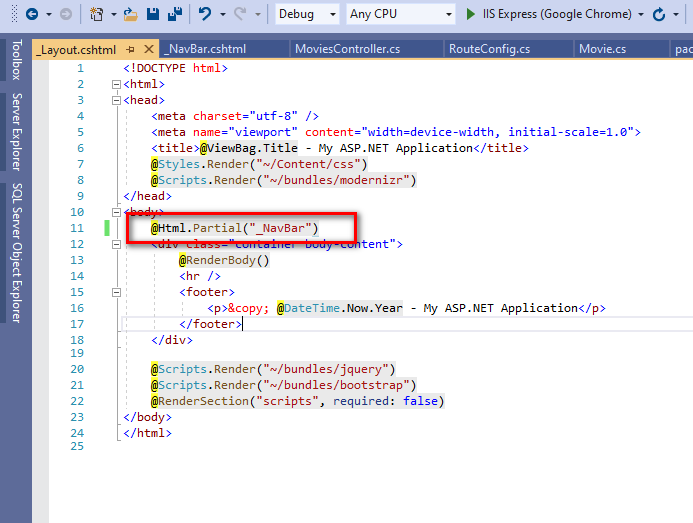
Partial views:

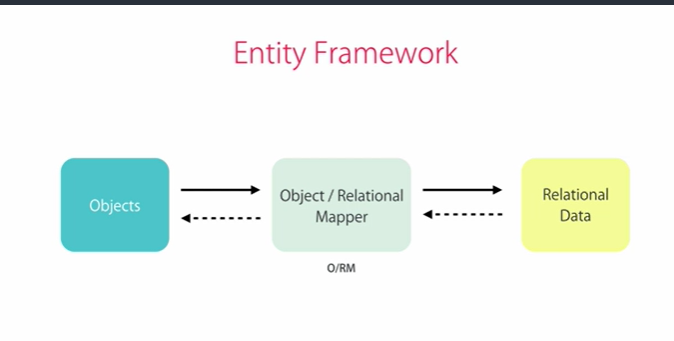
By convention partial view always prefix with (\_ (underscore)).

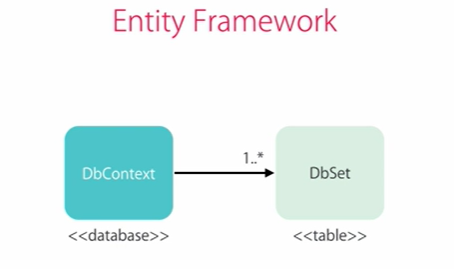
Similar to user controls which we have in asp.net with which we can reuse the controls across different module or the screens.

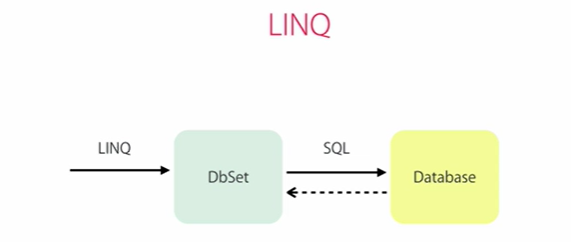
Create view with partial option selected

Render the partial view:

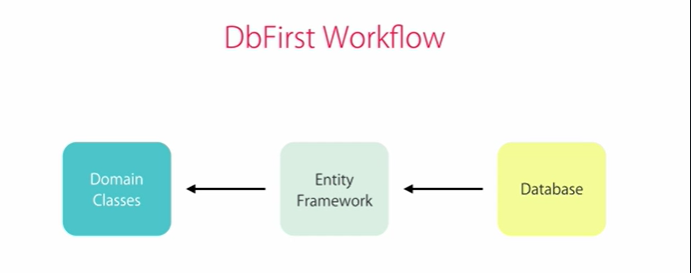


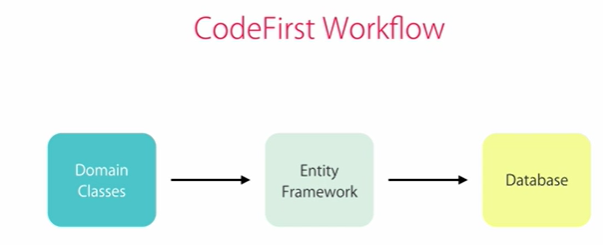






DB first vs Code first approach:



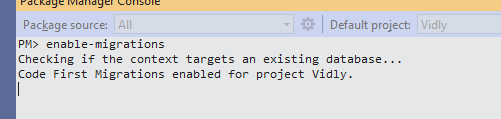


Code first migration:

Every changes we do in application we need to create a new migrations.

