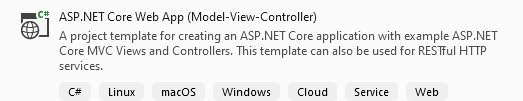
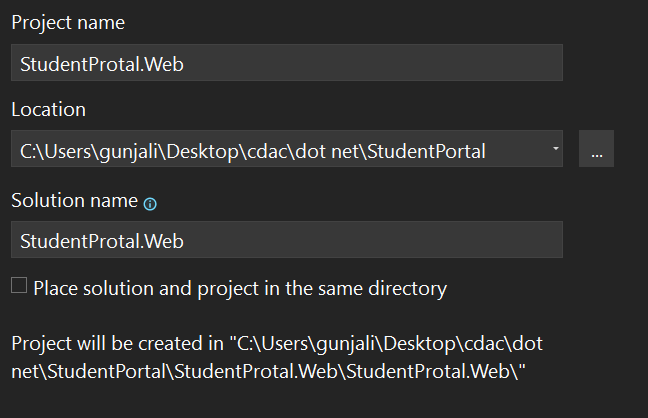
Create new folder

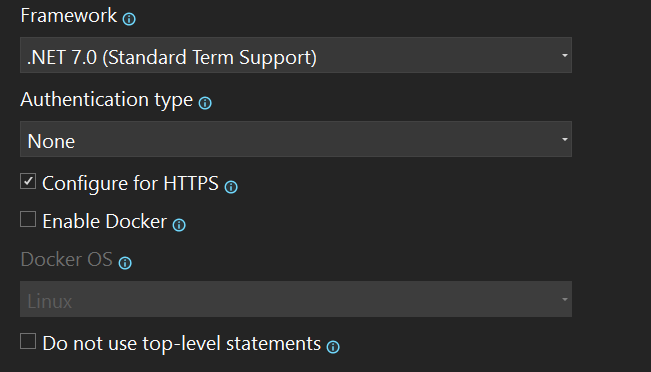
new project

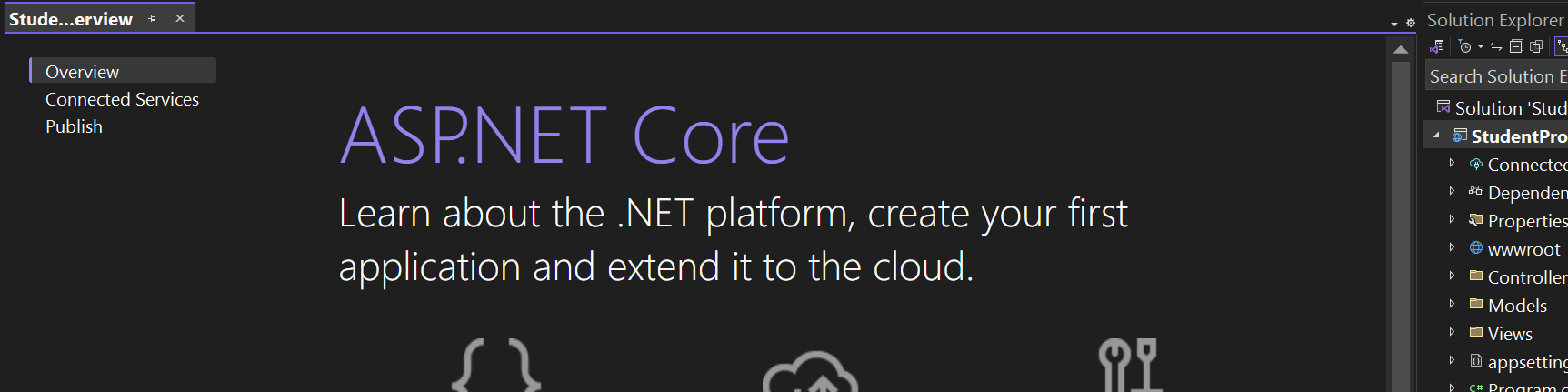




Create project folder =>copyt its path in location=>Give project name solution name







