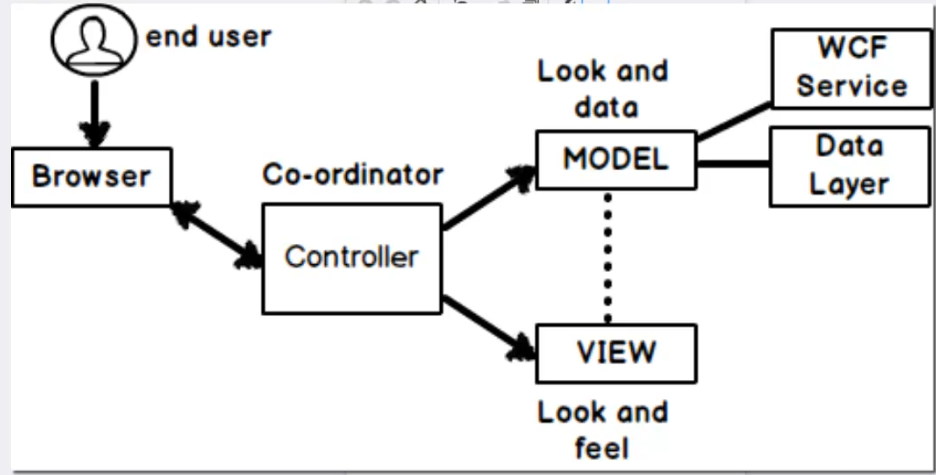
File > new > project> Asp.net application MVC

Architectural patter divided into 3 part

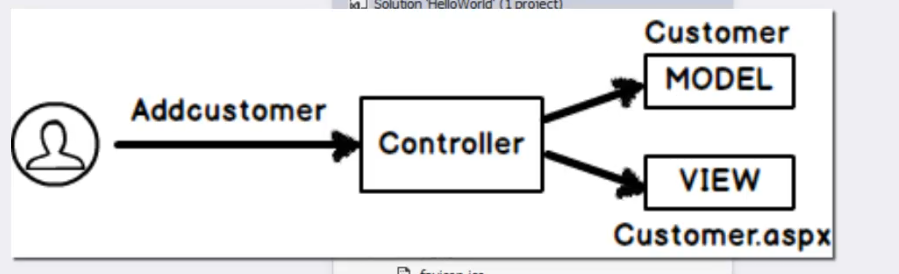
Model: supplied data and business login

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.

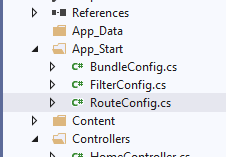
Shared folder in views folder basically contains list of all the view.

