

How is .NET Core different from .NET framework ?

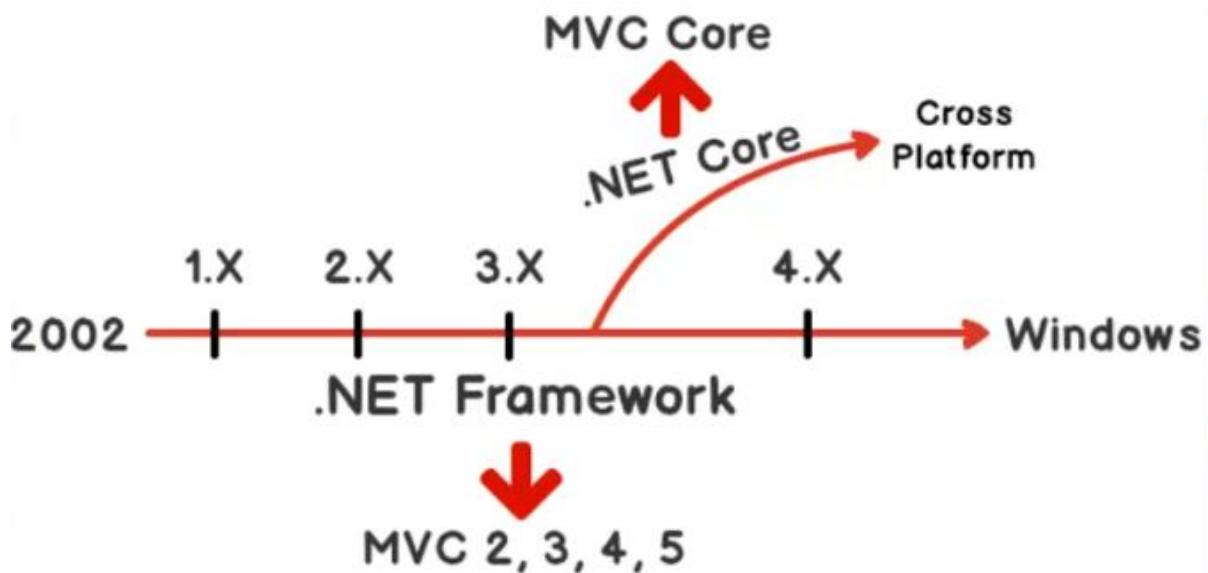
What is Dotnet CLI ?

How to create a Dotnet Core application using .Net CLI ?

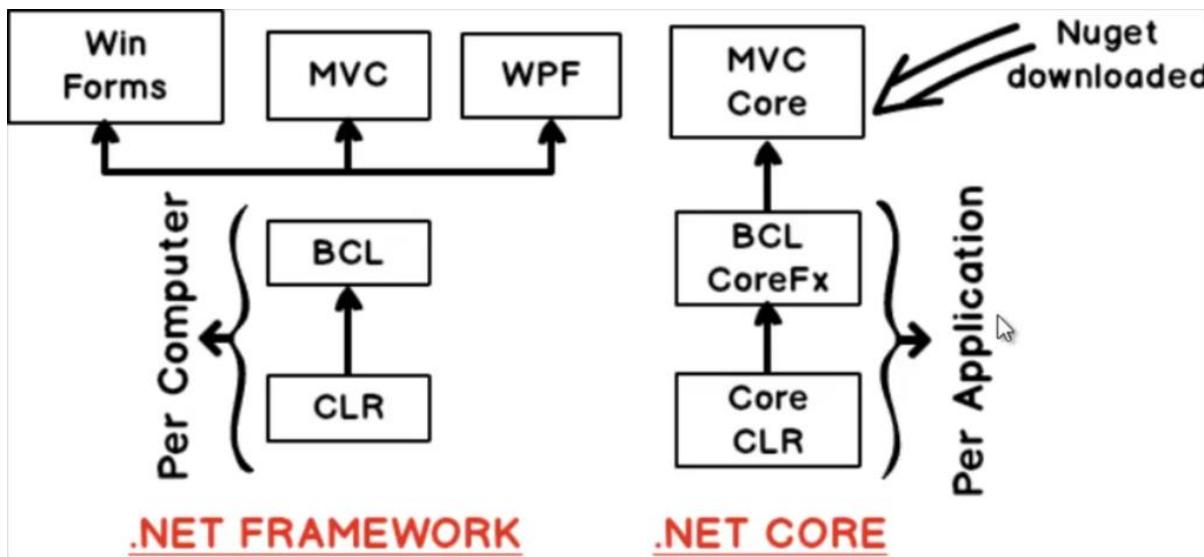
What is the need of PCL ?

What are the issues with PCL ?

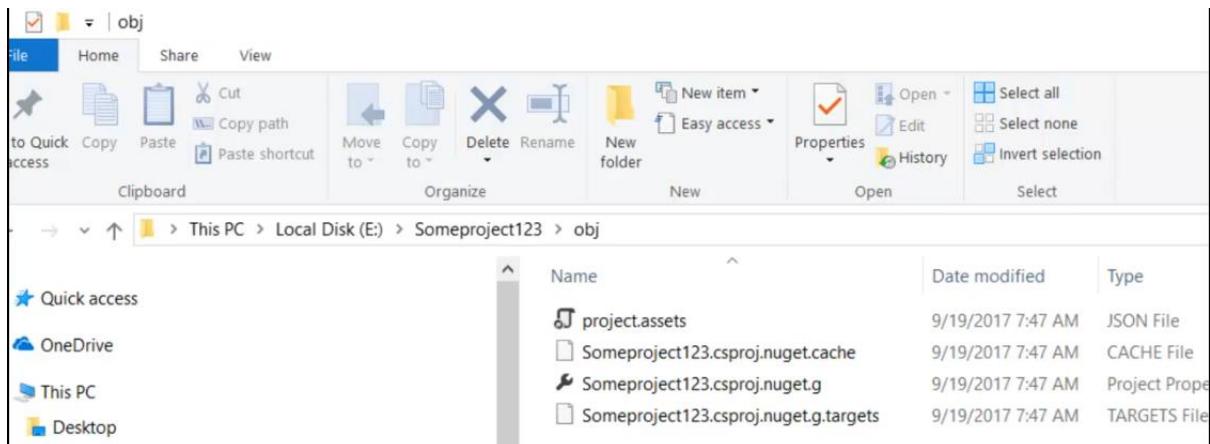
Explain .NET Standards ?



.NET Core is a cross-platform version of .NET for building websites, services, and console apps.



NuGet is the mechanism to bring in the component in your project.



```
E:\Someproject123>dotnet new console
The template "Console Application" was created successfully.

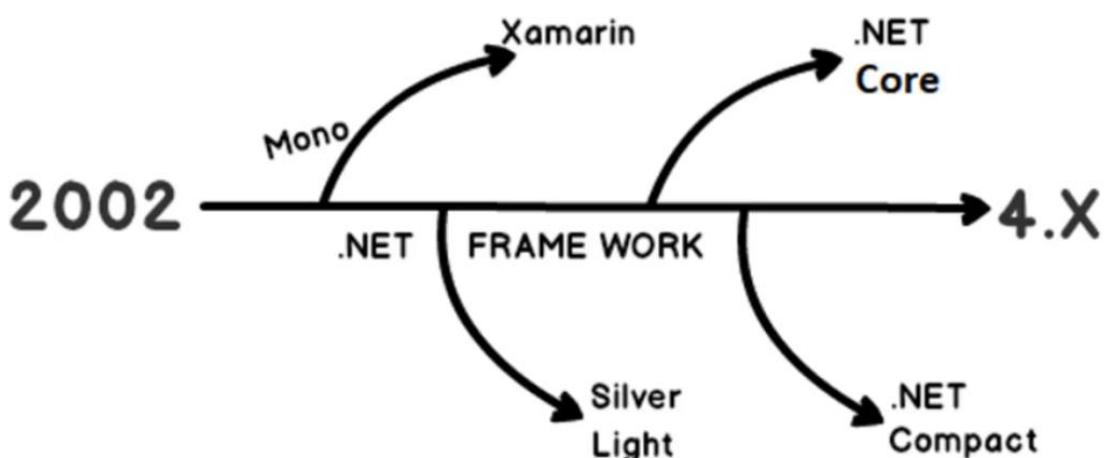
Processing post-creation actions...
Running 'dotnet restore' on E:\Someproject123\Someproject123.csproj...
Restoring packages for E:\Someproject123\Someproject123.csproj...
Generating MSBuild file E:\Someproject123\obj\Someproject123.csproj.nuget.g.targets
Generating MSBuild file E:\Someproject123\obj\Someproject123.csproj.nuget.g.targets
Restore completed in 563.34 ms for E:\Someproject123\Someproject123.csproj
```

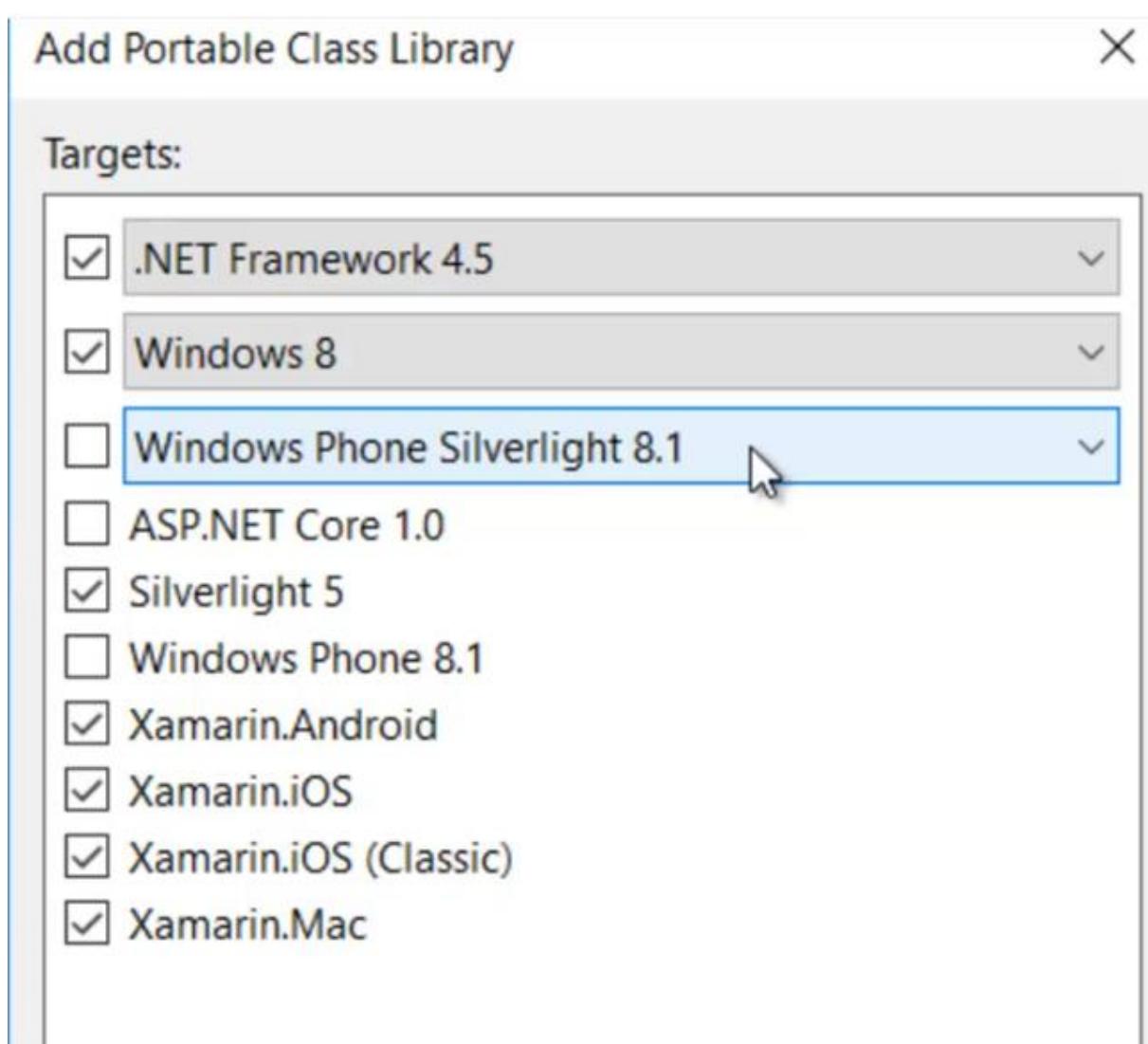
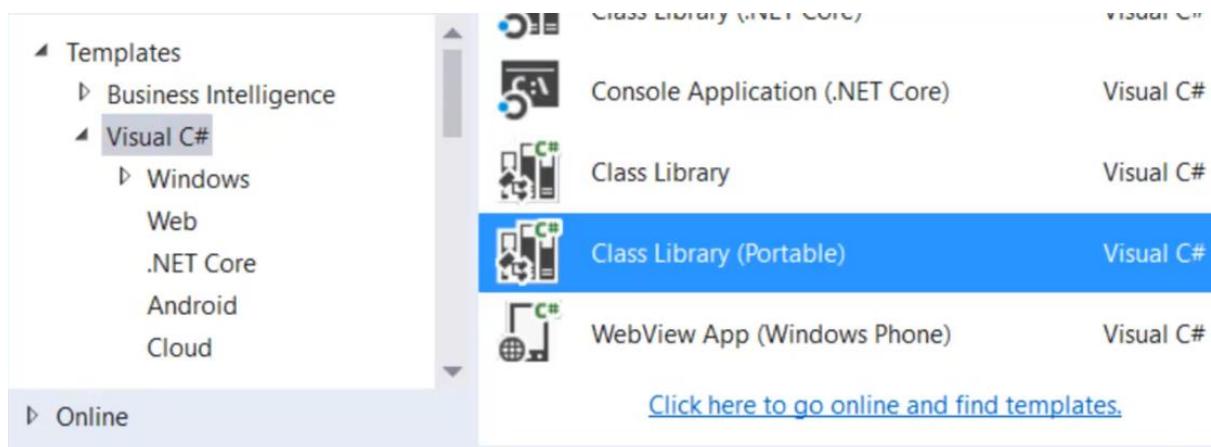
Restore succeeded.

```
E:\Someproject123>dotnet build
Microsoft (R) Build Engine version 15.3.409.57025 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Someproject123 -> E:\Someproject123\bin\Debug\netcoreapp2.0\Someproject1

Build succeeded.
  0 Warning(s)
```





If suppose new family of .net introduced , then I have to again re compile my project

Add Portable Class Library

Targets:

.NET Framework 4.5

Windows 8

Windows Phone Silverlight 8.1

ASP.NET Core 1.0

Silverlight 5

Windows Phone 8.1

Xamarin.Android

Xamarin.iOS

Xamarin.iOS (Classic)

Xamarin.Mac

```
ClassLibrary3.csproj  ↗ ×
 7      <Platform Condition="$(Platform) == ''>AnyCPU</Platform>
 8      <ProjectGuid>{0bef8be4-de15-4c13-b5ec-f7bd3aa00b72}</ProjectG
 9      <OutputType>Library</OutputType>
10     <AppDesignerFolder>Properties</AppDesignerFolder>
11     <RootNamespace>ClassLibrary3</RootNamespace>
12     <AssemblyName>ClassLibrary3</AssemblyName>
13     <DefaultLanguage>en-US</DefaultLanguage>
14     <FileAlignment>512</FileAlignment>
15     <ProjectTypeGuids>{786C830F-07A1-408B-BD7F-6EE04809D6DB};{FAE
16     <TargetFrameworkProfile>Profile47</TargetFrameworkProfile>
17     <TargetFrameworkVersion>v4.0</TargetFrameworkVersion>
18   </PropertyGroup>
19   <PropertyGroup Condition="$(Configuration)|$(Platform) == 'D
20     <DebugSymbols>true</DebugSymbols>
21     <DebugType>full</DebugType>
22     <Optimize>false</Optimize>
```

Whether to emit symbols (boolean)

Portable Class Library (PCL) profiles as of **VS2015 Update 3**.

profile

Filter

Include legacy portable profiles

Show full framework names

#	Frameworks	NuGet Target
Profile5	.NET Framework 4.0 Windows 8.0	portable-net4+win8
Profile6	.NET Framework 4.0.3 Windows 8.0	portable-net403+win8
Profile7	.NET Framework 4.5 Windows 8.0	portable-net45+win8
Profile14	.NET Framework 4.0	portable-net4+sl50

platforms you intend to run on. For instance, if you target .NET Standard 1.1.

The .NET Standard is a formal specification of .NET APIs that are intended to be available on all .NET implementations. The motivation behind the .NET Standard is establishing greater uniformity in the .NET ecosystem.

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	vNext

platforms you intend to run on. For instance, if you target .NET Standard 1.1.

The problem is centralized control of standards.

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0
.NET Framework	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	vNext

Understanding MVC architecture.

Creating a MVC core project.

Need of Program.cs and startup.cs file.

Importance of appsettings.json file.

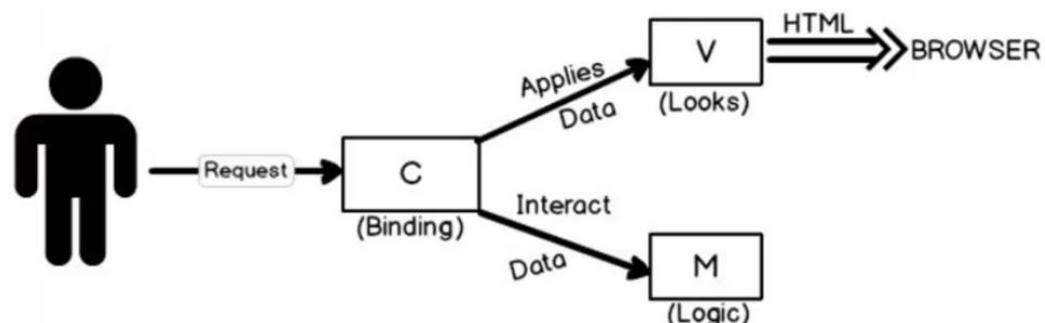
How to read configuration data ?.

