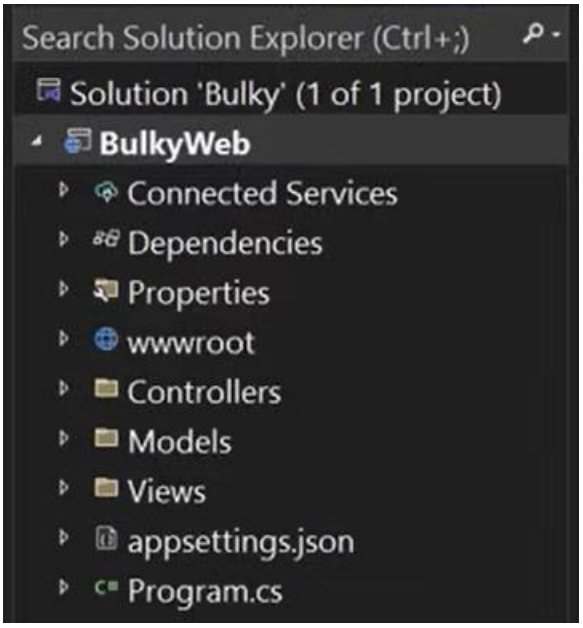
DotNET core MVC

Tutorial - [Introduction to ASP.NET Core MVC (.NET 8) (youtube.com)]

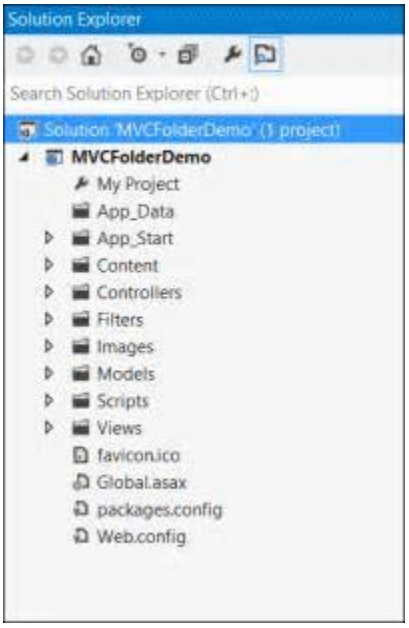
.Net core – fast, open-source, cross platform, Built-in Dependency Injection, Cloud friendly, high performance

Foler Structure –

.net core MVC



.net MVC -



A solution can have multiple projectss

Project File – target Frameowork, nullable, Implicit using

Dependencies – Frameworks -EntityFramework core

Properties – Launch Settings.json – IIS Settings, Profiles – Http, https. IIS

Environment variable – global variable

wwwroot – has static contents – CSS, js, lib. favicon (can have PPT, files, images)

Data – in EFCore, has DBContext.cs files

Migrations – in EFCore, migration files will be stored here

Controllers – handles the user request and acts an interface between Model and View

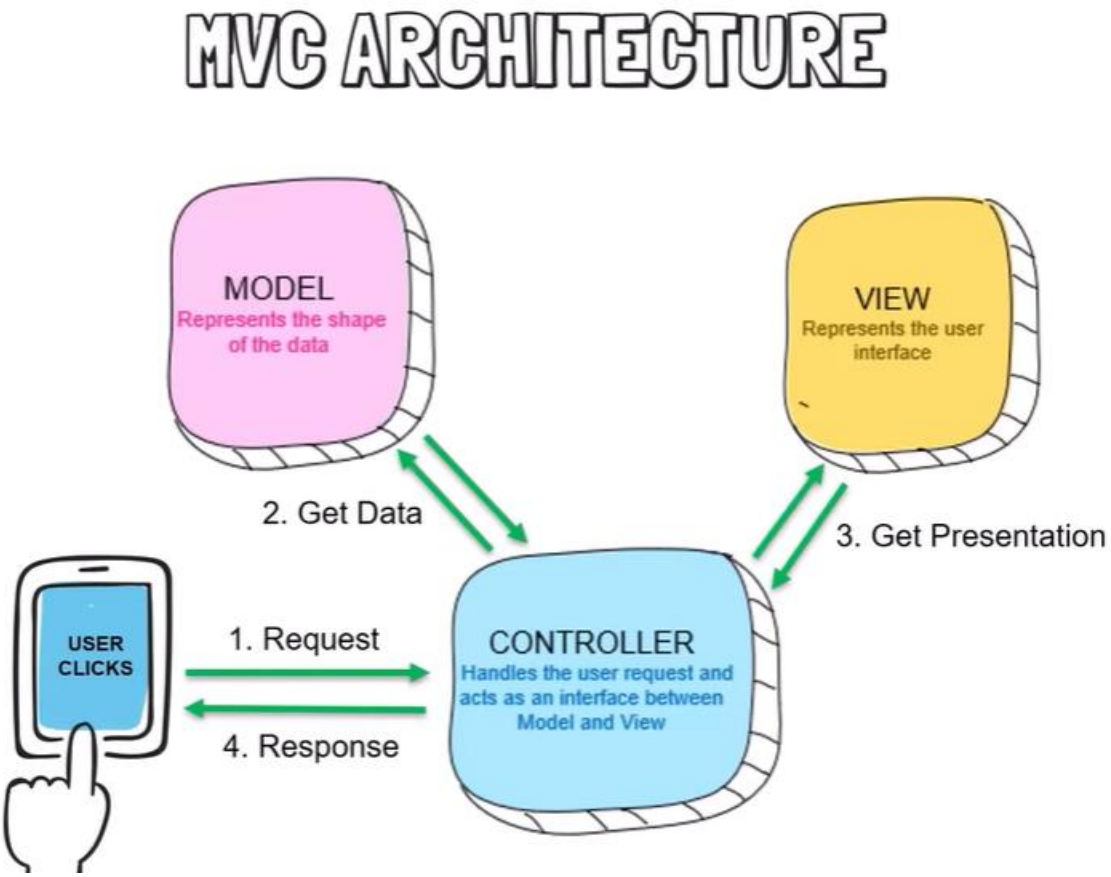
Models – Represents shape of data ,a class file

