**TUTORIALS START**

-First Open Visual Studio 2022

-Create a new project and type MVC and select C# ASP.Net Core, then click next

-Name the Project Name: BulkyBookWeb

And the Solution Name: BulkyBook

Must be different name but similar.

-Next select the Framework we selected: .NET 6.0(Long Term Support)

-Authentication type : None

**-Check** Configure to HTTPS.

-Then Create the project

IN the solution explorer we can see the folder created by default.

-Once we right click the **BulkyBookWeb** project name and select “**Edit project file**” we can see the “**Configuration**” which we can see the target framework.

Inside we can see the **<ItemGroup></ItemGroup>** which contains all the **NuGet packages** that we are using inside the project. In future we will be installing more pacjages when we connect to database using the **Entity Framework** core and so on. When we add **Nuget** package new entry will be made into csproj file or the project file. Now usually you do not work with the project file. It is always good to know that we have all the packages and the references listed in the project file if incase you need to acces that.

We can expand the **Dependencies** in **Solution Explorer** and we can see the **Packages** we saw previously inside our **project file** above example.

**launchSettings.json**

Next we see the **Properties** folder which we can see the **“launchSettings.json”** We have the different profiles “Using” which we can run our application. We can see the **BulkyBookWeb** profile and **IIS Express Profile.** If we use the **IIS Express** profile, we know the port number that will be used here is explicitly defined above **“sslPort”: 44351.** If we us the **BulkyBookWeb**, in that case, it will use **localhost:7059** and **localhost:5112** base on **https** or **http**. If we run it in visual studio 2022 we can see an command and prompt pop and it will launch the website on **Port:7059** we can see the logs in Command and prompt. If we select the run through **IIS Express** it will use the **port:44351.** The default is using the BulkyBookWeb but, we will be changing that and use the IIS express later on.

**wwwroot** (folder) [**CSS,JavaScript,Images and Libraries stored inside ( wwwroot folder)**]

NOTE: *Does not have any C# Files inside.*

We will see all the static files of our project. So any **static files** like **CSS**, **JavaScript**, **images**, or any **libraries**, everything will go inside the **wwwroot** folder. The **wwwroot** folder **will not have any C# Files,** this folder is only meant to serve the static files of our application. So we will be using the static folder extensively throughout the course, when we are adding some JS or some images or any other libraries. The **wwwroot** folder will be the root folder of our application. Always remember, if we ever have to add any **static file**, it will always go inside the **wwwroot** folder.

**appsettings.json**

This is the file in which we will be **adding all of the connection strings and secrets of our application**, like we might have some **API KEYS**, we might have some **SendGrid keys**, we have **stripe payment keys** any of the **static secret keys** that we want to save, we will be storing them inside **appsettings.json**. If we expand **appsettings.json** we can see the **appsettings.Development.json** so we can create new JSON files and it will **automatically bundle them inside appsettings.json**. Like if we create for another environment, **appsettings.staging.json** then we can have **appsettings.production.json** all of them will be bundle in **one umbrella**. And then based on the environment variable, we can configure it to use the different **appsettings file**. Because connection string for a database in development will be different if we compare that to staging preview of production(Enterprise). So that way, we can go into those configuration. We will be using just **appsettings.json** right now, because we will be working with the **localhost**. Also in production, there are **multiple ways of saving secrets** like you can add them to the **Azure Storage** world and much more. To get started, **we need to remember, all of our application secrets must be inside appsettings.json**, and not direclty inside any of our CS or Class Files. Now we will add our **connection string** next.

**Program.cs**

This is the file that is responsible for running the application. When we open this file we can see that we have a variable builder “**var builder = WebApplication.CreateBuilder(args);**” where the **WebApplication.CreateBuilder** is passed with the built in arguments **(args).** When you run with the dotnet command, you can pass custom arguments here if you want, with that, it will configure the application, and it will create the web application builder object. Now in the previous topic, we saw that we can use **dependency injection** with **.Net Core**. When we want to **register** anything with our **dependency injection** container, we will be doing that here:

builder.Services.AddControllersWithViews();

var app = builder.Build();

So let’s say if we want to register our database or email or anything else, we will have to do that between the builder. And before we call build on the builder object. So right here, we are just adding one service to the container which is : **builder.Services.AddControllersWithViews();** we are adding the service in the container because we are using MVC application for our project. If we we’re using razor pages, then service **builder.Services.AddControllersWithViews();** will be different.Now in the future, when we configure database in our project, and we add that to dependency injection, we will be adding a new service here in our container => **builder.Services.AddControllersWithViews();** our **DBcontext**. If we are working with any version prior to .NET 6 or even some of the initial preview versions of .NET 6, then this file was divided into a separate startup.cs class file. And the services that we add to container were inside a method configure services. And everything from line 9 onwards was inside a configure method. So what we have on the top is we will be adding services to our container, then we need to configure request pipeline. And that pipeline will be configured from this section:

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");