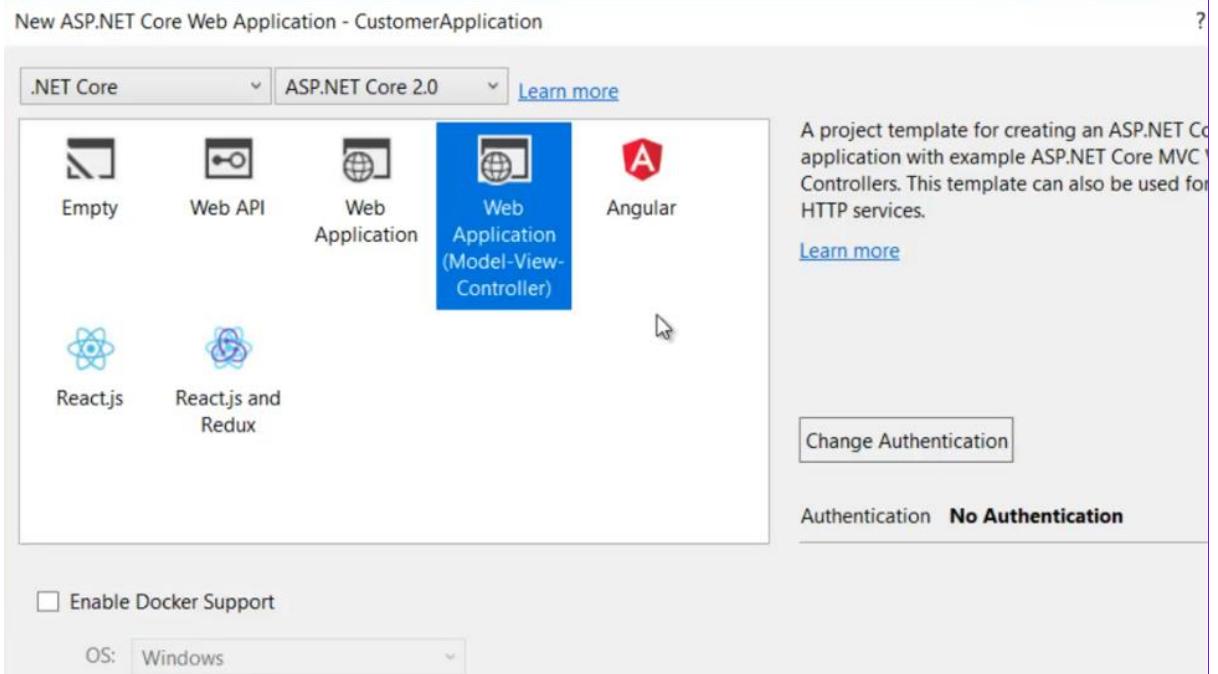
Kestrel webserver and reverse proxy concept.

Configuration and Configuration Services.

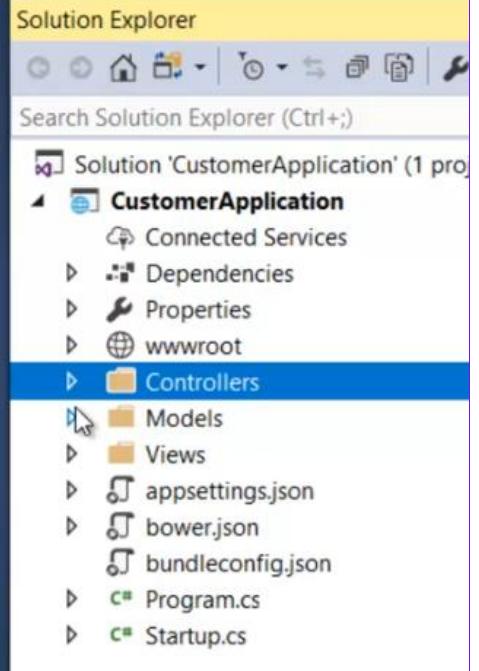
Revising the full flow of MVC startup process

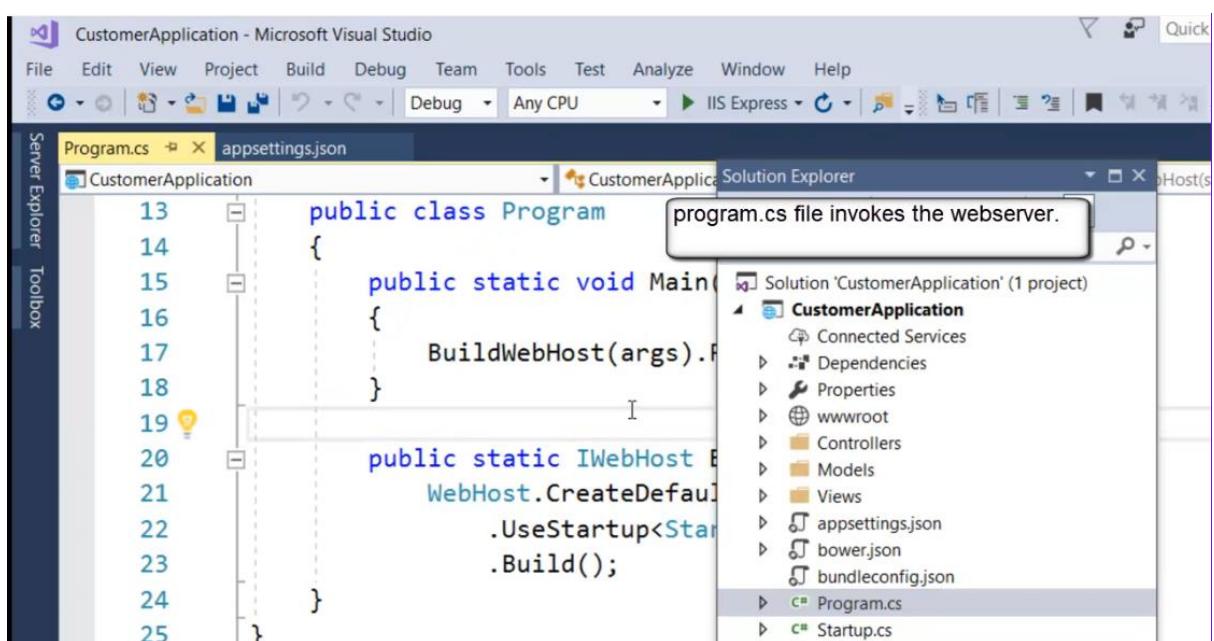
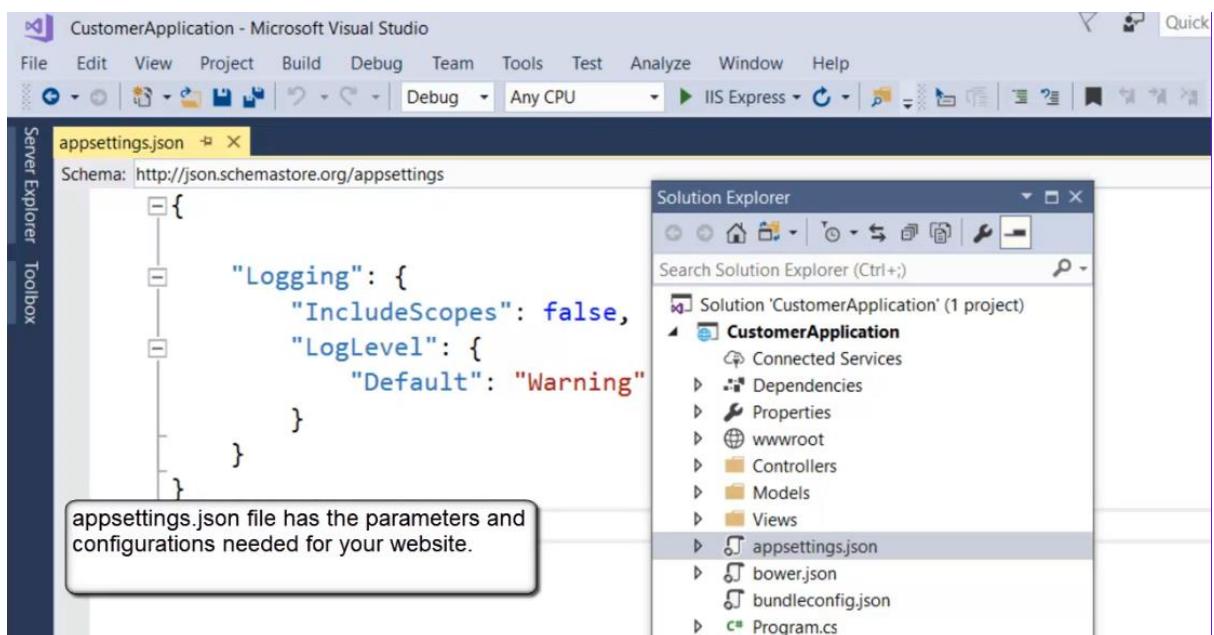
C :- Controller , M :- Model , V :- View





Controller - This will have binding code.
Model :- This will have classes of model.
Views :- This has the HTML code.
WWWRoot :- This has static content.
Startup :- Loads the initial configurations.





CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Help

Startup.cs Program.cs appsettings.json

CustomerApplication

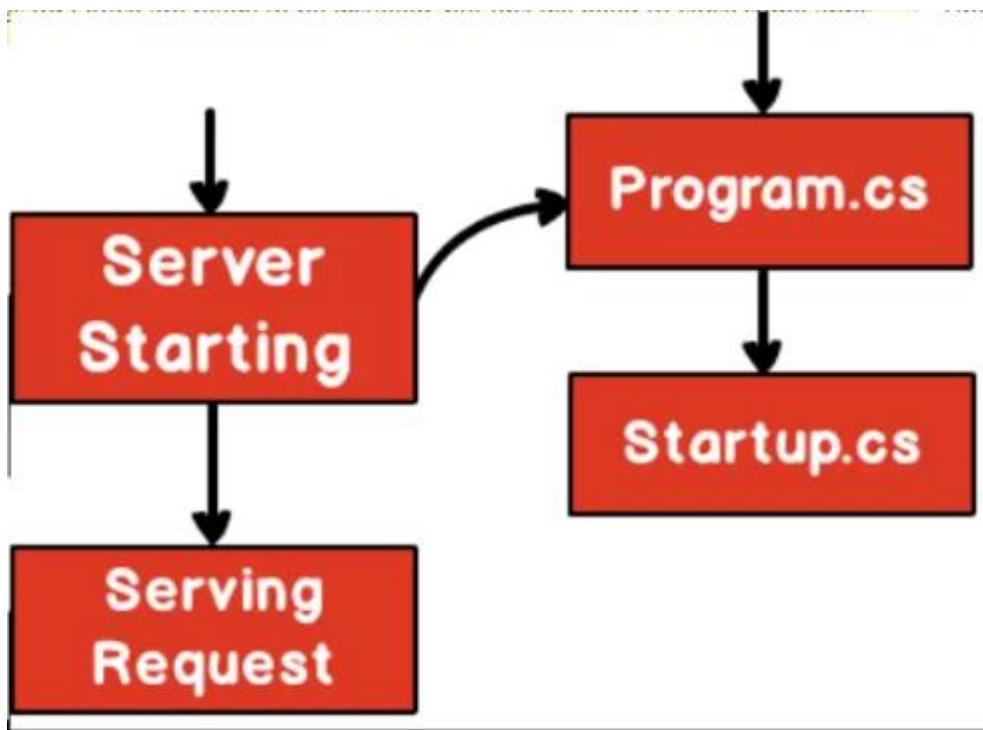
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Builder;
6  using Microsoft.AspNetCore.Hosting;
7  using Microsoft.Extensions.Configuration;
8  using Microsoft.Extensions.DependencyInjection;
9
10 namespace CustomerApplication
11 {
12     public class Startup
13     {
```

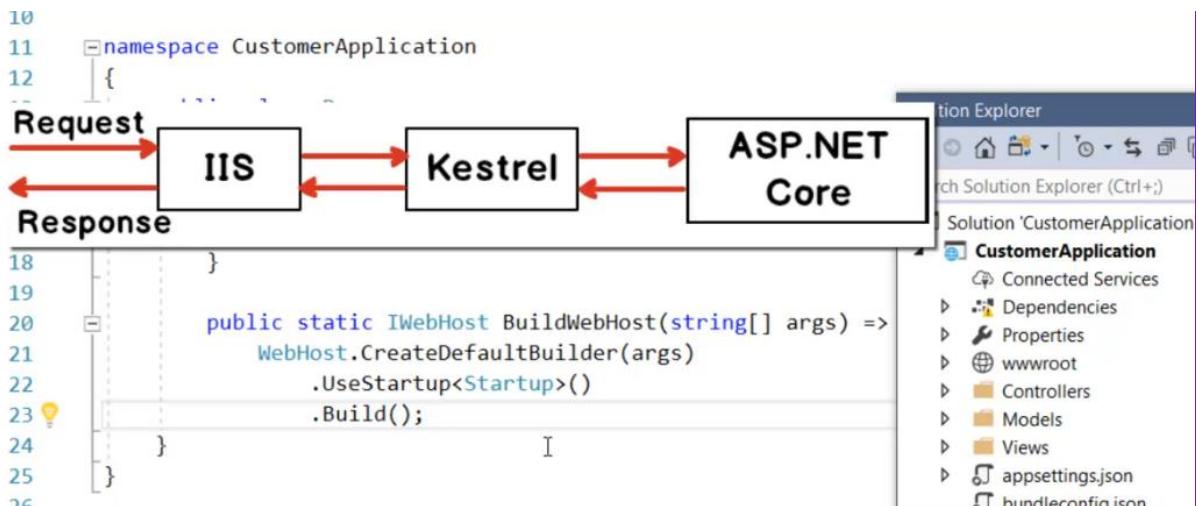
Controller - This will have binding code.
Model :- This will have classes of model.
Views :- This has the HTML code.
WWWRoot :- This has static content.
Startup :- Loads the initial configurations.
appsettings.json :- Stores Parameters and configuration.
program.cs :- invokes the webserver.

Search Solution Explorer (Ctrl+Shift+F)

Solution 'CustomerApplication' (1 project)

- CustomerApplication
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Controllers
 - Models
 - Views
 - appsettings.json
 - bower.json
 - bundleconfig.json
- Program.cs
- Startup.cs





The screenshot shows the Microsoft Visual Studio interface with the Startup.cs file open. The code defines a Startup class that implements the IWebHostBuilder interface. It uses dependency injection to get IConfiguration and sets up services like AddMvc.

```

10     namespace CustomerApplication
11     {
12         public class Startup
13         {
14             public Startup(IConfiguration configuration)
15             {
16                 Configuration = configuration;
17             }
18
19             public IConfiguration Configuration { get; }
20
21             // This method gets called by the runtime. Use this method to add services to
22             // the container.
23             public void ConfigureServices(IServiceCollection services)
24             {
25                 services.AddMvc();
26             }
27         }
28     }

```

File menu:

- File
- Edit
- View
- Project
- Build
- Debug
- Team
- Tools
- Test
- Analyze
- Window
- Help

Toolbars:

- Quick Launch
- Server Explorer
- Toolbox

Code editor:

```

10     namespace CustomerApplication
11     {
12         public class Startup
13         {
14             public Startup(IConfiguration configuration)
15             {
16                 Configuration = configuration;
17             }
18
19             public IConfiguration Configuration { get; }
20
21             // This method gets called by the runtime. Use this method to add services to
22             // the container.
23             public void ConfigureServices(IServiceCollection services)
24             {
25                 services.AddMvc();
26             }
27         }
28     }

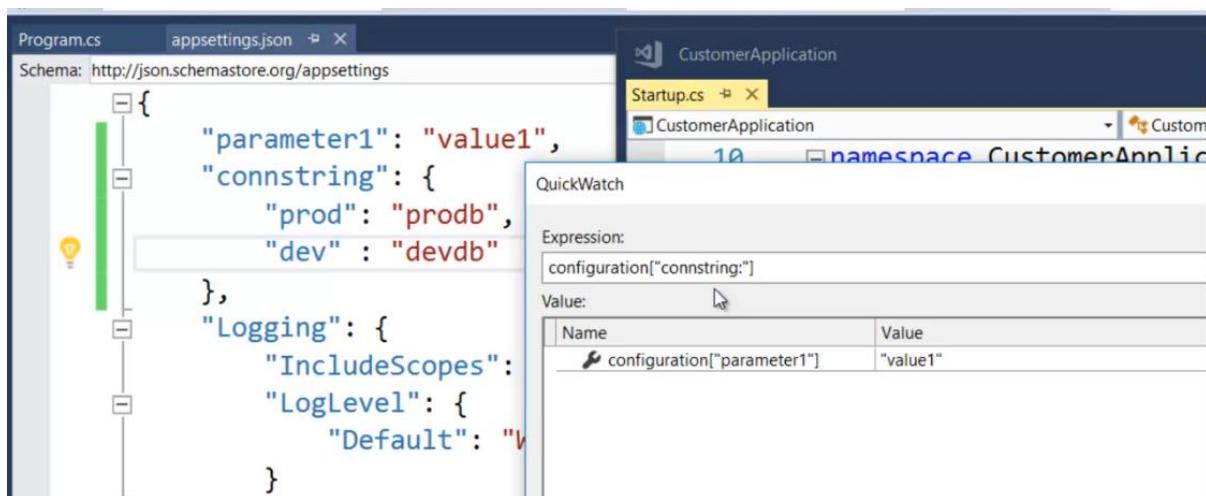
```

This configuration using dependency injection.