Views – User interface

appsettings.json / appsettings.Development.json – connection string, any secret key

Program.cs – (also has startup.cs in old .net core) builder, app var builder= WebApplication.CreateBuilder(args);

- **Add services** – builder.Services.AddControllerWithView();, var app=builder.Build();
- **Configure HTTP pipeline** – when a request comes to an application, how it should be handeled
    - App.useHttpRedirection()
    - App.UseStaticFiles();
    - App.UseRouting();
    - App.UseAuthorization();
    - App.MapControllerRoute() – default route
    - App.Run()



Action Methods – endpoints in controller

Routing – states what controller and which action method should be used

The URL pattern for routing is considered after the domain name.

- https://localhost:55555/Category/Index/3
- https://localhost:55555/{controller}/{action}/{id}

URL	Controller	Action	Id
https://localhost:55555/Category/Index	Category	Index	Null
https://localhost:55555/Category	Category	Index	Null
https://localhost:55555/Category/Edit/3	Category	Edit	3
https://localhost:55555/Product/Details/3	Product	Details	3

Controller –

- Should be in controller folder
- Should have controller suffix
- Have action methos

Model –

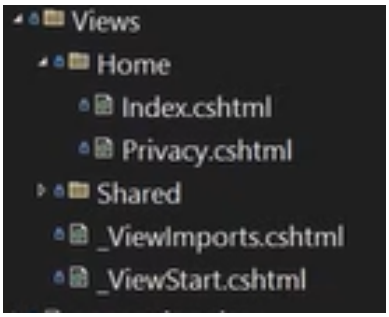
- Cs files has properties
- Represents data
- Not always needed for view
- Data annotation to mark a property as primary key

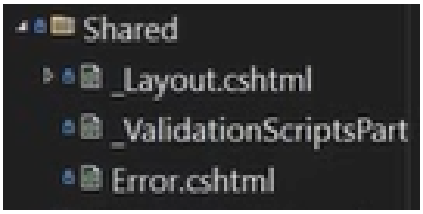
```
using System.ComponentModel.DataAnnotations;

namespace BulkyWeb.Models
{
    public class Category
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public int DisplayOrder { get; set; }
    }
}
```

Views –

- Cshtml files

- Same folder structure is same as controller
- _layout.cshtml is master page, has renderbody() to render whatever passed by view from controller





_ValidationScriptsPartial – a partial view, has reference for validation scripts

_ViewStart.cshtml – configure default page

```
@{
    Layout = "_Layout";
}
```

_viewImports.cshtml – for global usage in views

```
@using BulkyWeb
@using BulkyWeb.Models
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

IActionResults – interface declared in .net, which has all possible return types

Entity Framework core –

- Add packages using package manager
    - Microsoft.EntityFrameworkcore
    - Microsoft.EntityFrameworkcore.SqlSever
    - Microsoft.EntityFrameworkcore.Tools – for using commands like add-migration
- Add connection string in appsettings.json

```
appsettings.json  Program.cs  Applica...text.cs
Schema: https://json.schemastore.org/appsettings.json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=.;Database=Bulky;Trusted_Connection=True;TrustServerCertificate=True"
  }
}
```

- Add new folder – Data and create a class dbContext.cs. Dervie from DBContext and add DBSet property

```
using Microsoft.EntityFrameworkCore;

namespace BulkyWeb.Data
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
        {

        }

        public DbSet<Category> Categories { get; set; }
    }
}
```

- Seeding data in DBContext (Optional)

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Category>().HasData(
        new Category { Id = 1, Name = "Action", DisplayOrder = 1 },
        new Category { Id = 2, Name = "SciFi", DisplayOrder = 2 },
        new Category { Id = 3, Name = "History", DisplayOrder = 3 }
        );
}
```

- Register DB context in Program.cs

```
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<ApplicationDbContext>(options=>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

- In package manager console – add-migration name
- Update-database

Retrieving values from DB

```
public class CategoryController : Controller
{
    private readonly ApplicationDbContext _db;
    public CategoryController(ApplicationDbContext db)
    {
        _db= db;
    }
    public IActionResult Index()
    {
        List<Category> objCategoryList = _db.Categories.ToList();
        return View();
    }
}
```