**Routing in mVC:**

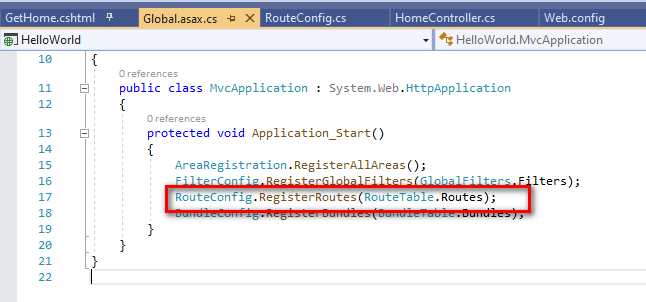
Simplifies the mv curls

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

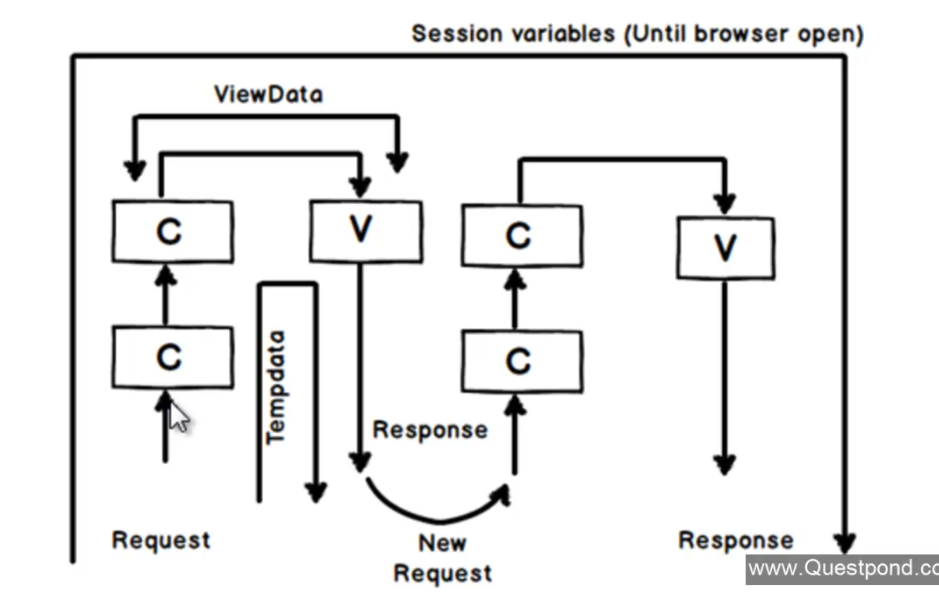
To maintain data to entire request between controller to controller and then from action to view uses temp data.

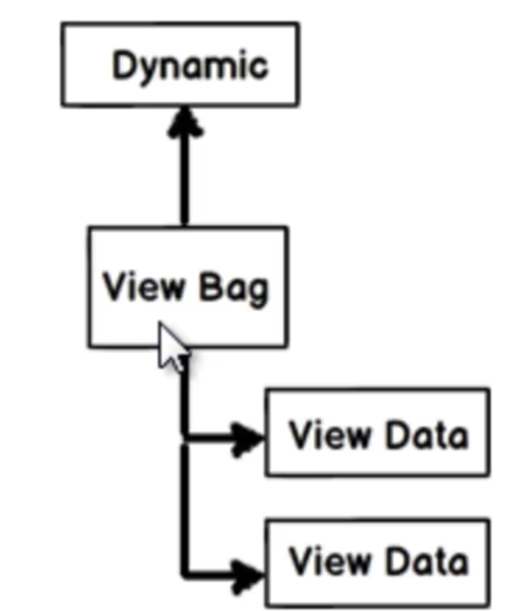
View bag is collection of view data.

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:

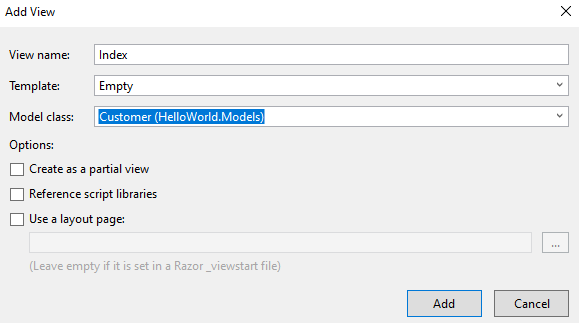
return RedirectToAction(“Action name”,”Controller name”);

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



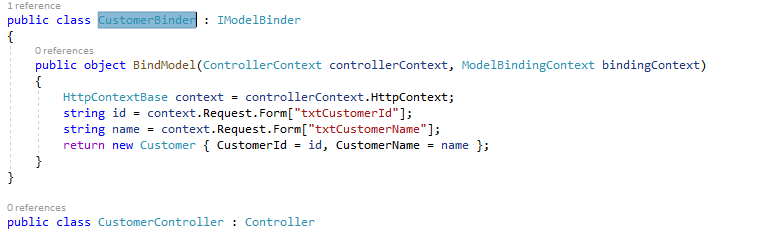
Model binders:

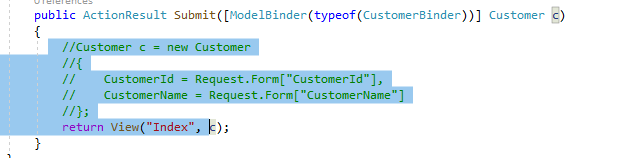
Automatic binding for this model property and form fields name should match with each other.