```
Startup.cs Program.cs appsettings.json ✘ X
Schema: http://json.schemastore.org/appsettings

{
    "parameter1": "value1",
    "connstring": {
        "prod": "proddb",
        "dev": "devdb"
    },
    "Logging": {
        "IncludeScopes": false,
        "LogLevel": {
            "Default": "Warning"
        }
    }
}
```



```
Program.cs appsettings.json ✘ X
Schema: http://json.schemastore.org/appsettings

{
    "parameter1": "value1",
    "connstring": {
        "prod": "proddb",
        "dev": "devdb"
    },
    "Logging": {
        "IncludeScopes": false,
        "LogLevel": {
            "Default": "Warning"
        }
    }
}
```

CustomerApplication

Startup.cs ✘ X

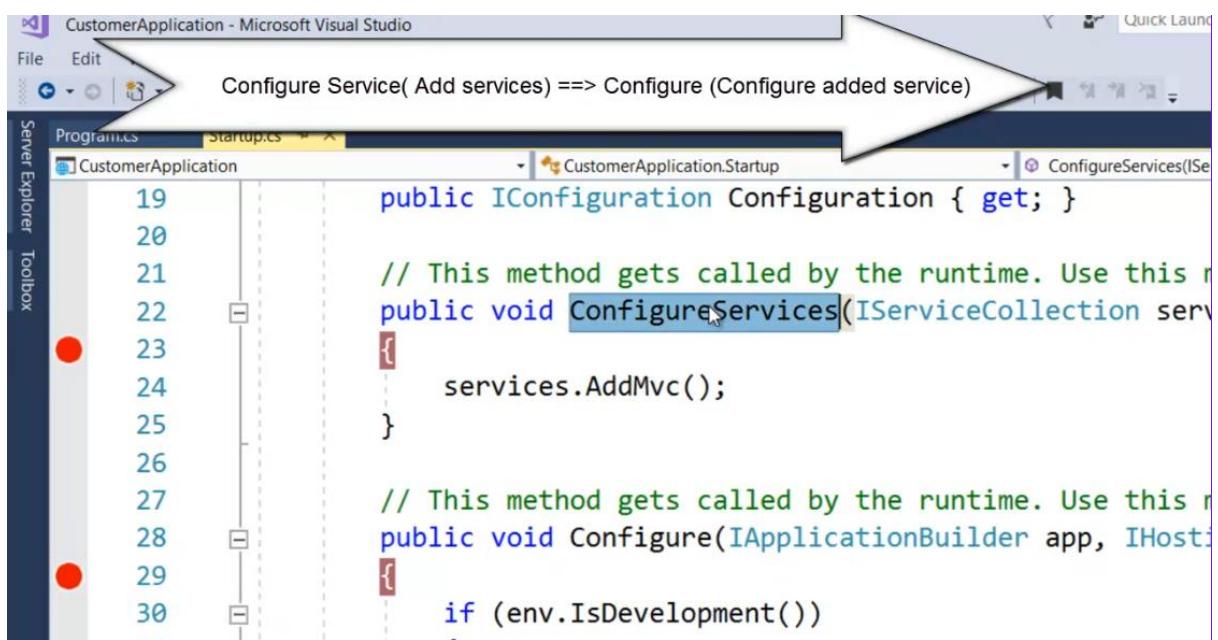
CustomerApplication

QuickWatch

Expression: configuration["connstring"]

Value:

Name	Value
configuration["parameter1"]	"value1"

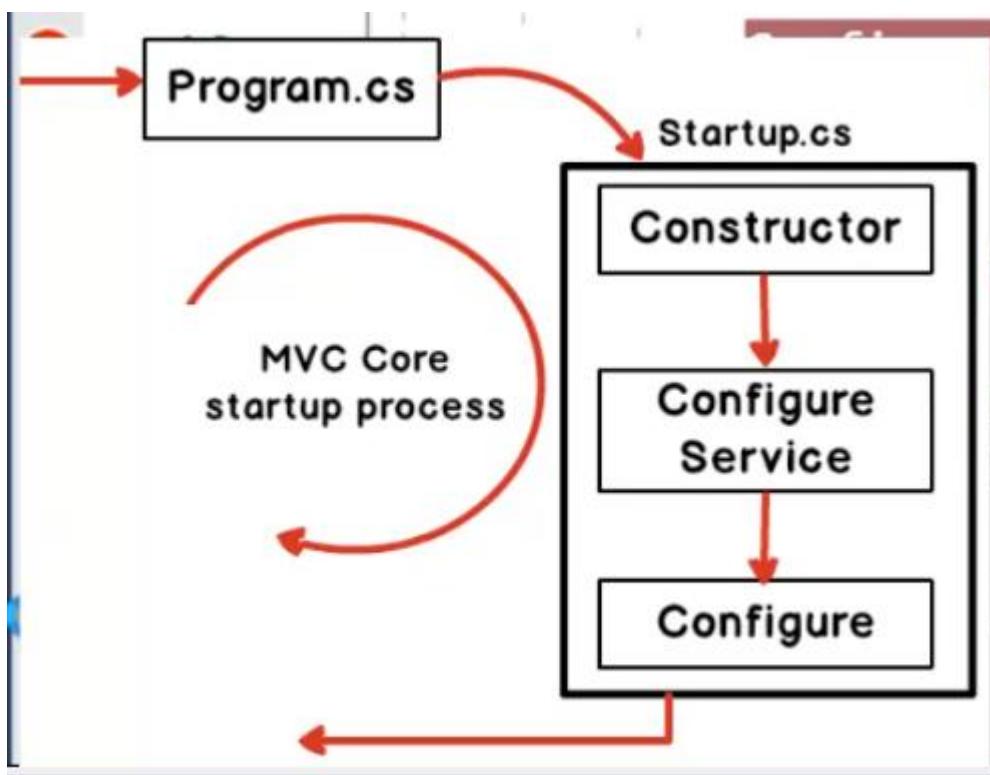


```
CustomerApplication - Microsoft Visual Studio
File Edit
Configure Service( Add services) ==> Configure (Configure added service)

Programs Startup.cs ✘ X
CustomerApplication Startup ConfigureServices(IServiceCollection services)
CustomerApplication.Startup Configure(IApplicationBuilder app, IWebHostEnvironment env)
CustomerApplication.Configure(IApplicationBuilder app, IWebHostEnvironment env)

19     public IConfiguration Configuration { get; }
20
21
22     public void ConfigureServices(IServiceCollection services)
23     {
24         services.AddMvc();
25     }
26
27     public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
28     {
29         if (env.IsDevelopment())
30             app.UseDeveloperExceptionPage();
31         else
32             app.UseExceptionHandler("/Home/Error");
33
34         app.UseStaticFiles();
35
36         app.UseMvc(routes =>
37         {
38             routes.MapRoute(
39                 name: "default",
40                 pattern: "{controller=Home}/{action=Index}/{id?}");
41         });
42     }
43 }
```

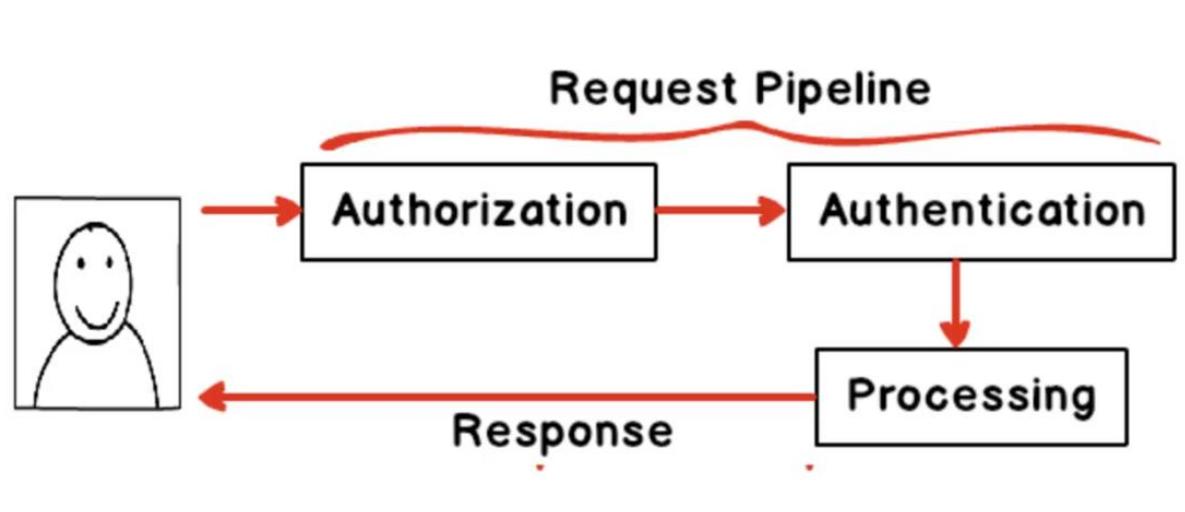
These two method are used to see , which service you want to use and how you want to configure them.



MVC they have made plug and play model



Context = Request+Response



Startup.cs

```

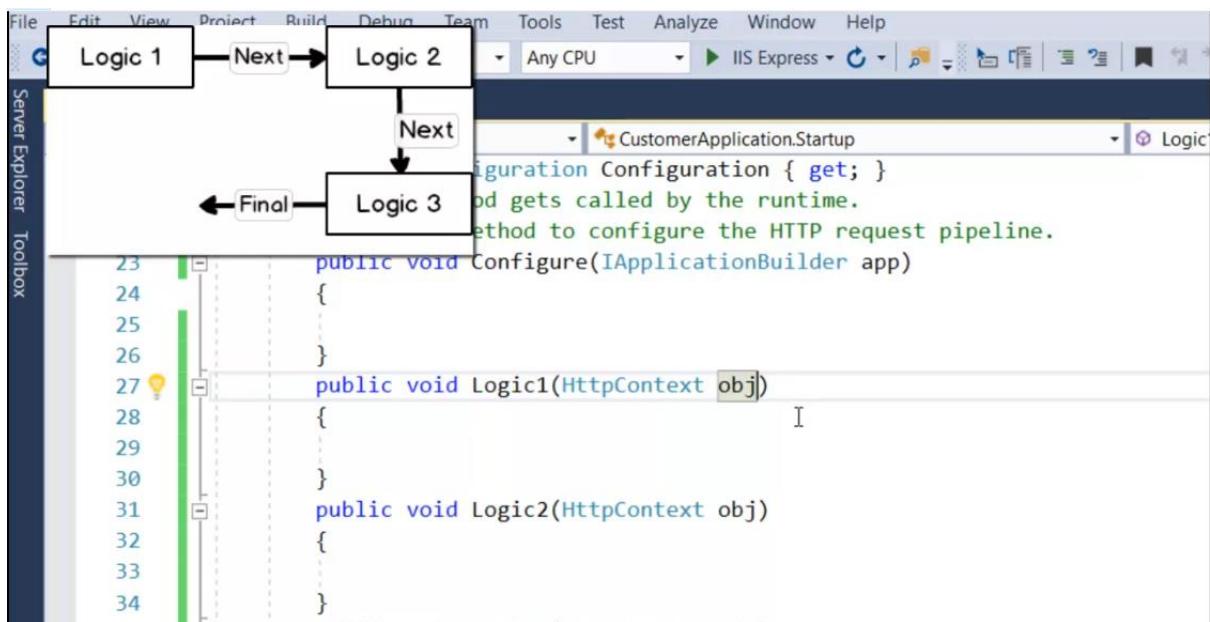
22     public void ConfigureServices(IServiceCollection services)
23     {
24         services.AddMvc();
25     }
26
27     // This method gets called by the runtime. Use this method to
28     public void Configure(IApplicationBuilder app, IHostingEnvironment env)
29     {
30         if (env.IsDevelopment())
31         {
32             app.UseDeveloperExceptionPage();
33             app.UseBrowserLink();
34         }
35         else
36         {
37             app.UseExceptionHandler("/Home/Error");
        }
    }
  
```

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Startup.cs

```

13     public Startup(IConfiguration configuration)
14     {
15         Configuration = configuration;
16     }
17
18     public IConfiguration Configuration { get; }
19
20
21
22
23     // This method gets called by the runtime.
24     // Use this method to configure the HTTP request pipeline.
25     public void Configure(IApplicationBuilder app)
26     {
27
        }
  
```



The screenshot shows the `Startup.cs` file in the Visual Studio editor. The code implements the `IApplicationBuilder` interface to define the request pipeline:

```

26     }
27     public async Task Logic1(HttpContext obj,
28                             Func<Task> next)
29     {
30         await obj.Response.WriteAsync("Logic 1\n");
31         await next.Invoke();
32     }
33     public async Task Logic2(HttpContext obj, Func<Task> next)
34     {
35         await obj.Response.WriteAsync("Logic 2\n");
36         await next.Invoke();
37     }
38     public async Task Logic3(HttpContext obj)
39     {
40         await obj.Response.WriteAsync("Logic 3\n");
41     }

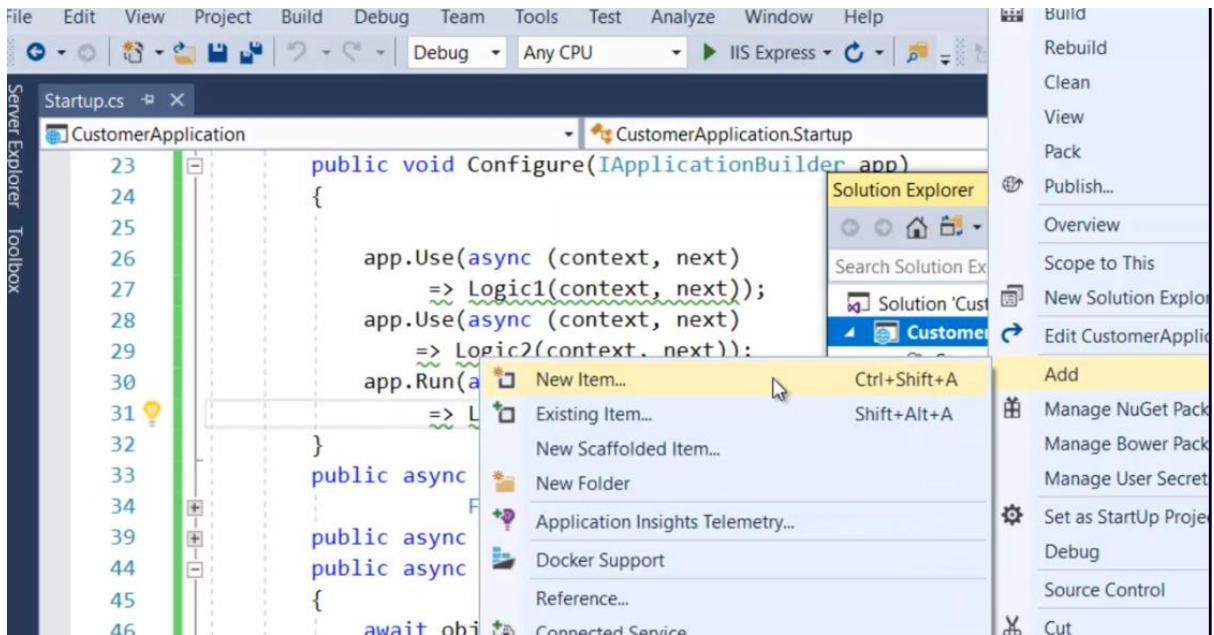
```

```
File Edit View Project Build Debug Team Tools Test Analyze Window Help
CustomerApplication.Startup
Startup.cs
17     Configuration = configuration;
18 }
19
20     public IConfiguration Configuration { get; }
21     // This method gets called by the runtime.
22     // Use this method to configure the HTTP request pipeline.
23     public void Configure(IApplicationBuilder app)
24     {
25         app.Use();
26     }
27     public async Task Logic1(HttpContext obj,
28         Func<Task> next)...
29     public async Task Logic2(HttpContext obj, Func<Task> next)...
30     public async Task Logic3(HttpContext obj)...
31 }
32 }
33 }
34 }
```

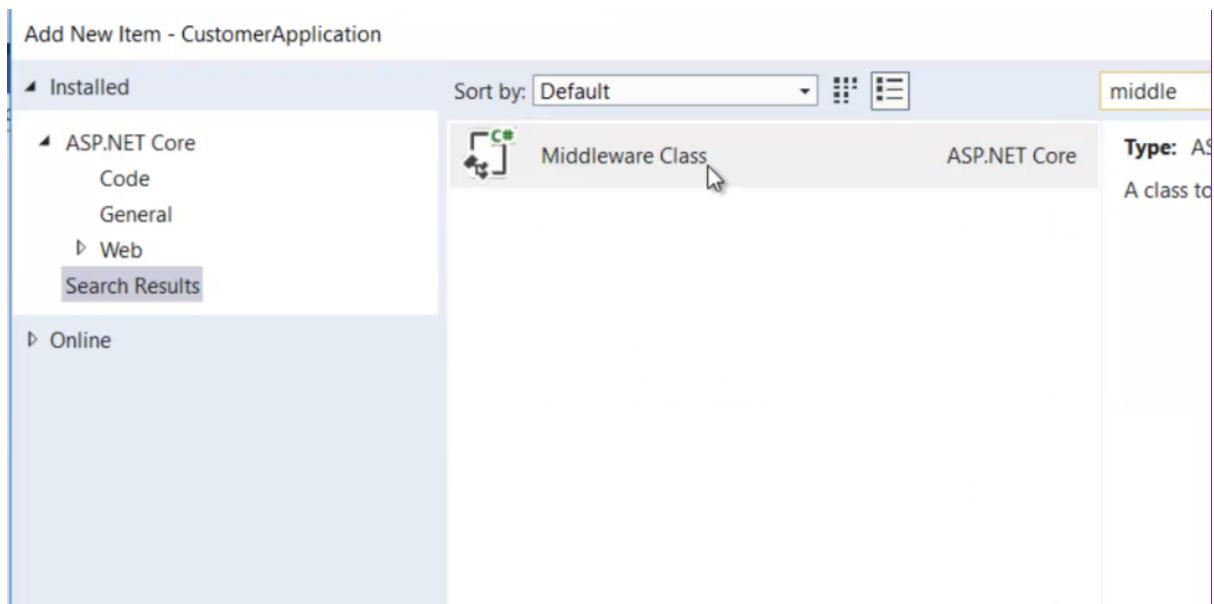
This app has two method USE and RUN