Mvc created we can tell it from model view controller folders in right

In program.cs =>we configuye

With controller we get default home controller it has 2 action methods index and privacy

In program.cs default value of route is set to home and action method set to index

So by default application will come to home controller and look for index method and when find it render index view

It goes to views folder,it goes to home folder because it is name of controller and find 2 view index and privacy

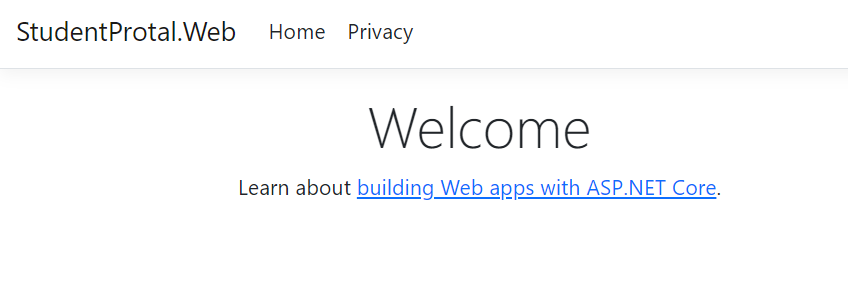
In properties launch.json(the grron http button run on below url



It loads website on this url



If notifications click no



This is home/index

Stopt the application

Now if directories not open u can go in view ->soln explorer

Store and retrieve from db

So we need to install entity framework dependencies

Install entity framework core

(it is microorm and talk to db)

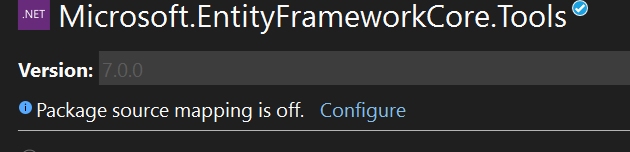
Right click on dependencies ->manage budget dependency(if cant be seen then it is in debug mode,do stop debugging)