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

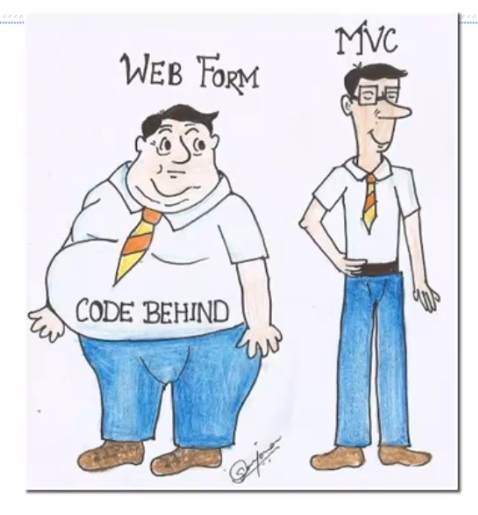
In such scenarios we need to used model binders as attribute on top of the model object bound to the controller action method

Interface iModeBinder > method BindModel





Why MVC:



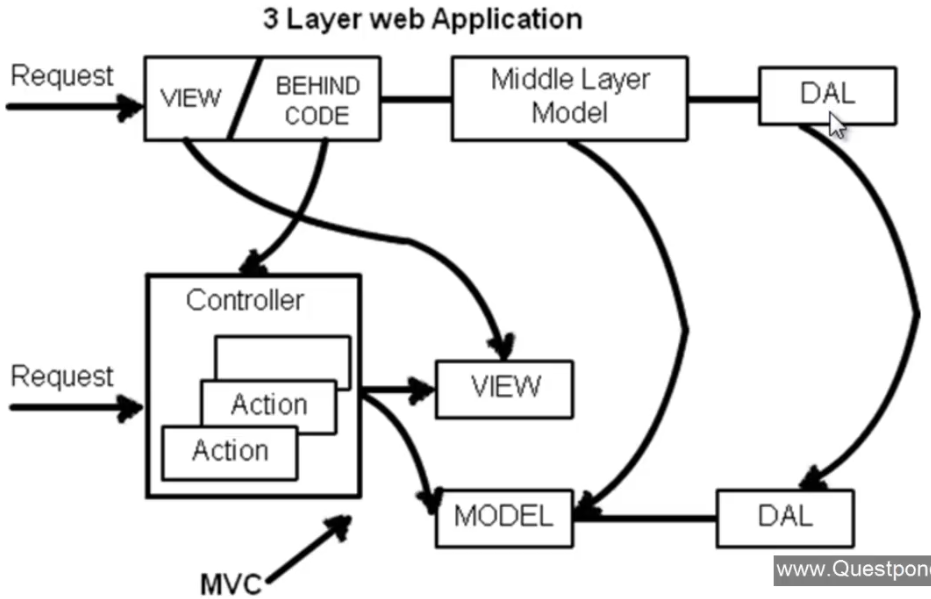
1. View based architecture, web action based architecture

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

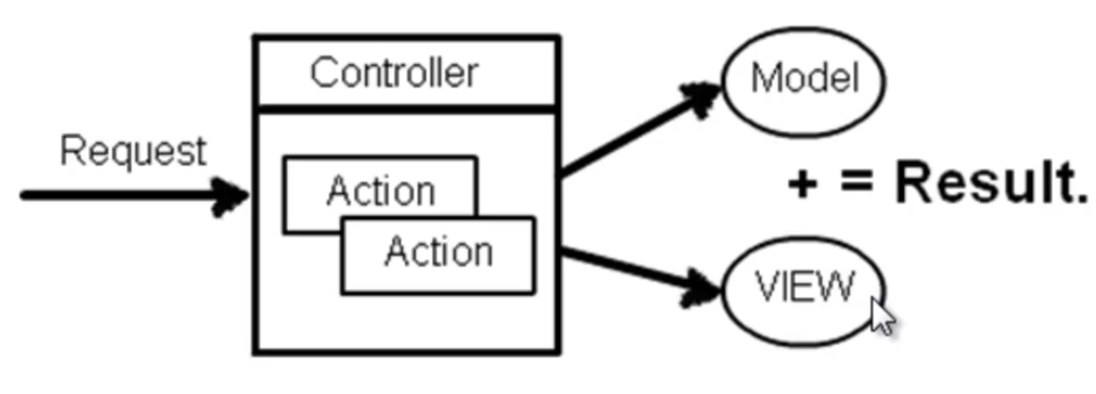
3. Fixed response type in web

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.