```
File Edit View Project Build Debug Team Tools Test Analyze Window Help
CustomerApplication.Startup
Startup.cs
19     Configuration = configuration;
20
21     public IConfiguration Configuration { get; }
22     // This method gets called by the runtime.
23     // Use this method to configure the HTTP request pipeline.
24     public void Configure(IApplicationBuilder app)
25     {
26         app.Use();
27     }
28     public async Task Logic2(HttpContext obj, Func<Task> next)...
29     public async Task Logic3(HttpContext obj)...
30 }
```

They are the middle ware



In order to add middle ware add class you can template also



Add three two Middleware file

The screenshot shows the Visual Studio IDE interface with the following details:

- File Menu:** File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help.
- Toolbox:** Server Explorer, Toolbox.
- Toolbar:** Standard icons for file operations, search, and navigation.
- Project Explorer:** CustomerApplication.
- Code Editor:** The active file is `MiddlewareLogic1.cs`. The code defines a middleware logic class:

```
// You may need to install the Microsoft.AspNetCore.Http.Abstractions package to reference RequestDelegate
public class MiddlewareLogic1
{
    private readonly RequestDelegate _next;

    public MiddlewareLogic1(RequestDelegate next)
    {
        _next = next;
    }

    public Task Invoke(HttpContext httpContext)
    {
        return _next(httpContext);
    }
}
```

The screenshot shows the Visual Studio IDE interface with the following details:

- File Menu:** File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help.
- Toolbox:** Server Explorer, Toolbox.
- Toolbar:** Standard icons for file operations, search, and navigation.
- Project Explorer:** CustomerApplication.
- Code Editor:** The active file is `MiddlewareLogic1.cs`. The code defines a middleware logic class with a break point set on the line `await _next(HttpContext);`:

```
private readonly RequestDelegate _next;

public MiddlewareLogic1(RequestDelegate next)
{
    _next = next;
}

public async Task Invoke(HttpContext httpContext)
{
    await httpContext.Response.WriteAsync("This is logic1 \n");
    await _next(httpContext);
}

// Extension method used to add the middleware to the HTTP pipeline

```

Same with logic2

Screenshot of Visual Studio showing the code for `MiddlewareLogic2.cs`. The code defines a middleware component that writes "This is logic1 \n" to the response and then calls the next middleware in the chain.

```
16     _next = next;
17 }
18 }
19
20 public async Task Invoke(HttpContext httpContext)
21 {
22     await httpContext.Response.WriteAsync("This is logic1 \n");
23     await _next(httpContext);
24 }
25
26 }
27
28 // Extension method used to add the middleware to the HTTP request pipeline
29 public static class MiddlewareLogic2Extensions
30 {
31 }
```

I have to use logic below in place of following

Screenshot of Visual Studio showing the code for `Startup.cs`. The `Configure` method uses the `app.Use` extension method to add middleware logic to the application builder.

```
23     app.Use(async (context, next)
24     {
25         await Logic1(context, next);
26         await Logic2(context, next);
27         await Logic3(context);
28     });
29 }
```

The highlighted code block contains the logic:

```
public void Configure(IApplicationBuilder app)
{
    app.Use(async (context, next)
        => Logic1(context, next));
    app.Use(async (context, next)
        => Logic2(context, next));
    app.Run(async (context)
        => Logic3(context));
}
```

Screenshot of Visual Studio showing the code editor with the file `Startup.cs` open. The cursor is at line 26, column 10, where the code `app.usem` is being typed. A tooltip shows the available completions for `usem`, which includes `UseMiddleware<>` (highlighted in blue), along with other options like `UseHttpMethodOverride`, `UseMicrosoftAccountAuthentication`, and `UseMiddleware`. The tooltip also provides a brief description: "(extension) IApplicationBuilder IApplicationBuilder[] args) (+ 1 generic overload) Adds a middleware type to the application's request pipeline." The code in the editor is as follows:

```
23 public void Configure(IApplicationBuilder app)
24 {
25     app.usem
26 }
27 public
28 public
29 public
30 public
31 {
32     awa
33 }
34
35
36
37
38
39
40
41
42
43
44 }
```

Screenshot of Visual Studio showing the code editor with the file `Startup.cs` open. The cursor is at line 28, column 10, where the code `app.Run()` is being typed. A tooltip shows the available completions for `Run`, which includes `(extension) void IApplicationBuilder.Run(RequestDelegate handler)` (highlighted in blue), along with a description: "Adds a terminal middleware delegate to the application's request pipeline. handler: A delegate that handles the request." The code in the editor is as follows:

```
23 public void Configure(IApplicationBuilder app)
24 {
25
26     app.UseMiddleware<MiddlewareLogic1>();
27     app.UseMiddleware<MiddlewareLogic2>();
28     app.Run()
29 }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44 }
```

```
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express Config
MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs CustomerApplication.Startup
CustomerApplication
23     public void Configure(IApplicationBuilder app)
24     {
25
26         app.UseMiddleware<MiddlewareLogic1>();
27         app.UseMiddleware<MiddlewareLogic2>();
28         app.Run(async context =>
29             Logic3(context));
30     }
31     public async Task Logic1(HttpContext obj,
32                             Func<Task> next)...
33     public async Task Logic2(HttpContext obj, Func<Task> next)...
34     public async Task Logic3(HttpContext obj)
35     {
36         await obj.Response.WriteAsync("Logic 3\n");
37     }
38
39
40
41
42
43
44
```

The middle ware is nothing , the logic , that you want put in pipe line.

```
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express Config
MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs CustomerApplication.Startup
CustomerApplication
23     public void Configure(IApplicationBuilder app)
24     {
25
26         app.UseMiddlewareLogic1();
27         app.UseMiddlewareLogic2();
28         app.Run(async context =>
29             Logic3(context));
30     }
31     public async Task Logic1(HttpContext obj,
32                             Func<Task> next)...
33     public async Task Logic2(HttpContext obj, Func<Task> next)...
34     public async Task Logic3(HttpContext obj)
35     {
36         await obj.Response.WriteAsync("Logic 3\n");
37     }
38
39
40
41
42
43
44
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Startup.cs

```
CustomerApplication
```

```
32 app.UseMiddlewareLogic1();
33 app.UseMiddlewareLogic2();
34 app.Run(async context =>
35     Logic3(context));
36 }
37 public void Execute(IApplicationBuilder app)
38 {
39     app.Run(async context =>
40         LogicHome(context));
41 }
42 public async Task LogicSearch(HttpContext obj)
43 {
```

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Startup.cs

```
CustomerApplication
```

```
23 public void Configure(IApplicationBuilder app)
24 {
25     // /Home
26     app.Map("/Home", () =>
27         app.UseMiddlewareLogic1();
28         app.UseMiddlewareLogic2();
29         app.Run(async context =>
30             Logic3(context));
31     );
32 }
```

Default → Logic 1 → Logic 2

/ Home → Logic 3 → Logic 4

? search = Test → Logic 5 → Logic 6

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Startup.cs

```
CustomerApplication
```

```
32 Logics(context);
33 }
34 public void Execute(IApplicationBuilder app)
35 {
36     app.Run(async context =>
37         Logic3(context));
38 }
39 public async Task LogicHome(HttpContext obj)
40 {
41     await obj.Response.WriteAsync("Logic");
42 }
```

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs

```
CustomerApplication
29
30
31
32
33
34
35
36
37
38
39
40
```

CustomerApplication.Startup

```
    app.UseMiddlewareLogic1();
    app.UseMiddlewareLogic2();
    app.Run(async context =>
        Logic3(context));
}

public void Execute(IApplicationBuilder app)
{
    app.Run(async context =>
        LogicHome(context));
}

public async Task LogicHome(HttpContext context)
```

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs

```
CustomerApplication
20
21
22
23
24
25
26
27
28
29
30
31
32
```

CustomerApplication.Startup

```
public IHostConfiguration Configuration { get; }

// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app)
{
    // /Home
    app.Map("/Home", Execute);
    // ?search=test

    app.UseMiddlewareLogic1();
    app.UseMiddlewareLogic2();
    app.Run(async context =>
        Logic3(context));
```

Privacy.cshtml.cs Index.cshtml Error.cshtml.cs launchSettings.json Startup.cs WebApplicationCore

```
WebApplicationCore
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

WebApplicationCore.Startup

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    // /Home
    app.Map("/Home", Execute);

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        // The default HSTS value is 30 days. You may want to change this for production scenarios,
        app.UseHsts();
    }
}
```

So, if you want to build your pipe line, using the URL, using the query string, then you need to use map or map when.

The screenshot shows the Microsoft Visual Studio IDE with the Startup.cs file open. The code defines a middleware pipeline:

```
public Iconfiguration Configuration { get; }
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app)
{
    // /Home
    app.Map("/Home", Execute);
    // ?search=test

    app.UseMiddlewareLogic1();
    app.UseMiddlewareLogic2();
    app.Run(async context =>
        Logic3(context));
}
```

The browser window shows the output for the URL `localhost:53044`, which displays the text "This is logic1", "This is logic2", and "Logic 3".

The screenshot shows the Microsoft Visual Studio IDE with the Startup.cs file open. The code defines a middleware pipeline:

```
public Iconfiguration Configuration { get; }
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app)
{
    // /Home
    app.Map("/Home", Execute);
    // ?search=test

    app.UseMiddlewareLogic1();
    app.UseMiddlewareLogic2();
    app.Run(async context =>
        Logic3(context));
}
```

The browser window shows the output for the URL `localhost:53044/Home`, which displays the text "Logic Home".

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    //Home
    app.Map("/Home", Execute);
    //?search=test
    app.MapWhen(context =>
    {
        return context.Request.Query.ContainsKey("search");
    }, ExecuteSearch);
}
```

```
private void Execute(IApplicationBuilder app)
{
    app.Run(async context => LogicHome(context));
}

private async Task LogicHome(HttpContext obj)
{
    await obj.Response.WriteAsync("Logic Home");
}

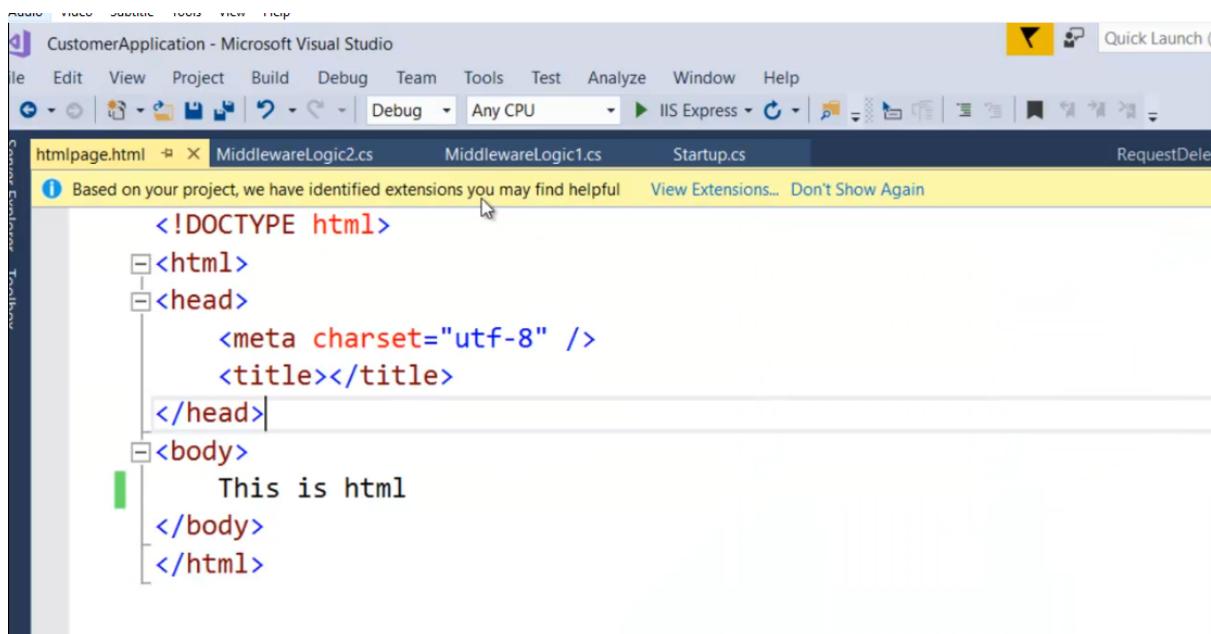
private void ExecuteSearch(IApplicationBuilder app)
{
    app.Run(async context => LogicSearch(context));
}

private async Task LogicSearch(HttpContext obj)
{
    await obj.Response.WriteAsync("Logic Search");
}
```

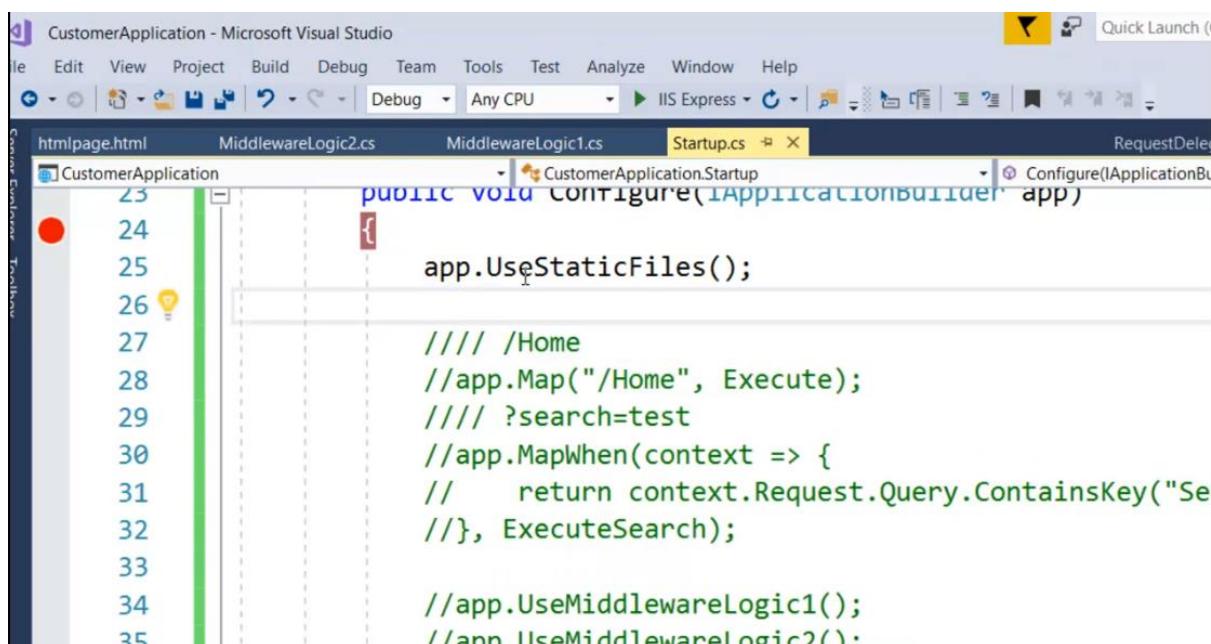
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    This is html
</body>
</html>
```

Solution Explorer

- CustomerApplication (1 project)
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - css
 - images
 - js
 - lib
 - favicon.ico
 - htmlpage.html



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    This is html
</body>
</html>
```



```
public void Configure(IApplicationBuilder app)
{
    app.UseStaticFiles();

    //// /Home
    //app.Map("/Home", Execute);
    //// ?search=test
    //app.MapWhen(context => {
    //    return context.Request.Query.ContainsKey("Se
    //}, ExecuteSearch);

    //app.UseMiddlewareLogic1();
    //app.UseMiddlewareLogic2();
```

Note : Static file should be wwwroot folder

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

Startup.cs

```
public void Configure(IApplicationBuilder app)
{
    app.UseStaticFiles();
    //app.Map("/Home");
    //app.Map("/?search=test");
    //app.MapWhen(context => {
    //    return context.Request.Query.ContainsKey("Search");
    //}, ExecuteSearch);
}
```

CustomerApplication

htmlpage.html MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs RequestDelete

localhost:53044/

This is html

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

Startup.cs