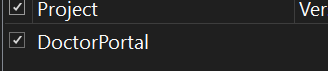
Search in browse not installed

Search tools

Click on project

Version 7.00

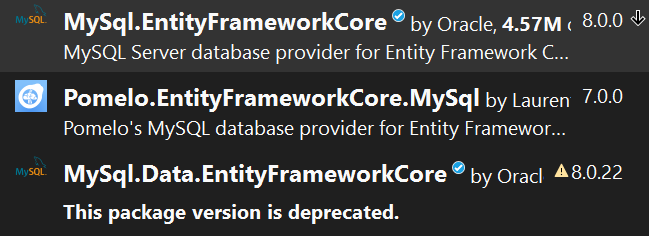




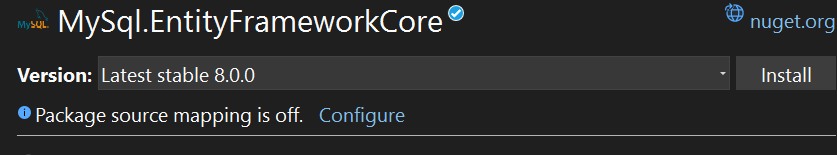
Click install

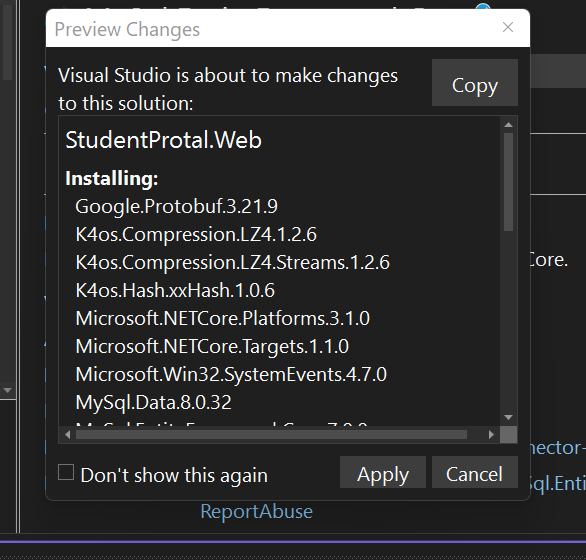
Search mysql.entity

Verion 7.0.0

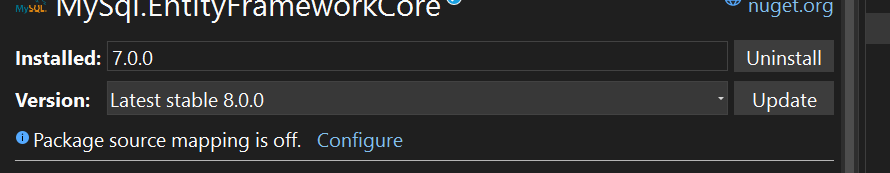


Change version and install





Apply



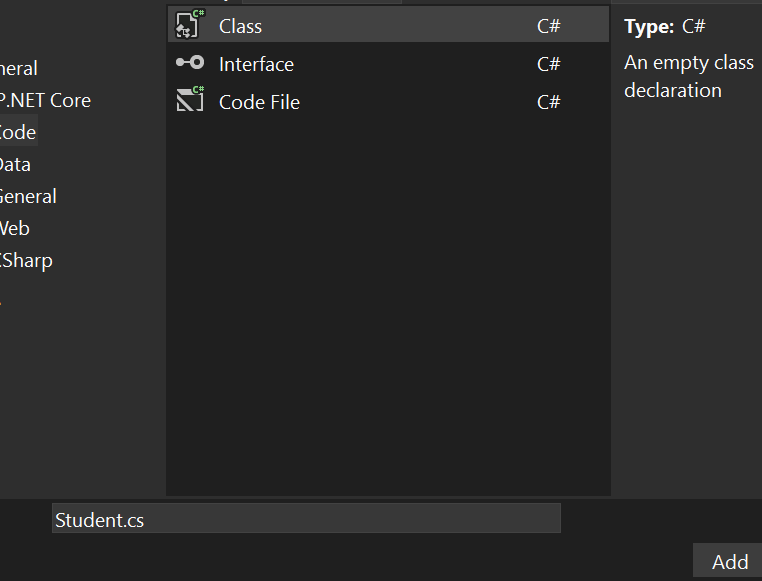
Now it is installed

Not mysql.data

Right cick on project

**Entities folder**

Right click=>addclass=>below give name book.cs



**Entity folder Student.cs**

Give field name first letter capital

Dont write private or else error

Intelisance give small

public int id { get; set; }

public string name { get; set; }

public int age { get; set; }

public int Id { get; set; }

public string Name { get; set; }

public int Age { get; set; }

No ;

public string? Name { get; set; } //for nullable=true

**Data folder**

Add new folder data

Add class

**ApplicationDbContext.cs**

1. : dbcontext

public class ApplicationDbContext : DbContext

1. Create func to connect
2. Dbcontext alread has it so override
3. Onconfiguring
4. Remove base
5. optionBuilder =>useMysql=>
6. Server user pass..

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseMySQL("server=localhost;user=root;password=root123;database=doctordb");

}

//remember not : but=

Not host but server

No port

Separate by ;

And only one big “” not far all

configure the DbContext to use a MySQL database

Override OnConfiguring: method inherited from dbcontext

DbContextOptionsBuilder:

* + Class toconfigure dbcontext
  + UseMySQL is a method of DbContextOptionsBuilder to connect to a MySQL database.
* ("server=localhost,user=root,password=root123,database=studdb") contains key-value pairs

**Mysql**

Create database and use it

**ApplicationDbContext.cs**

New function

Onmodelcreating

Keep base not remove

Primary key id

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Doctor>(entity =>

{

entity.HasKey(e => e.Id);

entity.Property(e => e.Name);

entity.Property(e => e.Salary);

});

}

* define how the Doctor class is stored in the database,what the primary key is and how specific properties are handled.
* Override onmodelcreating
* Base Method Call:
  + base.OnModelCreating(modelBuilder) calls the default configuration for the model.
  + modelBuilder.Entity<Doctor>(entity => { ... });