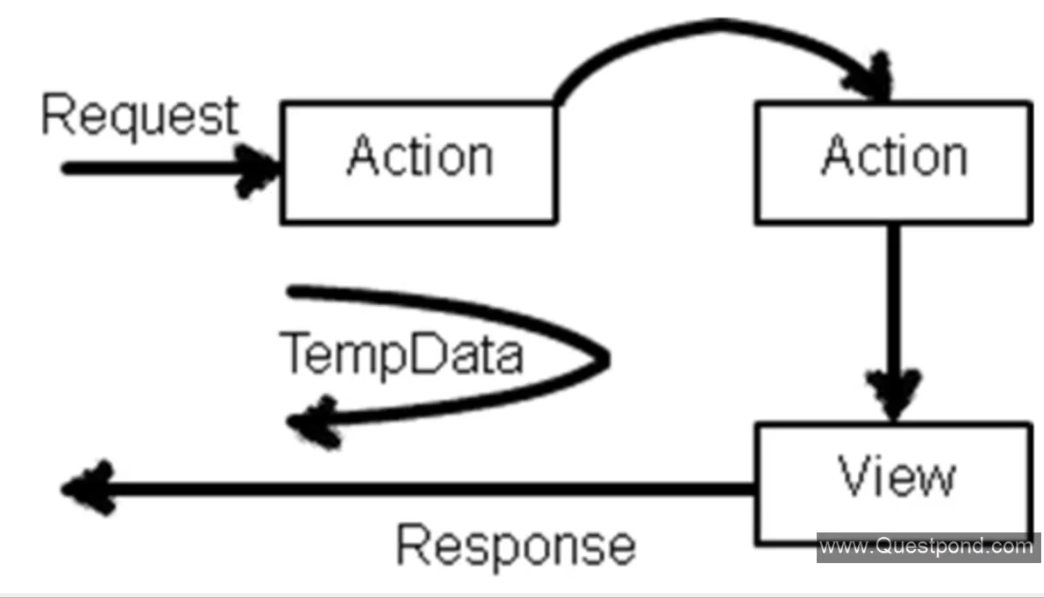


MVC

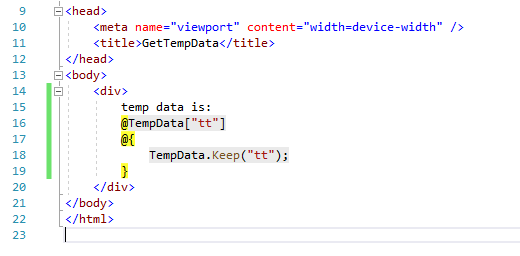


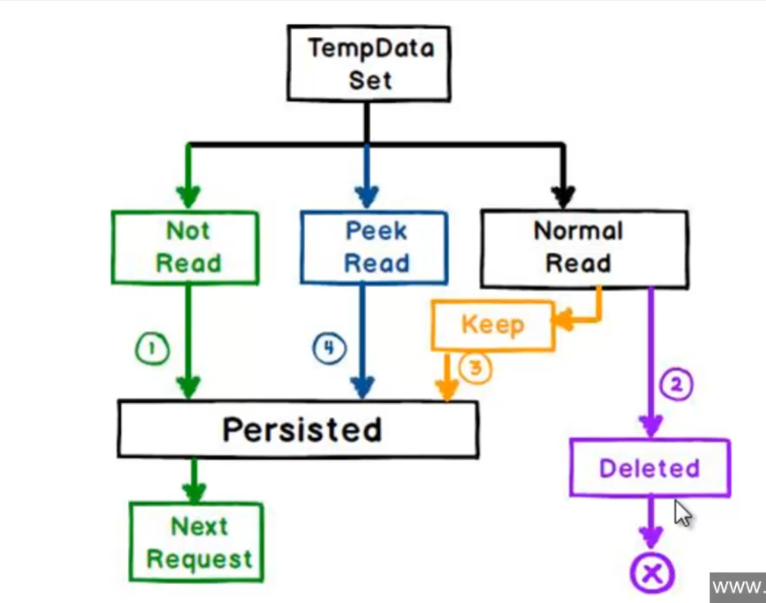
Temp data keep and peek:

Temp data can be preserved between multiple request

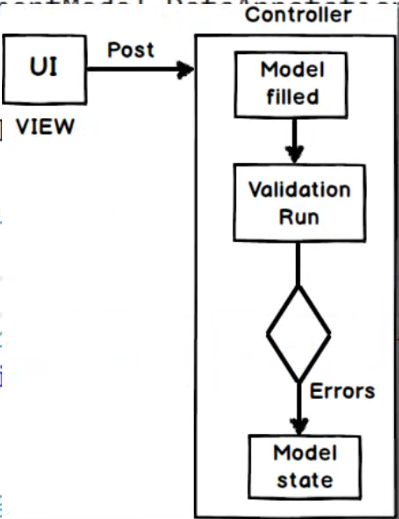


1. If the value is not read from the temp data value will persist
2. If the temp data value is read the value will be cleared or lost
3. If you run keep method on the temp data after reading its value the data will persist.
4. Read value using peek method it will persist the value.