```
public IConfiguration Configuration { get; }
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app)
{
    app.UseStaticFiles();
    app.UseMvc();
    //app.Map("/Home");
    //app.Map("/?search=test");
    //app.MapWhen(context => {
    //    return context.Request.Query.ContainsKey("Search");
    //}, ExecuteSearch);
}
```

CustomerApplication

htmlpage.html MiddlewareLogic2.cs MiddlewareLogic1.cs Startup.cs RequestDelete

localhost:53044/

Lets test what you have learnt.

Define context , request and response ?

Explain terms request pipeline and middleware?

In MVC core where do we configure request pipeline?

What is the importance of "IApplicationBuilder"?

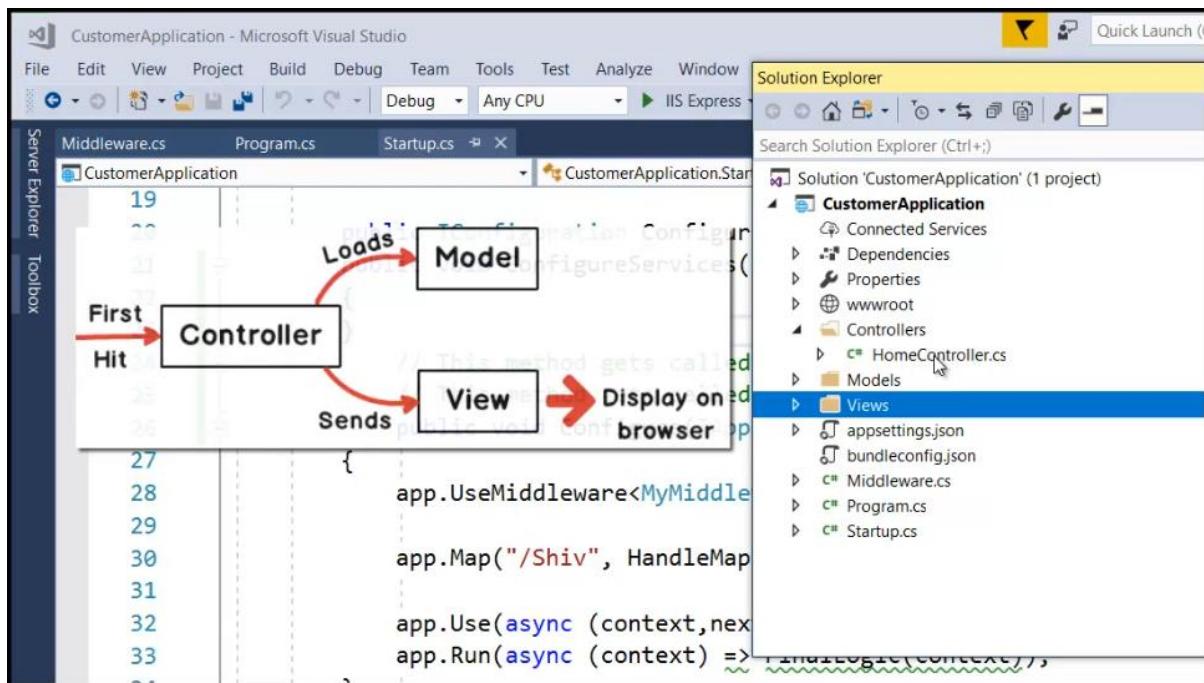
Explain "use" , "map" and "run" ?

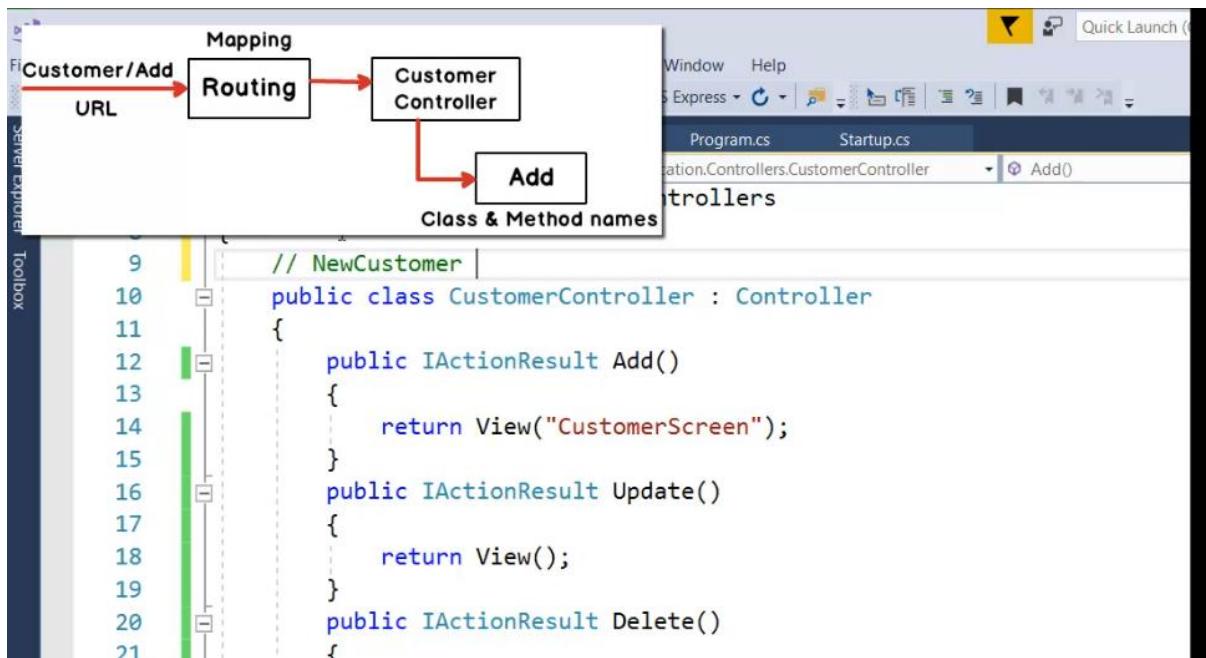
How can you create middleware using classes?

Differentiate between "map" and "mapwhen" ?

Implementing basic Model, View & Controller

Create Controllers and Views
Understanding ActionResults in MVC
What is Scaffolding ?
MVC Core namespaces
Layout , Routing and Razor pages





The screenshot shows the `Startup.cs` file in Microsoft Visual Studio:

```

22     services.AddMvc();
23 }
24 // This method gets called by the runtime. Use this method
25 // This method gets called by the runtime. Use this method
26 public void Configure(IApplicationBuilder app)
27 {
28     app.UseMvc();
29     app.UseMvcWithDefaultRoute();
30 }
31 private static void HandleMapTest(IApplicationBuilder app)
32 {
33     app.Run(async context =>
34     {
35     });
36 }

```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Output CustomerScreen.cshtml CustomerController.cs X Middleware.cs Program.cs Startup.cs

CustomerApplication

```
7  namespace CustomerApplication.Controllers
8  {
9      // NewCustomer
10     // http://localhost/Customer/Add
11     // http://localhost/Supplier/Add
12     public class CustomerController : Controller
13     {
14         [Bind(Exclude = "")]
15         public IActionResult Index()
16         {
17             return View();
18         }
19     }
20 }
```

localhost

localhost:53044/Customer/New

localhost:53044/Customer/Add

localhost:53044/Customer

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Solution Explorer

CustomerScreen.cshtml _ViewStart.cshtml _ViewImports.cshtml

```
@using CustomerApplication
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Solution 'CustomerApplication' (1 project)

- CustomerApplication
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - Controllers
 - CustomerController.cs
 - Models
 - Views
 - Customer
 - CustomerScreen.cshtml
 - Shared
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - appsettings.json
 - appsettings.Development.json
 - bundleconfig.json
 - Middleware.cs

The diagram illustrates a common namespace structure. At the top, a box labeled '(Common namespace)' contains the text '_View Imports'. Three arrows point from this box down to three separate boxes labeled 'View 1', 'View 2', and 'View 3'. To the right of 'View 1' is the text 'Apply to all'.

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

CustomerScreen.cshtml _ViewStart.cshtml _ViewImports.cshtml Startup.cs

CustomerApplication

```
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6
7 namespace CustomerApplication.Controllers
8 {
9     // NewCustomer
10    // http://localhost/Customer/Add
11    // http://localhost/Supplier/Add
12    public class CustomerController : Controller
13    {
14        public IActionResult Add()
15        {
16            return View("CustomerScreen");
17        }
18        public IActionResult Update()
19    }
}
```

Note : If you add any namespace In the view imports , then it will be access by across view.

The diagram is identical to the one above, showing a common namespace structure with '_View Imports' at the top, pointing to 'View 1', 'View 2', and 'View 3', with 'Apply to all' to the right of 'View 1'.

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

CustomerScreen.cshtml _ViewStart.cshtml _ViewImports.cshtml Startup.cs

CustomerApplication

```
@using CustomerApplication.Controllers
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

CustomerScreen.cshtml* _ViewStart.cshtml _ViewImports.cshtml Startup.cs

Server Explorer Toolbox

```
Layout = null;
int i = 0; ICustomerController
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>CustomerScreen</title>
</head>
```

(Common namespace)

```
graph TD
    ViewImports["_View Imports"] --> View1[View 1]
    ViewImports --> View2[View 2]
    ViewImports --> View3[View 3]
    ViewImports -- "Apply to all" --> View1
    ViewImports -- "Apply to all" --> View2
    ViewImports -- "Apply to all" --> View3
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

CustomerScreen.cshtml* _ViewStart.cshtml _ViewImports.cshtml

Server Explorer Toolbox

```
Layout = "_Layout";
```

(Common Layout)

```
graph TD
    ViewStart["_View Start"] --> View1[View 1]
    ViewStart --> View2[View 2]
    ViewStart --> View3[View 3]
    ViewStart -- "Apply to all" --> View1
    ViewStart -- "Apply to all" --> View2
    ViewStart -- "Apply to all" --> View3
```

Solution Explorer

- Properties
- wwwroot
- Controllers
- CustomerController.cs
- Models
- Views
- Customer
- CustomerScreen.cshtml
- Shared
- _Layout.cshtml
- _ValidationScriptsPartial.cshtml
- Error.cshtml
- _ViewImports.cshtml
- _ViewStart.cshtml
- appsettings.json

If no layout will be apply on , so by default layout will apply.

```

graph TD
    CL[Common Layout] --> V1[View 1]
    CL --> V2[View 2]
    CL --> V3[View 3]
    VS[View Start] --> ATA[Apply to all]
    ATA --> V1
    ATA --> V2
    ATA --> V3

```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window

Server Explorer Toolbox

_Layout.cshtml CustomerScreen.cshtml* _ViewStart.cshtml

```

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>CustomerScreen</title>
</head>
<body>
    Customer Code :- <input type="text"/>
    Customer name :- <input type="text"/>

```

Views

- Customer
- Shared
- CustomerScreen.cshtml
- ValidationScriptsPartial.cshtml
- Error.cshtml
- _ViewImports.cshtml
- _ViewStart.cshtml

Properties

wwwroot

Controllers

CustomerController.cs

Models

appsettings.json

Lets revise what we have learnt.....

Which are the reserved folders for controllers , views and models ?

What is the naming conventions of MVC controller classes ?

What steps do we need to add MVC services to MVC core project ?

Explain the difference between "Services.AddMvc" and "app.UseMVC" ?

What are actions in Controller ?

What is "IActionResult" in MVC Controller ?

Explain the term scaffolding ?

What are the two different types of controllers in MVC ?

All MVC controller class inherit from _____ class.

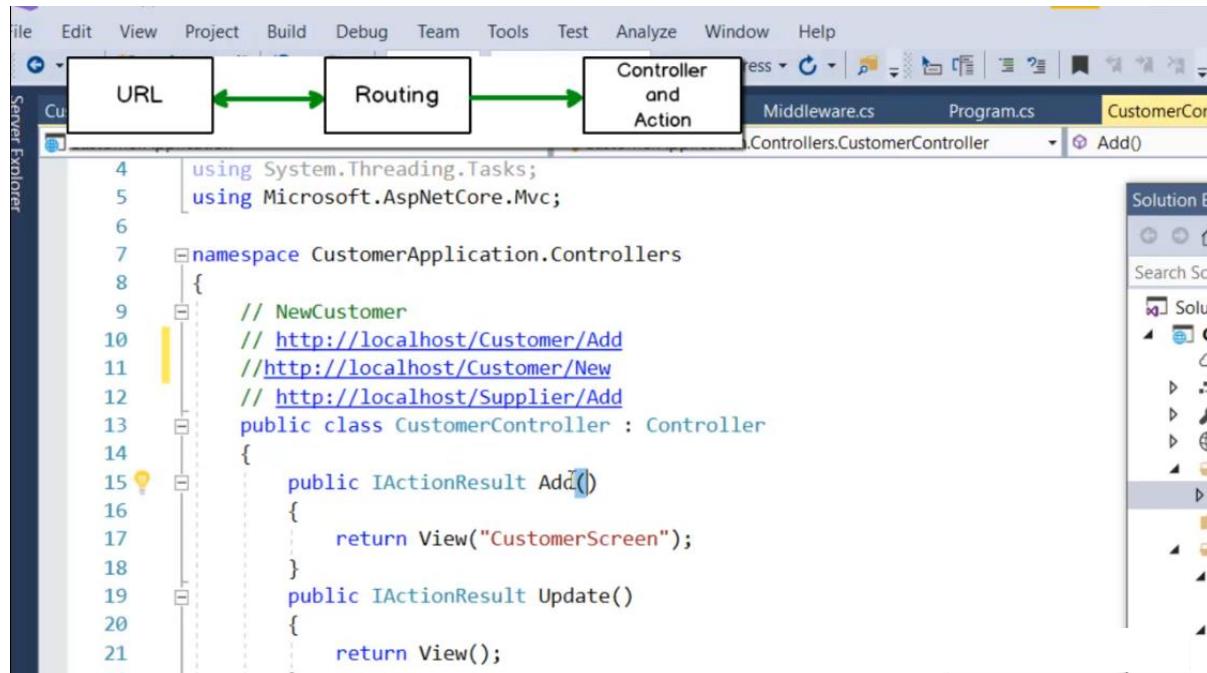
Which is the main namespace of MVC core ?

Explain layout pages ?

What is routing in MVC and how to enable default routes ?

Explain Razor page (CSHTML) ?

Routing, Conventional, Attribute and Route constraint



In default route the controller name and action name becomes your URL.

Now if we are looking for route like

The screenshot shows the Microsoft Visual Studio interface. The top menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, and Help. The toolbar below has icons for file operations, search, and navigation. The status bar indicates "CustomerApplication - Microsoft Visual Studio" and "Any CPU | IIS Express". The code editor displays the `CustomerController.cs` file:

```
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Mvc;
6
7  namespace CustomerApplication.Controllers
8  {
9      // NewCustomer
10     // http://localhost/Customer/Add
11     // http://localhost/Customer/New
12     // http://localhost/Supplier/Add
13     public class CustomerController : Controller
14     {
15         public IActionResult Add()
16         {
17             return View("CustomerScreen");
18         }
19         public IActionResult Update()
20         {
21             return View();
22         }
23     }
24 }
```

To the right of the code editor is a browser window titled "CustomerScreen" showing the URL `localhost:53044/Customer/New/1`. The page contains a text input field labeled "Customer Amount :-" and a button labeled "Add Customer".

The screenshot shows the Microsoft Visual Studio interface with the `Startup.cs` file open in the code editor. The code defines a `Configure` method:

```
22
23     services.AddMvc();
24 }
25 // This method gets called by the runtime. Use this method to add
26 // This method gets called by the runtime. Use this method to config
27 public void Configure(IApplicationBuilder app)
28 {
29     app.UseStaticFiles(); // html
30     app.UseMvc();
31     app.UseMvcWithDefaultRoute();
32     app.UseMvc(routes =>
33     {
34         routes.MapRoute(
35             name: "CustomerRoute",
36             template: "Customer/{id}/{action}/{id2?}",
37             handleMaps: HandleMapTest);
38     });
39 }
40 private static void HandleMapTest(IApplicationBuilder app)
41 {
42     app.Run(async context =>
```

A red arrow points from the bottom of the code editor area down towards the `HandleMapTest` method definition.