How modelBuilder using entity framework map to db table

Doctor is a class

Primary Key:

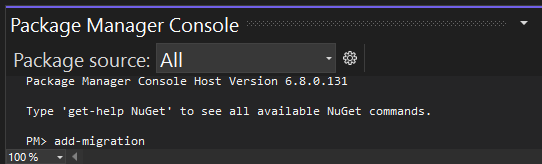
* + entity.HasKey(e => e.Id); Id property in the Doctor class should be the primary key
  + entity.Property(e => e.Name); and entity.Property(e => e.Salary); set up the Name and Salary properties of the Doctor class for storage in the database.

After function public dbset

In tools manage => package control

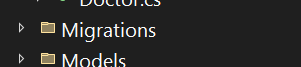
Add-migration “init”

update-database



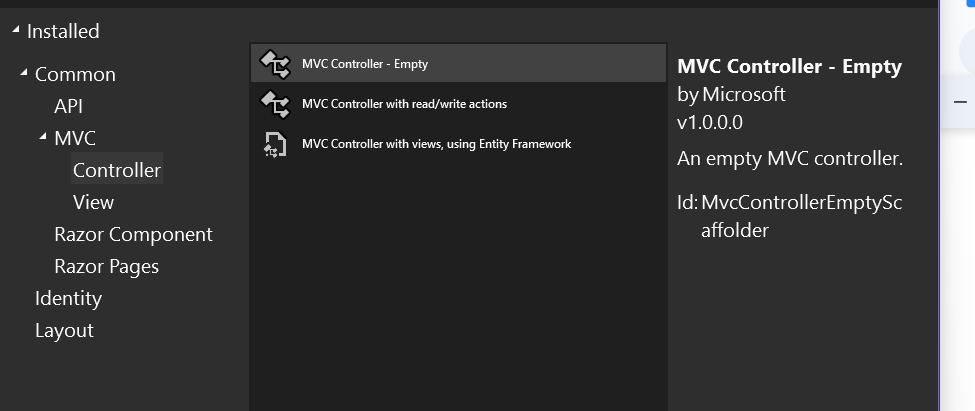
Check in db shoe tables

If succes



In **controller** =>

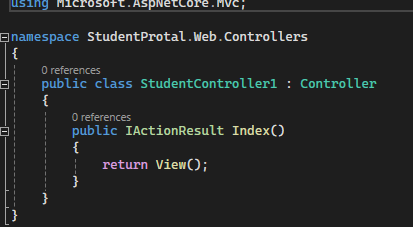
add=>controller=>



Click on add

Give name **DoctorController.cs**

It shows this



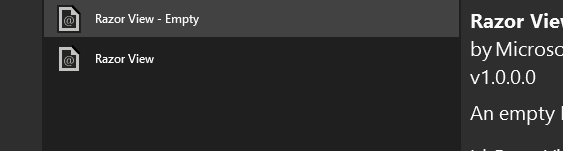
Jo method ka name h woh hi view kan name

In view (not home)

Doctor folder

Add view

Right click on view=>add view=>razor view empty=>

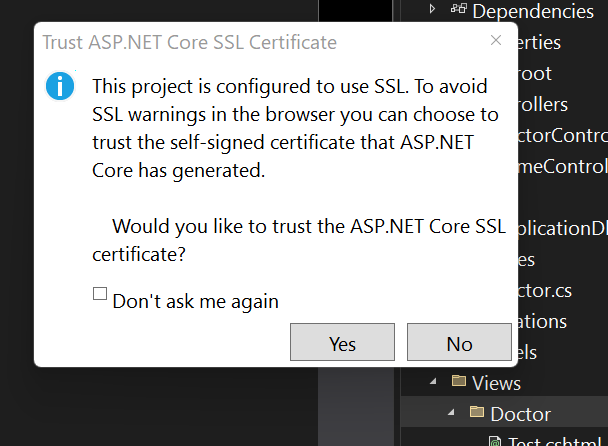


Then <h1>hi<h1>

Run for test Student.test //classname not students

If certificate yes =>no

In views add



Click yes

In **entities** add class **doctormodel**

And copy everything except id(like dto)