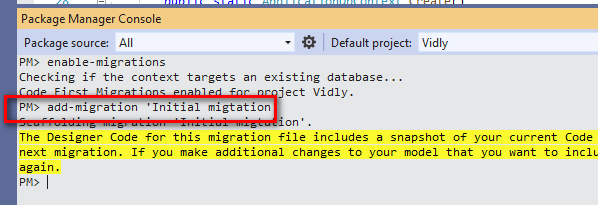
First step enable migrations:

Enable-Migrations



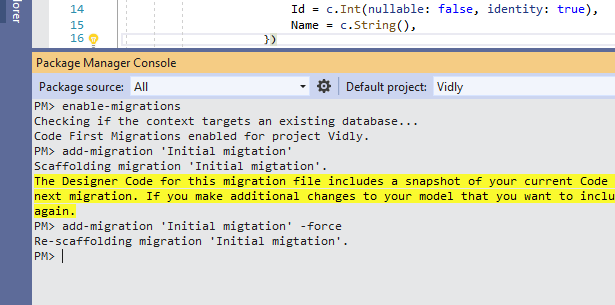
Add-migration ‘name of migration’

This will be the initial migration

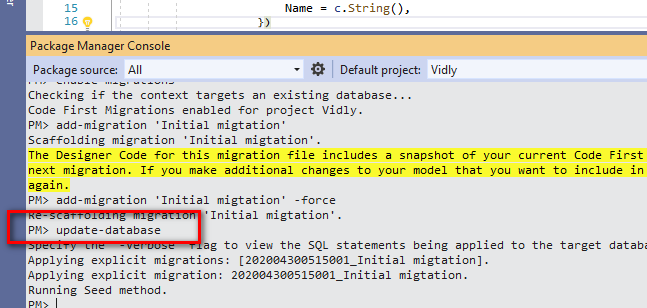


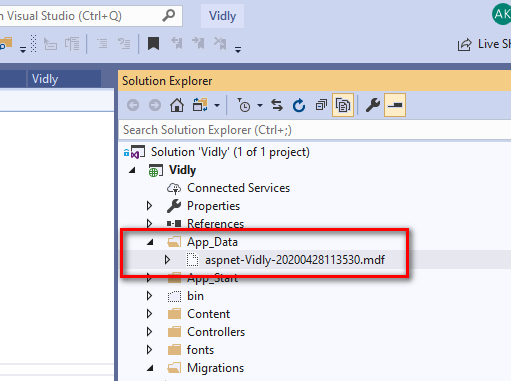
Open and see the migration file created we will find the scripts for updating db deleting relation s in db all other things

To override the existing migration

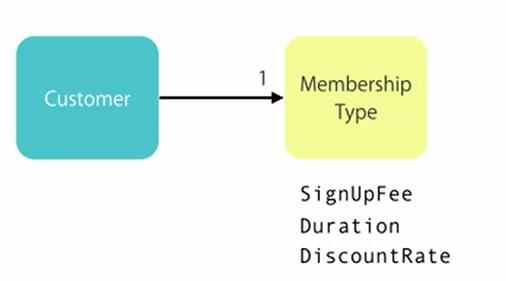


Updating changes to the DB





Add the mdf file in App\_Data folder



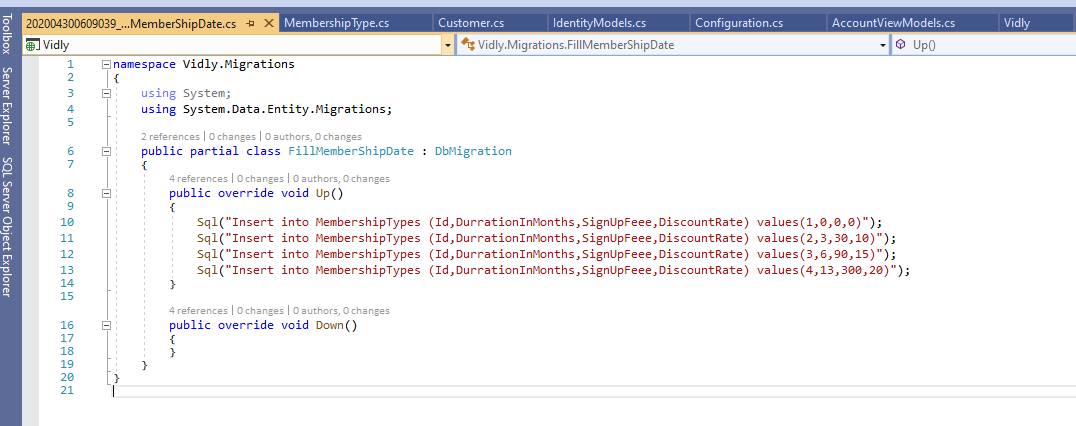
Always add small and simple migrations.

Seeding a database:

If any fixed table value always value should go from migration

Its also possible to populate table using migration

1. Add empty migration
2. Then write sql statement inside the up method of the migration



Then update the Db

Overriding the default conventions. Called data annotation

Why with this we can have constraints put on top of our columns.

Eger loading:

Using system.Data.Entity:

This will load the related entity along with the entity we are going to load.