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

CustomerScreen.cshtml _Layout.cshtml Startup.cs ✎ Middleware.cs Program.cs CustomerController.cs appse

CustomerApplication

CustomerApplication.Startup

LoadRoutes(IRouteBuilder builder)

```
17     Configuration = configuration;
18 }
19 public void LoadRoutes(IRouteBuilder builder)
20 {
21     builder.MapRoute("route1", "Customer/Add", new
22     {
23         controller = "Customer",
24         action = "Add"
25     });
26     builder.MapRoute("route2", "Customer/New", new
27     {
28         controller = "Customer",
29         action = "Add"
30     });
31 }
32 public IConfiguration Configuration { get; }
33 public void ConfigureServices(IServiceCollection services)
34 {
35 }
```

Builder is route collection only

Solution Explorer

Search Solution E

Solution 'CustomerApplication' (CustomerApplication)

- Conn
- Depen
- Prop
- www
- Cont
- C
- Mode
- View
- apps
- bundle
- Midd
- Profil

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

CustomerScreen.cshtml _Layout.cshtml Startup.cs ✎ Middleware.cs Program.cs CustomerController.cs appse

CustomerApplication

CustomerApplication.Startup

Configure(IApplicationBuilder app)

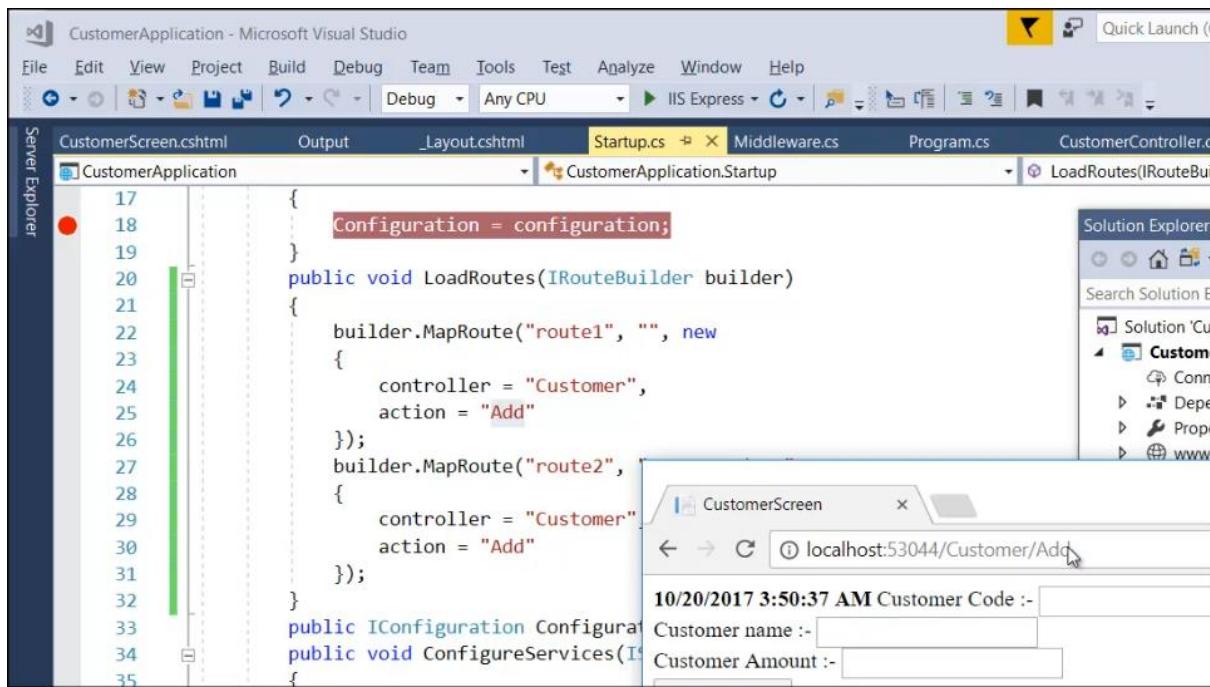
```
23     }
24 }
25 public IConfiguration Configuration { get; }
26 public void ConfigureServices(IServiceCollection services)
27 {
28     services.AddMvc();
29     // This method gets called by the runtime. Use this method to add
30     // This method gets called by the runtime. Use this method to config
31     public void Configure(IApplicationBuilder app)
32     {
33         app.UseStaticFiles(); // html
34         app.UseMvc();
35         app.UseMvcWithDefaultRoute();
36         app.UseMvc(routes =>
37         {
38             LoadRoutes(routes);
39         });
40     }
41 private static void HandleMapTest(IApplicationBuilder app)
```

Solution Explorer

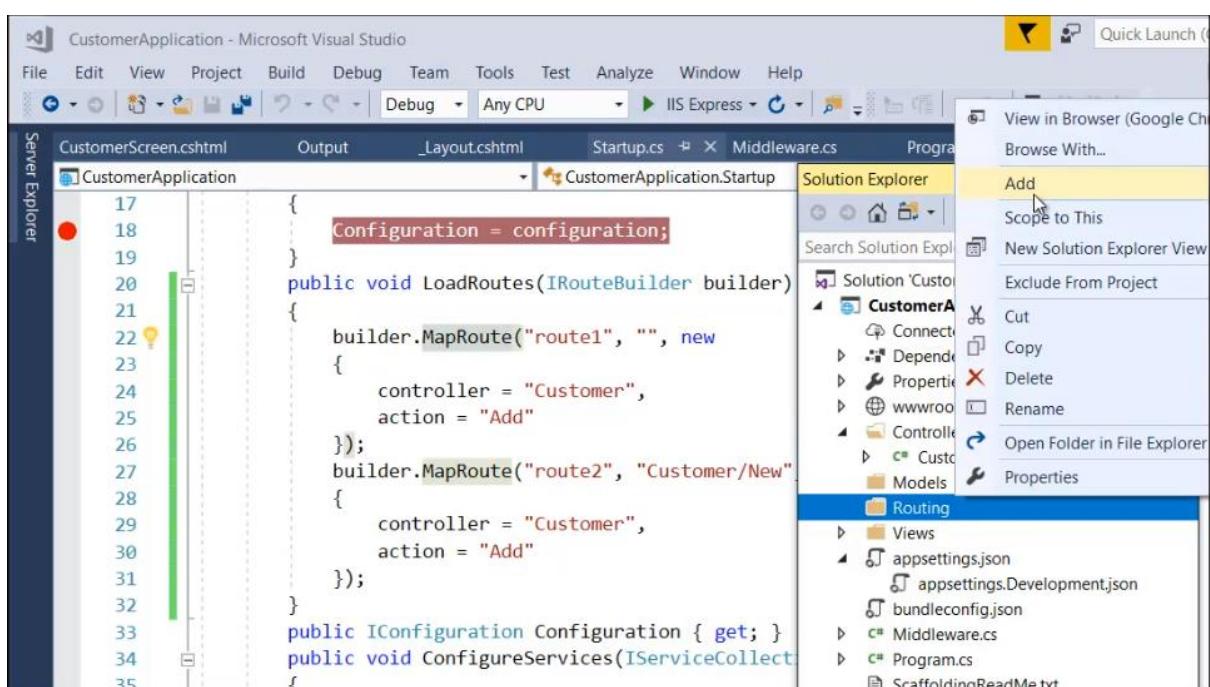
Search Solution E

Solution 'CustomerApplication' (CustomerApplication)

- Conn
- Depen
- Prop
- www
- Cont
- C
- Mode
- View
- apps
- bundle
- Midd
- Profil



Now you can also create routing in separate folder. Add Routing folder and, add static class , with any name which is not reserved.



CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

Routing.cs CustomerScreen.cshtml Output _Layout.cshtml Startup.cs Middleware.cs Program.cs

CustomerApplication CustomerApplication.Routing.ApplicationRoutes LoadRoutes(IRouteBuilder)

```
7
8     namespace CustomerApplication.Routing
9     {
10        public static class ApplicationRoutes
11        {
12            public static void LoadRoutes(IRouteBuilder builder)
13            {
14                builder.MapRoute("route1", "", new
15                {
16                    controller = "Customer",
17                    action = "Add"
18                });
19                builder.MapRoute("route2", "Customer/New", new
20                {
21                    controller = "Customer",
22                    action = "Add"
23                });
24            }
25        }
}
```

CustomerApplication - Microsoft Visual Studio

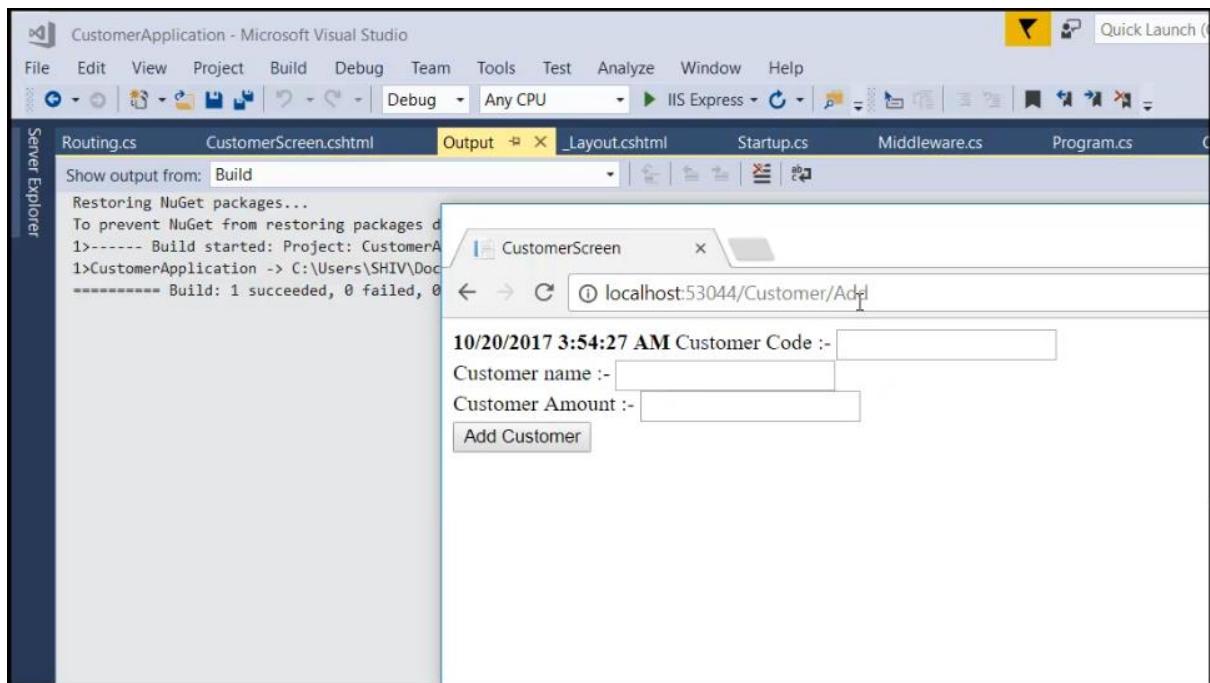
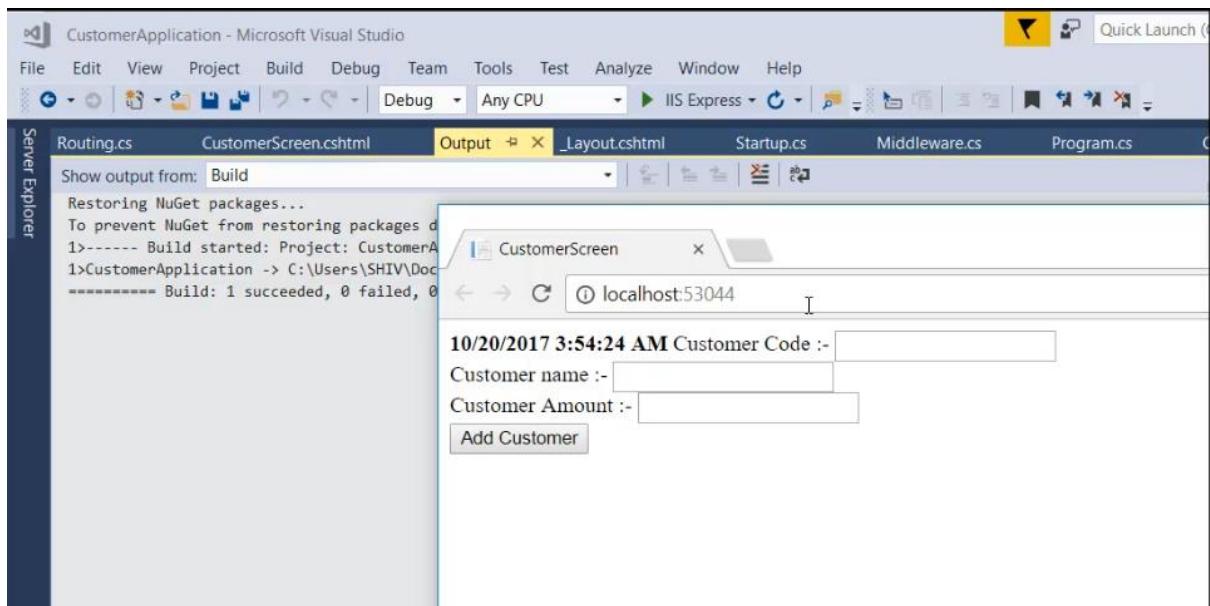
File Edit View Project Build Debug Team Tools Test Analyze Window Help

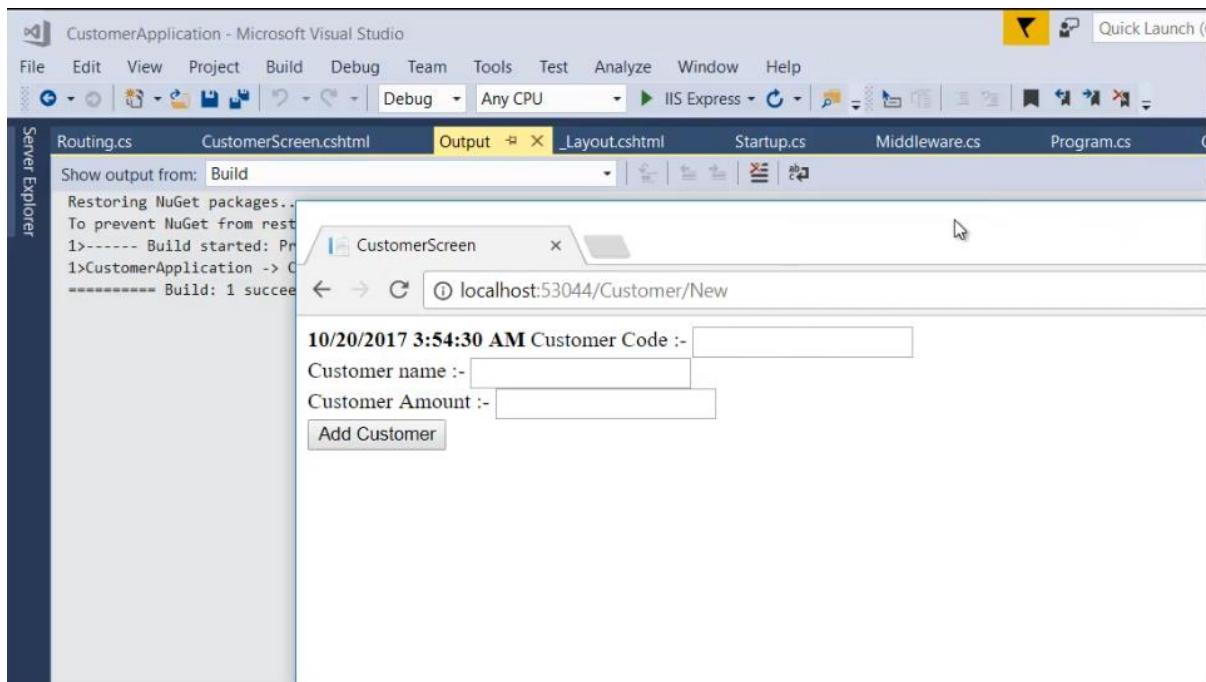
Debug Any CPU IIS Express

Routing.cs CustomerScreen.cshtml Output _Layout.cshtml Startup.cs Middleware.cs Program.cs

CustomerApplication CustomerApplication.Startup Configure(IApplicationBuilder)

```
22     public void ConfigureServices(IServiceCollection services)
23     {
24         services.AddMvc();
25     }
26     // This method gets called by the runtime. Use this method to add services to
27     // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
28     public void Configure(IApplicationBuilder app)
29     {
30         app.UseMvc(); // loads MVC
31         app.UseMvcWithDefaultRoute(); // loads the defaults
32         app.UseMvc(routes =>
33         {
34             ApplicationRoutes.LoadRoutes(routes);
35         }); // loading customer routes
36     }
37     private static void HandleMapTest(IApplicationBuilder app)
38     {
39
40         app.Run(async context =>
```





The screenshot shows the Microsoft Visual Studio interface with the title bar "CustomerApplication - Microsoft Visual Studio". The menu bar and toolbar are identical to the previous screenshot. The main area shows the code editor with the file "Routing.cs" open. The code defines a static class "ApplicationRoutes" within the namespace "CustomerApplication.Routing". It contains two route definitions: "route1" which maps to "Customer/Add" and "route2" which maps to "Customer/New".

```
8  namespace CustomerApplication.Routing
9  {
10     public static class ApplicationRoutes
11     {
12         public static void LoadRoutes(IRouteBuilder builder)
13         {
14             // conventional
15             // data base
16             builder.MapRoute("route1", "", new
17             {
18                 controller = "Customer",
19                 action = "Add"
20             });
21             builder.MapRoute("route2", "Customer/New", new
22             {
23                 controller = "Customer",
24                 action = "Add"
25             });
26         }
27     }
28 }
```

Attribute based routing

```
10 // http://localhost/
11 //http://localhost/Customer/New
12 // http://localhost/Customer/Add
13
14 [Route("Cust1")]
15 public class CustomerController : Controller
16 {
17     [Route("Add")]
18     public IActionResult Add()
19     {
20         return View("CustomerScreen");
21     }
22     public IActionResult Update()
23     {
24         return View();
25     }
26     public IActionResult Delete()
27     {
28         return View();
29     }
30 }
```

```
10 // http://localhost/
11 //http://localhost/Customer/New
12 // http://localhost/Customer/Add
13
14 [Route("Cust1")]
15 public class CustomerController : Controller
16 {
17
18     [HttpPost("Add")]
19     public IActionResult Add()    I
20     {
21         return View("CustomerScreen");
22     }
23     public IActionResult Update()
24     {
25         return View();
26     }
27     public IActionResult Delete()
28     {
29     }
30 }
```

Note: Attribute based routing precedence over the conventional based routing. And attribute based you are not supposed to provide constraint.