**Add**

1)Change index to add

2)add [HttpGet]

[HttpGet]

public IActionResult Add()

{

return View();

}

New function

Add

[HttpPost]

public IActionResult Add(Doctor doctor)

{

var context = new ApplicationDbContext();

context.Doctors.Add(doctor);

context.SaveChanges();

return View();

}

//i/p classobj

2)applicationdbcontext new obj

3)by that new obj ,in db table,add i/pobj

4)by that new onj,save

In **view** =>**Studen**t=> add view **add.cshtml**

Change name to add

**In add.cshtml**

@model DoctorPortal.Entities.Doctor

<h1>redister doctor</h1>

<form method="post">

<div>

<label for="name">Name: </label>

<input type="text" name="name" asp-for="Name" />

</div>

<div>

<label for="salary">Salary: </label>

<input type="number" name="salary" asp-for="Salary" />

</div>

<button type="submit" asp-controller="Doctor" asp-action="Add">Save</button>

</form>

@model DoctorPortal.Entities.Doctor

import

projname.entities.classname

<button type="submit" asp-controller="Doctor" asp-action="Add">Save</button>

Jab submit hogat tab doctor controller k add method main jayega

Run on browser /Doctors/add =>db

**shared=>layout**

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Student" asp-action="Test">Test</a>

</li>

**list**

[HttpGet]

public IActionResult List() {

var context = new ApplicationDbContext();

var books = context.Books.ToList();

return View(books);

}

1)get

2)list method

3)new applicationDbcontext obj

4)by that obj call dbtable list

50 return list to view

Right click on method name => add view

**List.cshtml**

@model List<BookStore.Entity.Book>

Import but with list

@\*

For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

\*@

@{

}

@model List<StudentPortal2.Entities.Student>

<h1>List</h1>

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>NAME</th>

<th></th>

<th></th>

</tr>

@foreach (var s in Model)

{

<tr>

<td value="@s.Id">@s.Id</td>

<td>@s.Name</td>

<td><a class="btn btn-outline-primary" asp-controller="Student" asp-action="Edit" asp-route-id=@s.Id>Edit</a></td>

<td>

<form asp-controller="Student" asp-action="Delete" method="post">

<input type="hidden" name="id" value="@s.Id" />

<button class="btn btn-danger" type="submit">Delete</button>

</form>

</td>

</tr>

}

</thead>

</table>

Shared=>Layout

**Submit method**

1)httpost

2)same method name add

In models add new file errorViewModel

Add method

In views=> add Student folder=>add.cshtml

In browser localhost:7054/book/add

In cs html create form

In controller add method

run

In browser give url

In db

In url student/add

Enter data submit

**In visual studio code**

Create new folder

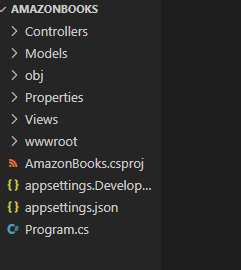
Extension-nuget paclake manager,c# dev kit,c#.net install tool

In terminal

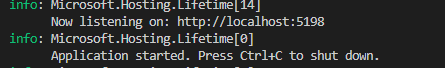
Dotnet

Dotnet -h(it shows all commands help)

Dotnet new mvc

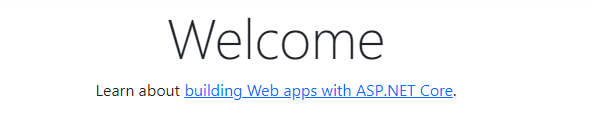


Dotnet run



Ctrl +click on that link

oPen brower



In model->Create BookEntities->

namespace AmazonBooks.Models;

public class BookEntitie{

public int Id{get;set;}

public string Title{get; set;}

public string Author{get; set;}

}

Create new folder data->applcatinDbContext->

namespace AmazonBooks.Data;

public class ApplicationDbContest{

}

For conncetion

Dotnet build

(specify connection string)

Open app.settind.json

{

"ConnectionStrings": {"DefaultConnection"},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}