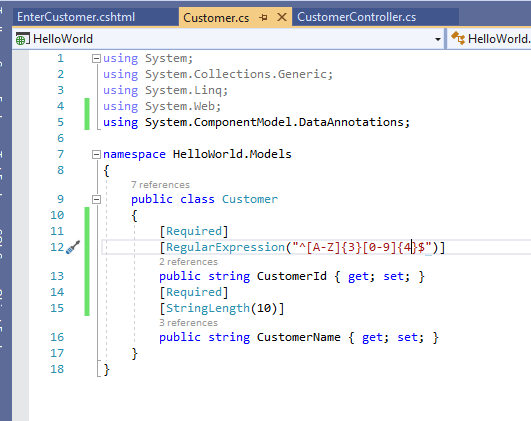


Data annotation and helper class:



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

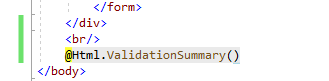


Decorated with constraints

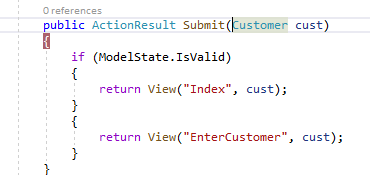
App setting:



Helper class to display validation summery

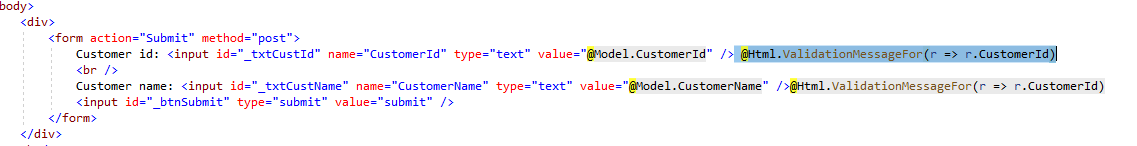


On model binding of the controller action method validation will be done and ModleState class will be set.



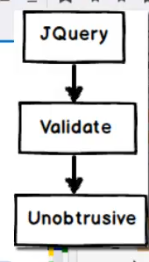
Instead of displaying the error message in on section how about displaying after each and every control.

@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.



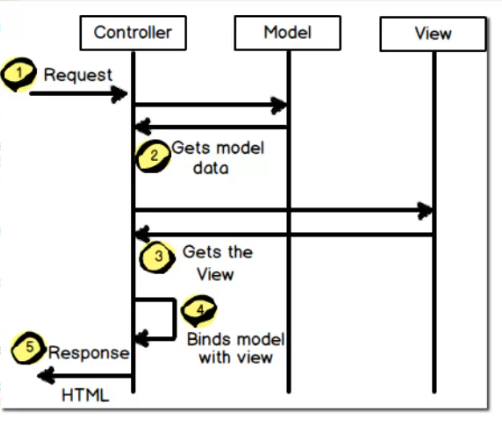
Steps:

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.



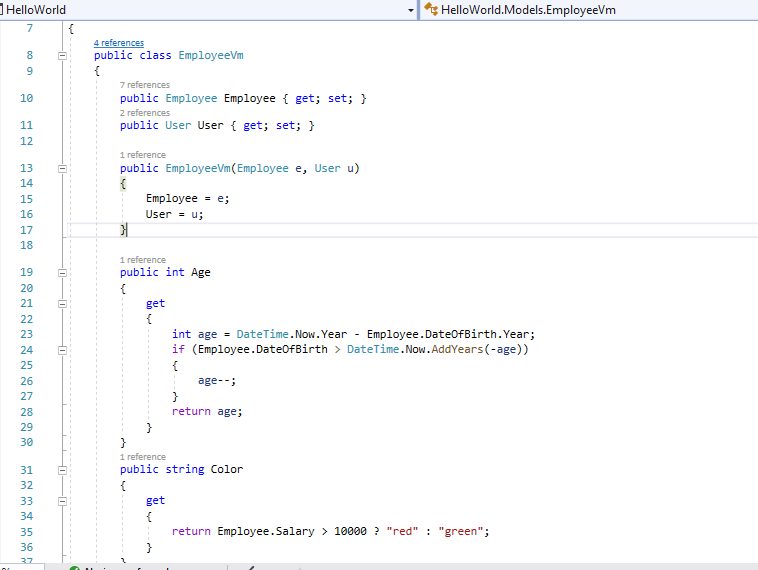
View Model:

Sequence diagram for mvc:

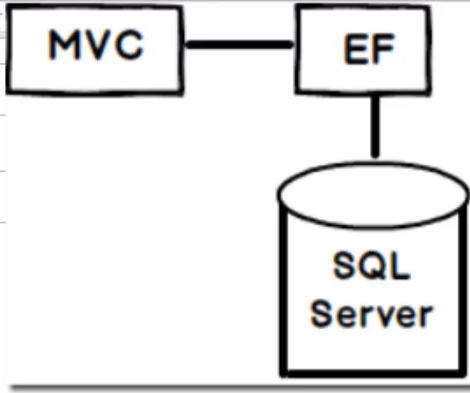


1. Make strongly typed view of multiple models
2. Data transformation logic
3. Presentation logic

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



Entity frame work in mvc



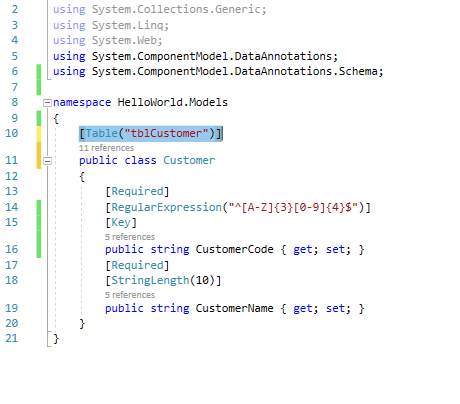
For code first approach:

1.Define db and the entity file

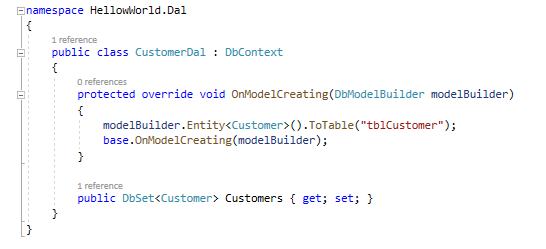
2. Add connection string in the webconfig file

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

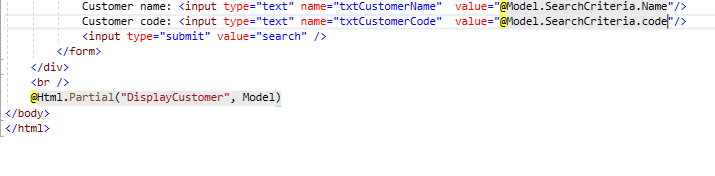
Eg: Customer application when in I have to add new customer display the list and search the customers.

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.

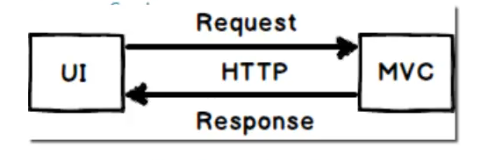
We use html helper class called HTML.Partial(“view name”, model);



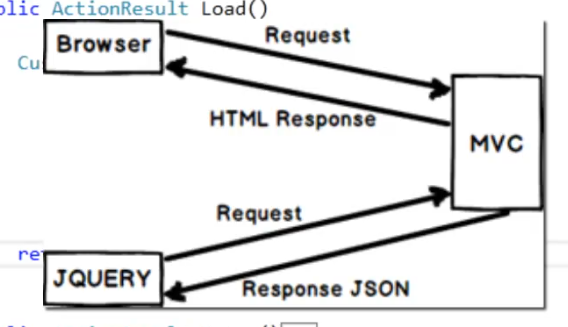
Web grid built in control to build grid



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