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express
Server Explorer Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerController
CustomerApplication
8  namespace CustomerApplication.Routing
9  {
10     public static class ApplicationRoutes
11     {
12         public static void LoadRoutes(IRouteBuilder builder)
13         {
14             // conventional
15             // data base
16             builder.MapRoute("route1", "", new
17             {
18                 controller = "Customer",
19                 action = "Add"
20             });
21             builder.MapRoute("route2", "Customer/New/{Id}", new
22             {
23                 controller = "Customer",
24                 action = "Add"
25             });
26         }
27     }
}
```

Validation over input parameter.

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express
Server Explorer Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerController
CustomerApplication
27
28
29
30
31
32     //{{ id: int}
33     //    { active: bool}
34     //    { dob: datetime}
35     //    { price: decimal}
36     //    { weight: double}
37     //    { weight: float}
38     //    { id: guid}
39     //    { ticks: long}
40     //    { username: minlength(4)}
41     //    { filename: maxlength(8)}
42     //    { filename: length(12)}
43     //    { filename: length(8, 16)}
44     //    { age: min(18)}
45     //    { age: max(120)}
46     //    { age: range(18, 120)}
47     //    { name: alpha}
48 }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Routing.cs* CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerContro

CustomerApplication

```
10     to: "ru. int"}  
11     { active: bool}  
12     { dob: datetime}  
13     { price: decimal}  
14     { weight: double} I  
15     { weight: float}  
16     { id: guid}  
17     { ticks: long}  
18     { username: minlength(4)}  
19     { filename: maxlength(8)}  
20     { filename: length(12)}  
21     { filename: length(8, 16)}  
22     { age: min(18)}  
23     { age: max(120)}  
24     { age: range(18, 120)}  
25     { name: alpha}  
26     { ssn: regex(^\\d{ 3} -\\d{ 2} -\\d{ 4} $)}  
27     { name: required}  
28 }  
29 builder.MapRoute("route1", "", new
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerContro

CustomerApplication

```
10 // http://localhost/  
11 //http://localhost/Customer/New  
12 // http://localhost/Customer/Add  
13  
14  
15 public class CustomerController : Controller  
16 {  
17  
18     public IActionResult Add(string Id)  
19     {  
20         return View("CustomerScreen");  
21     }  
22     public IActionResult Update()  
23     {  
24         return View();  
25     }  
26     public IActionResult Delete()  
27     {  
28         return View();  
29     }
```

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express
Output Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs
CustomerApplication
14 // conventional
15 // data base
16
17 builder.MapRoute("route1", "", new
18 {
19     controller = "Customer",
20     action = "Add"
21 });
22 builder.MapRoute("route2", "Customer/New/{Id:int}", new
23 {
24     controller = "Customer",
25     action = "Add"
26 });
27 }
28 }
29 }
```

OR

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU IIS Express
Output Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs
CustomerApplication
17 builder.MapRoute("route1", "", new
18 {
19     controller = "Customer",
20     action = "Add"
21 });
22 // ING1009
23 // ABC3004
24 builder.MapRoute("route2",
25     "Customer/New/{Id:regex(^[A-Z]{3,3}[0-9]{4,4}$)}", new
26 {
27     controller = "Customer",
28     action = "Add"
29 });
30 }
31 }
32 }
33 }
```

CustomerApplication (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [22140] dotnet.exe Lifecycle Events Thread: [5896] Worker Thread Stack Frame: CustomerApplication.Controllers.CustomerController

Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerController.cs

```
CustomerApplication
10 // http://localhost/
11
12 //localhost:53044/Custom
13 <--> X localhost:53044/Customer/New/INH1009
14
15
16
17
18
19
20
21
22
23
24
25
```

CustomerApplication (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [22140] dotnet.exe Lifecycle Events Thread: [5896] Worker Thread Stack Frame: CustomerApplication.Controllers.CustomerController

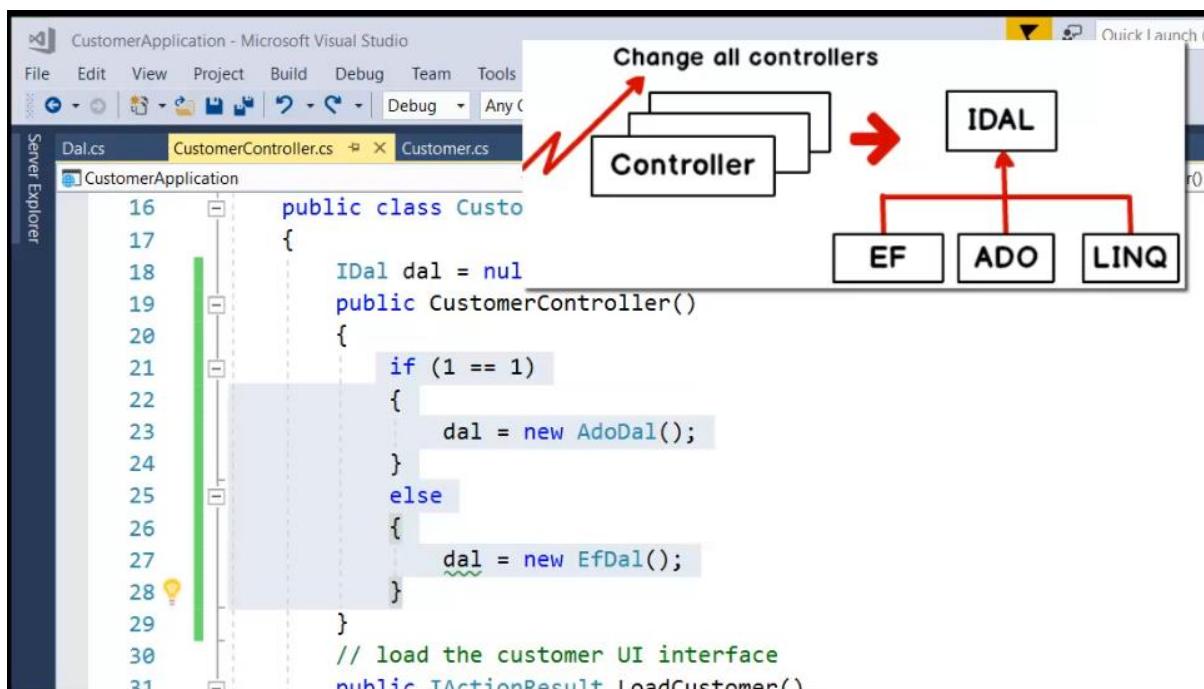
Routing.cs CustomerScreen.cshtml _Layout.cshtml Startup.cs Middleware.cs Program.cs CustomerController.cs

```
CustomerApplication
10 // http://localhost/
11 //http://localhost/Customer/New
12 // http://localhost/Customer/Add
13
14
15 public class CustomerController : Controller
16 {
17
18     public IActionResult Add(string Id)
19     {
20         if (Id == null || Id.Length < 1)
21             return View("CustomerScreen");
22     }
23     public IActionResult Update()
24     {
25         return View();
26     }
27 }
```

Syllabus completed

What is need of routing ?
Coventional based routing
Attribute based routing
Conventional vs attribute based
Route constraints

Depedency Injection (Scoped, Transient, Singleton & FromServ DI , IOC Scope, Trans, Single FromServices



Now, suppose I have to use, one more dal class, LINQDAL, in that case I have change code in entire controller. SO we looking some type of mechanism, in which, where I will change at one place, and change replicate throughout your project.

Sign of architect is that when you have made change in one place , it won't change in multiple places.

The diagram illustrates a software architecture pattern. At the top right, there is a legend with three boxes: 'Controller' (containing 'DAL'), 'IDAL' (containing 'EF', 'ADO', and 'LINQ'), and 'Change all controllers'. A red arrow points from the 'Controller' box to the 'IDAL' box, indicating that changes in the controller code will affect the DAL implementation. The code in the screenshot shows a CustomerController class that instantiates an IDal object. The 'dal' variable is highlighted in blue, and the line of code 'dal = new AdoDal();' is also highlighted.

```
public class CustomerController : Controller
{
    private IDal dal = null;
    public CustomerController()
    {
        if (1 == 1)
        {
            dal = new AdoDal();
        }
        else
        {
            dal = new EfDal();
        }
        else
        {
        }
    }
}
```

The diagram shows the same code structure as the first screenshot, but with a callout box containing the text 'Change at one place should not affect lot of places.' This emphasizes that modifying the controller's DAL dependency should not require changes in multiple places. The code in the screenshot is identical to the first one.

```
public class CustomerController : Controller
{
    private IDal dal = null;
    public CustomerController()
    {
        if (1 == 1)
        {
            dal = new AdoDal();
        }
        else
        {
            dal = new EfDal();
        }
        else
        {
        }
    }
}
```

Controller is not appropriate place creating the of class. Because the work of the controller, is too taking the request, deciding the view.

Invert Object creation

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze
Dal.cs CustomerController.cs* Customer.cs Output
CustomerApplication
public class CustomerController
{
    IDal dal = null;
    public CustomerController()
    {
    }
    // load the customer UI interface
    public IActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
    // will get invoked when add button
    // is clicked
    public IActionResult Add(Customer obj)
}
```

Invert Object creation

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze
Dal.cs CustomerController.cs* Customer.cs Output
CustomerApplication
public class CustomerController
{
    IDal dal = null;
    public CustomerController()
    {
        // inversion of control
    }
    // load the customer UI interface
    public IActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
    // will get invoked when add button
    // is clicked
    public IActionResult Add(Customer obj)
```

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Server Explorer Tools Solution Explorer Tools Task List Tools Output Tools DisplayCustomer.cshtml Tools Startup.cs Tools Routing.cs
CustomerController.cs
public class CustomerController : Controller
{
    IDal dal = null;
    public CustomerController(IDal _dal)
    {
        dal = _dal;
        // inversion of control
    }
    // load the customer UI interface
    public IActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
    // will get invoked when add button
    // is clicked
}
```

Note: ICO is a principle and DI is the way of implement IOC.

Now who decide to create object.

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Server Explorer Tools Solution Explorer Tools Task List Tools Output Tools DisplayCustomer.cshtml Tools Startup.cs Tools Routing.cs
CustomerController.cs
public class CustomerController : Controller
{
    IDal dal = null;
    public CustomerController(IDal _dal)
    {
        dal = _dal;
        // inversion of control
    }
    // load the customer UI interface
    public IActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
    // will get invoked when add button
    // is clicked
}
```

```
public IConfiguration Configuration { get; }
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddScoped<IDal, AdoDal>();
}
// This method gets called by the runtime. Use this method to
// This method gets called by the runtime. Use this method to
public void Configure(IApplicationBuilder app )
{
    app.UseMvc(); // loads MVC
    app.UseMvcWithDefaultRoute(); // loads the defaults
    //app.UseMvc(routes =>
    //{
    //    ApplicationRoutes.LoadRoutes(routes);
    //}); // Loading customer routes
```

```
public IConfiguration Configuration { get; }
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddScoped<IDal, EfDal>();
}
// This method gets called by the runtime. Use this method to
// This method gets called by the runtime. Use this method to
public void Configure(IApplicationBuilder app )
{
    app.UseMvc(); // loads MVC
    app.UseMvcWithDefaultRoute(); // loads the defaults
    //app.UseMvc(routes =>
    //{
    //    ApplicationRoutes.LoadRoutes(routes);
    //}); // Loading customer routes
```

Note: Dependency injection, there is separate framework who actually injects the object and the client just have generic interface which to an interface, in the interface we will get actual concrete type.

CustomerApplication (Running) - Microsoft Visual Studio

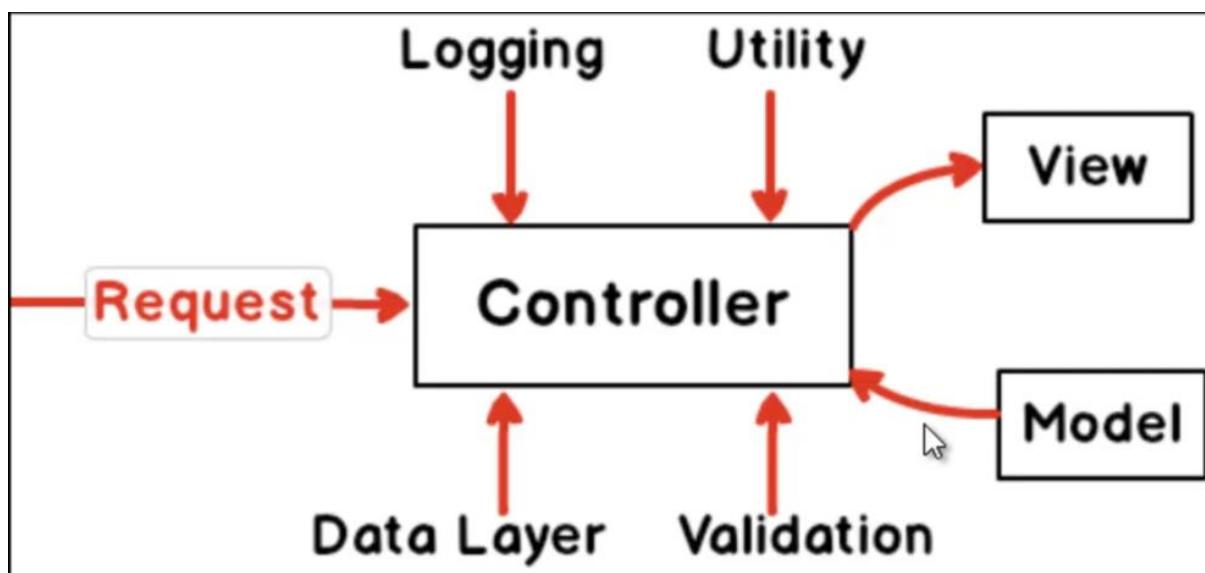
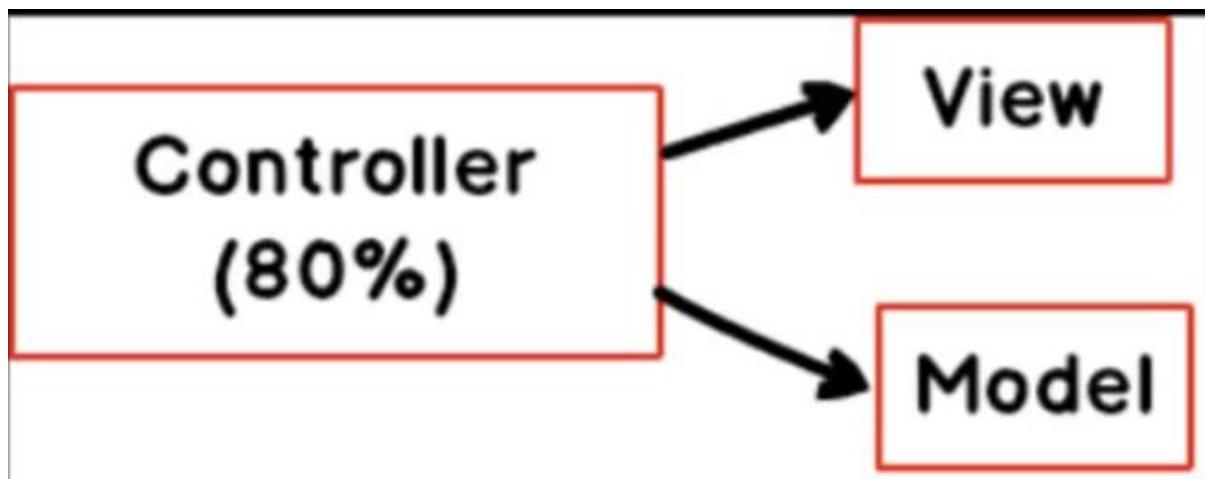
```
CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs Program.cs
CustomerApplication
23     public IConfiguration Configuration { get; }
24     public void ConfigureServices(IServiceCollection services)
25     {
26         services.AddMvc();
27         services.AddScoped<IDal, AdoDal>();
28     }
29
30     // This method gets called by the runtime. Use this method to add
31     // This method gets called by the runtime. Use this method to config
32     public void Configure(IApplicationBuilder app)
33     {
34         app.UseMvc(); // loads MVC
35         app.UseMvcWithDefaultRoute(); // loads the defaults
36         //app.UseMvc(routes =>
37         //    routes
```

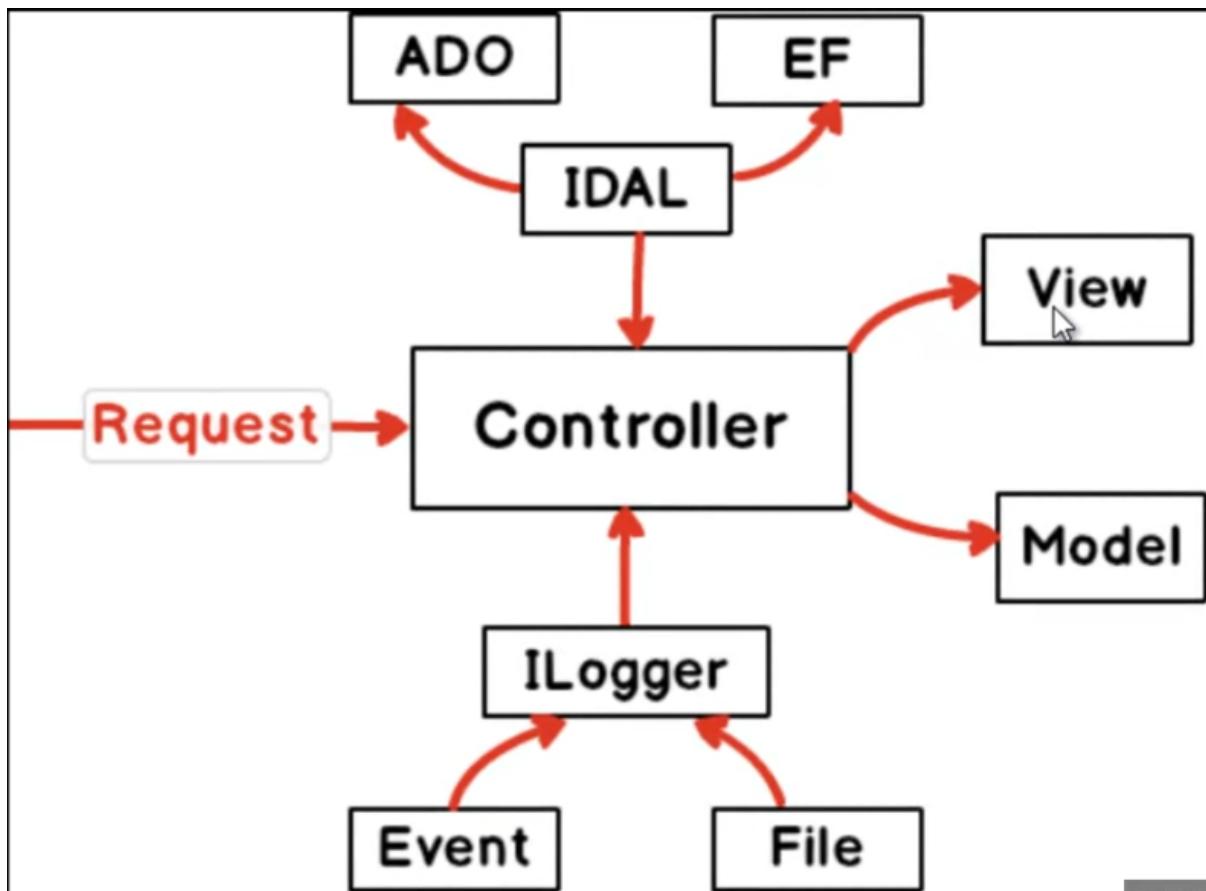
CustomerApplication (Debugging) - Microsoft Visual Studio

```
CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs Program.cs
CustomerApplication
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal)
20         {
21             dal = _dal;
22             // DI of MVC core
23             // inversion of control
24         }
25         // load the customer UI interface
26         public IActionResult LoadCustomer()
27         {
28
29             return View("CustomerScreen");
30         }

```

Note: Now customer controller doesn't do the decision, he has to inject ADODAL or EFDAL. It is done centrally with start-up file.





