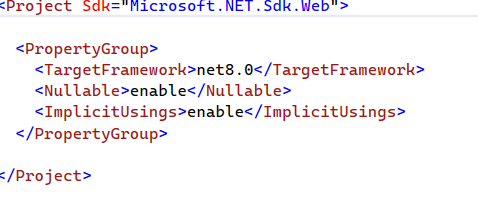
**.Net Core**

**Csproj File**

****

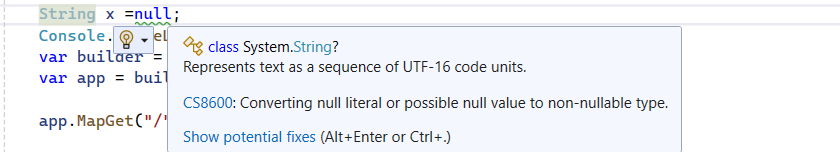
**<Nullable>enable</Nullable>:**

This line enables nullable reference types for the project. This feature was introduced in C# 8.0 and helps prevent null reference exceptions by making reference types (like a string, class, etc.) either explicitly nullable or non-nullable. When Nullable is set to enable:

* The compiler will give warnings if a potentially null value is not handled properly.
* It makes code safer and more resilient to null-related runtime errors by clearly distinguishing which variables are allowed to be null.

**Eg.** When enabled

String x = null will give below warning, it will ask to make string explicitly nullable and declare it as String? x = null



###### ****<ImplicitUsings>enable</ImplicitUsings>****

The ImplicitUsings setting is enabled. This feature, introduced in .NET 6, automatically includes commonly used using directives (e.g., System, System.Collections.Generic, System.Linq, Microsoft.AspNetCore.Http, etc.) without explicitly adding them to every file. This makes the project files cleaner and ensures that standard namespaces are included without requiring manual addition to each file.

**Understanding program.cs class**

public class Program

{

public static void Main(string[] args)

{

//This line initializes a new instance of the WebApplication class with preconfigured defaults.

//WebApplicationBuilder instance is responsible for configuring essential services

//(like Logging, Configuration, MVC, Web API, Dependency Injection, etc.) preconfigured defaults such as

// Set up Web Server(IIS, or Kestrel)

//Host the Application(InProcess or OutOfProcess)

//Logging(debugging and console logging)

//Configuration(How to access the Data from Configuration Files)

//Dependency Injection Container(registering built -in and custom services)

var builder = WebApplication.CreateBuilder(args);

//After configuring the essential services, when the Build() method is called on the WebApplicationBuilder instance,

//the actual WebApplication instance is built.The result, the app, represents the application itself.

//From this point, the application is ready to set up routes, configure the middleware component, and start handling requests, but it isn’t running yet.

var app = builder.Build();

// The below example code configures a single endpoint using the MapGet method.

// With the MapGet endpoint, when an HTTP GET request is sent to the application root URL /,

// the request delegate executes, i.e., the statement() => “Hello World!” will execute, and Hello World!will

// be written to the HTTP response.

app.MapGet("/", () => "Hello World!");

//This line starts the web server and begins listening for incoming requests.

//Without app.Run(), the web server will not start.The Run() method keeps the application running until manually stopped,

//and it is responsible for handling incoming HTTP requests.

app.Run();

}

}

**In Process Hosting model**

In In Process hosting model the application runs inside IIS worker process. This is default hosting model for ASP.Net core apps running on windows IIS or IIS express. Advantages are better performance coz there is no need for communication between iis and external process

**Out Of Process Hosting model**

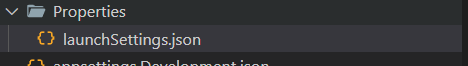
In out of process hosting model ASP.Net core application works in a separate worker process in Kestrel web server and servers like IIS,Apache,Nginx works as reverse proxy and forward Incoming HTTP requests to Kestrel web server.

We can define In process or out of process hosting model in csproj file

**<AspNetCoreHostingModel>**OutOfProcess**</AspNetCoreHostingModel>**

**LaunchSettings.Json**

This file is found in the properties folder in the .net core project.



LaunchSettings.Json is a json based configuration file that is used to configure how the application starts during development.

It is primarily used by VS,VS Code, Dotnet CLI(dotnet run) command to configure how to launch the application.

**Key Settings:**

**PROFILES** : Definesmultiple launch profiles for differene scenerios (IIS Express,Kestrel or both)

**Environment Variables:** Specifies environment variables available when the application runs

**Application URL:** Defines the URLs available when application runs.

**LaunchBrowser:** Defines if browser needs to be opened automatically when application runs