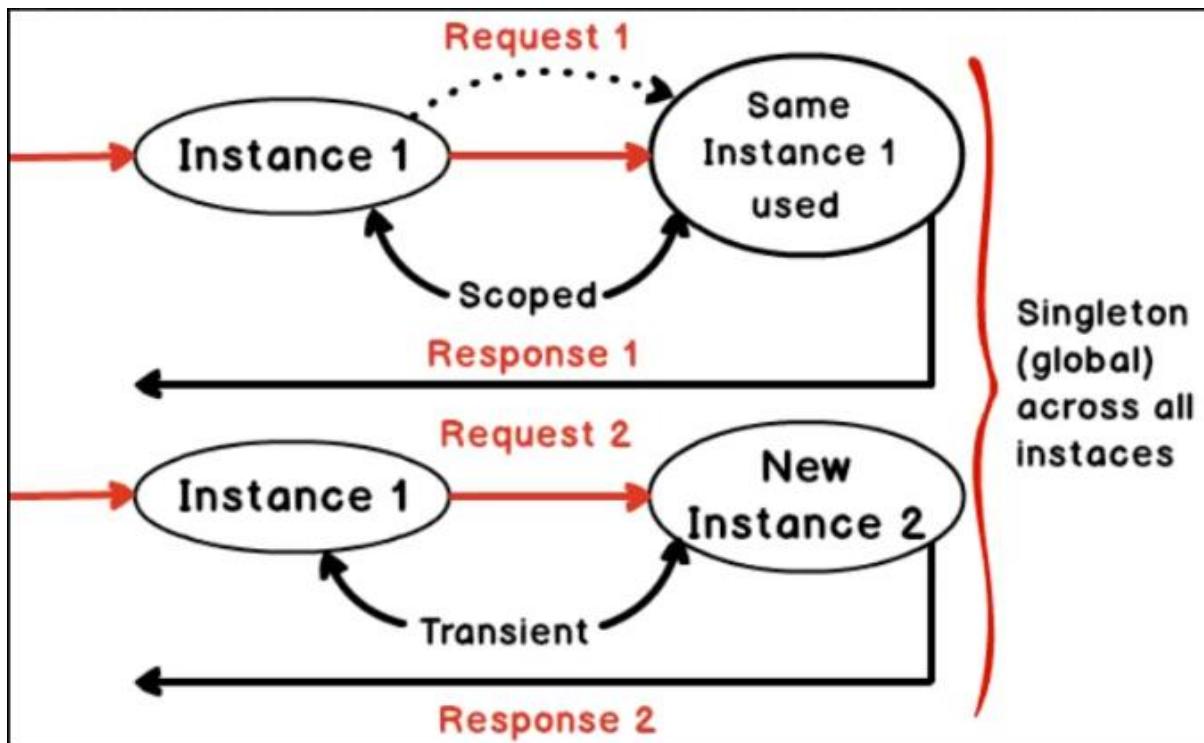
CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

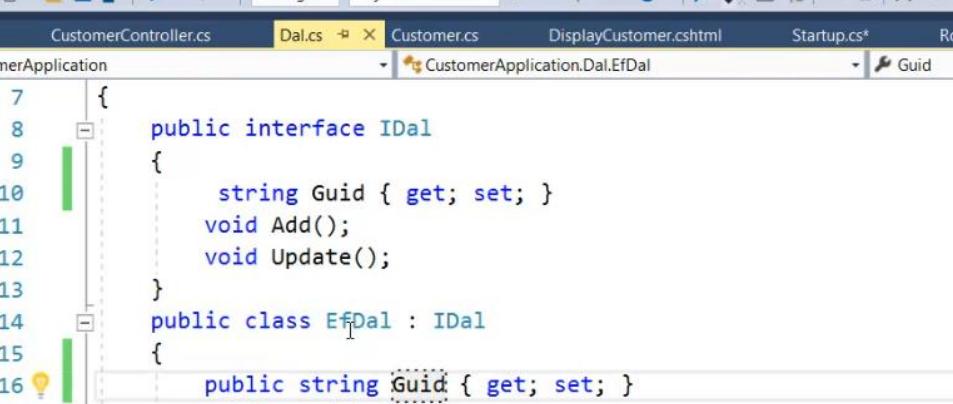
CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs

```

23     public IConfiguration Configuration { get; }
24     public void ConfigureServices(IServiceCollection services)
25     {
26         services.AddMvc();
27         services.AddScoped<IDal, AdoDal>();
28     }
29
30     // This method gets called by the runtime. Use this method to
31     // This method gets called by the runtime. Use this method to
32     public void Configure(IApplicationBuilder app )
33     {
34         app.UseMvc() // loads MVC
35         app.UseMvcWithDefaultRoute(); // loads the defaults
36         //app.UseMvc(routes =>
37         //{
38             // ApplicationRouter LoadRoutes(routes);
39         });
  
```



Now lets add property here and see, object that created how many time. Single time or multiple time.



The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "CustomerApplication - Microsoft Visual Studio". The menu bar includes File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help. The toolbar has icons for Save, Undo, Redo, Cut, Copy, Paste, Find, and others. The status bar at the bottom shows "CustomerApplication.Dal.cs" and "Line 1, Col 1".

The code editor window displays the file `Dal.cs`. The code defines an interface `IDal` with properties `string Guid` and methods `Add()` and `Update()`. It then implements this interface in the class `EfDal`, providing an implementation for `Guid` and a placeholder implementation for `Add()` (throwing a `NotImplementedException`). The code editor uses color coding for syntax and includes a yellow warning sign on line 16.

```
CustomerApplication - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Save Undo Redo Cut Copy Paste Find
CustomerApplication.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs* Routing.cs
CustomerApplication.cs
7 { public interface IDal
8 {
9     string Guid { get; set; }
10    void Add();
11    void Update();
12 }
13 public class EfDal : IDal
14 {
15     public string Guid { get; set; }
16     public void Add()
17     {
18         throw new NotImplementedException();
19     }
20 }
21
22 public void Update()
23 }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Output CustomerController.cs Dal.cs X Customer.cs DisplayCustomer.cshtml Startup.cs* Routing.cs

Server Explorer CustomerApplication CustomerApplication.Dal.AdoDal AdoDal()

```
25     }
26 }
27 public class AdoDal : IDal
28 {
29     public string Guid { get; set; }
30     public AdoDal()
31     {
32         this.Guid = System.Guid.NewGuid().ToString();
33     }
34     public void Add()
35     {
36         throw new NotImplementedException();
37     }
38
39     public void Update()
40 }
```

CustomerApplication (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [18144] dotnet.exe Lifecycle Events Thread: [14668] Worker Thread Stack Frame: CustomerApplication

CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs Program.cs

CustomerApplication CustomerApplication.Controllers.CustomerController CustomerController(IDal _dal)

```
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal)
20             , IDal _dal
21         {
22             dal = _dal;
23             // DI of MVC core
24             // inversion of control
25         }
26         // load the customer UI interface
27         public IActionResult LoadCustomer()
28         {
29
30             return View("CustomerScreen");
31         }
32     }
33 }
```

CustomerApplication (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [18144] dotnet.exe Lifecycle Events Thread: [14668] Worker Thread Stack Frame: CustomerApplication

CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs Program.cs

CustomerApplication

```
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal
20                                     ,IDal _dal1)
21         {
22             dal = _dal;
23             // DI of MVC core
24             ((CustomerApplication.Dal.AdоЯdal)_dal).Guid P ~"8dd051aa-c0fe-4f3a-be4b-920478eb
25             ((CustomerApplication.Dal.AdоЯdal)_dal1).Guid P ~"8dd051aa-c0fe-4f3a-be4b-920478e
26         }
27         [
28
29
30         return View("CustomerScreen");
31     }
32 }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

Output CustomerController.cs Dal.cs Customer.cs IIS Express Customer.cshtml Startup.cs Routing.cs

CustomerApplication

```
23     public IConfiguration Configuration { get; }
24     public void ConfigureServices(IServiceCollection services)
25     {
26         services.AddMvc();
27         services.AddTransient<IDal, AdоЯdal>();
28         //services.addsin
29     }
30     // This method gets called by the runtime. Use this method to
31     // This method gets called by the runtime. Use this method to
32     public void Configure(IApplicationBuilder app )
33     {
34         app.UseMvc(); // loads MVC
35         app.UseMvcWithDefaultRoute(); // loads the defaults
36         //app.UseMvc(routes =>
37         //{
38             // ApplicationRoutes.LoadRoutes(routes);
39         });
40     }
41 }
```

CustomerApplication (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [1444] dotnet.exe Lifecycle Events Thread: [25580] Worker Thread Stack Frame: CustomerApplication

CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs Program.cs

CustomerApplication

```
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal
20                               ,IDal _dal1)
21         {
22             dal = _dal;
23             // DI of MVC core
24             ((CustomerApplication.Dal.AdоЯDal)_dal).Guid = "727439cd-5685-4d1a-a474-a376bb4
25             ((CustomerApplication.Dal.AdоЯDal)_dal1).Guid = "94be821d-9574-4ec1-b546-81cbd0
26             public IActionResult LoadCustomer()
27             {
28
29             return View("CustomerScreen");
30         }
31     }
32 }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

Output CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs

CustomerApplication

```
23     public IConfiguration Configuration { get; }
24     public void ConfigureServices(IServiceCollection services)
25     {
26         services.AddMvc();
27         services.AddSingleton<IDal, AdоЯDal>();
28         //services.addsin
29     }
30     // This method gets called by the runtime. Use this method to
31     // This method gets called by the runtime. Use this method to
32     public void Configure(IApplicationBuilder app )
33     {
34         app.UseMvc(); // loads MVC
35         app.UseMvcWithDefaultRoute(); // loads the defaults
36         //app.UseMvc(routes =>
37         //{
38             // ApplicationRouter.LoadRoutes(routes);
39         });
40     }
41 }
```

```
CustomerController.cs  Dal.cs  Customer.cs  DisplayCustomer.cshtml  Startup.cs  Routing.cs  Program.cs  C
CustomerApplication
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal
20                               ,IDal _dal1)
21         {
22             dal = _dal;
23             // DI of MVC core
24             ((CustomerApplication.Dal.AdoDal)_dal).Guid | "0b37b32e-a818-4165-8df2-c06edd5645d0"
25         }
26         ((CustomerApplication.Dal.AdoDal)_dal1).Guid | "0b37b32e-a818-4165-8df2-c06edd5645
27         public ActionResult LoadCustomer()
28         {
29             ≤ 13ms elapsed
30             return View("CustomerScreen");

```

Note : Now hit again the same request again to see.

```
CustomerController.cs  Dal.cs  Customer.cs  DisplayCustomer.cshtml  Startup.cs  Routing.cs  Program.cs  C
CustomerApplication
16     public class CustomerController : Controller
17     {
18         IDal dal = null;
19         public CustomerController(IDal _dal
20                               ,IDal _dal1)
21         {
22             dal = _dal;
23             // DI of MVC core
24             ((CustomerApplication.Dal.AdoDal)_dal).Guid | "0b37b32e-a818-4165-8df2-c06edd5645d0"
25         }
26         ((CustomerApplication.Dal.AdoDal)_dal1).Guid | "0b37b32e-a818-4165-8df2-c06edd5645
27         public ActionResult LoadCustomer()
28         {
29             ≤ 13ms elapsed
30             return View("CustomerScreen");

```

```
CustomerController.cs
public class CustomerController : Controller
{
    IDal dal = null;
    public CustomerController(IDal _dal
                             ,IDal _dal1)
    {
        dal = _dal; // DI of MVC core
    }
    public ActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
}
```

Note : The two ways you create the instance , by using constructor and another way [FromServices]

```
CustomerController.cs
public class CustomerController : Controller
{
    IDal dal = null;
    public CustomerController([FromServices] IDal _dal
                             ,[FromServices] IDal _dal1)
    {
        dal = _dal; // DI of MVC core
        // inversion of control
    }
    // load the customer UI interface
    public ActionResult LoadCustomer()
    {
        return View("CustomerScreen");
    }
}
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Output CustomerController.cs Dal.cs Customer.cs DisplayCustomer.cshtml Startup.cs Routing.cs

CustomerApplication

```
7  {
8      public class Customer
9      {
10         public string CustomerCode { get; set; }
11         public string CustomerName { get; set; }
12         public double CustomerAmount { get; set; }
13     }
14     public class GoldCustomer : Customer
15     {
16     }
17     public class DiscountedCustomer : Customer
18     {
19     }
20 }
21 }
22 }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze

Output CustomerController.cs Dal.cs Customer.cs

CustomerApplication

```
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

```
public IServiceProvider ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddScoped<IDal, AdoDal>();
    //services.addsin
}

// This method gets called by the runtime. Use this method to
// This method gets called by the runtime. Use this method to
public void Configure(IApplicationBuilder app )
{
    app.UseMvc(); // loads MVC
    app.UseMvcWithDefaultRoute(); // loads the defaults
    //app.UseMvc(routes =>
    //{
        // ApplicationRouter.LoadRoutes(routes);
    //}
}
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

CustomerController.cs Customer.cs DisplayCustomer.cshtml Startup.cs X Routing.cs Program.cs CustomerSc

CustomerApplication

```
24     public IConfiguration Configuration { get; }  
25     public void ConfigureServices(IServiceCollection services)  
26     {  
27         services.AddMvc();  
28         services.AddScoped<IDal, AdoDal>();  
29         services.AddScoped<Customer>((ctx) =>  
30         {  
31             IHttpContextAccessor con =  
32                 ctx.GetService< IHttpContextAccessor>();  
33             double amt = Convert.ToDouble(  
34                 con.HttpContext.Request.Form["CustomerAmount"]);  
35             if (amt > 100)  
36             {  
37                 return GoldCustomer();  
38             }  
39         });  
40     }
```

CustomerApplication - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug Any CPU IIS Express

CustomerController.cs Customer.cs DisplayCustomer.cshtml Startup.cs X Routing.cs Program.cs CustomerSc

CustomerApplication

```
30     {  
31         IHttpContextAccessor con =  
32             ctx.GetService< IHttpContextAccessor>();  
33             double amt = Convert.ToDouble(  
34                 con.HttpContext.Request.Form["CustomerAmount"]);  
35             if (amt > 100)  
36             {  
37                 return new GoldCustomer();  
38             }  
39             else  
40             {  
41                 return new DiscountedCustomer();  
42             }  
43     }  
44 }  
45 // This method gets called by the runtime. Use this method to
```

```
CustomerController.cs
25 }
26 // load the customer UI interface
27 public IActionResult LoadCustomer()
28 {
29
30     return View("CustomerScreen");
31 }
32 // will get invoked when add button
33 // is clicked
34 public IActionResult Add([FromServices] Customer obj)
35 {
36     return View("DisplayCustomer", obj);
37 }
38 public IActionResult Update()
39 {
40     return View();
}
```

By Condition , the FromService customer obj , its depends on factory code we have written into this start-up file.

```
Startup.cs
30 {
31     IHttpContextAccessor con =
32         ctx.GetService<IHttpContextAccessor>();
33     string str123 = con.HttpContext.GetRouteValue("Controller");
34     string str = con.HttpContext.Request.Query["test"];
35     double amt = Convert.ToDouble(
36         con.HttpContext.Request.Form["CustomerAmount"]);
37     if (amt > 100)
38     {
39         return new GoldCustomer();
40     }
41     else
42     {
43         return new DiscountedCustomer();
44     }
45 }
```

What is IOC and DI conceptually?

In Which file do we define MVC CORE DI objects?

Differentiate between Transient, Scoped, Singleton and Factory.

What is the syntax of DI in MVC core?

What is the use of ContextAccessor and how do we get it ?