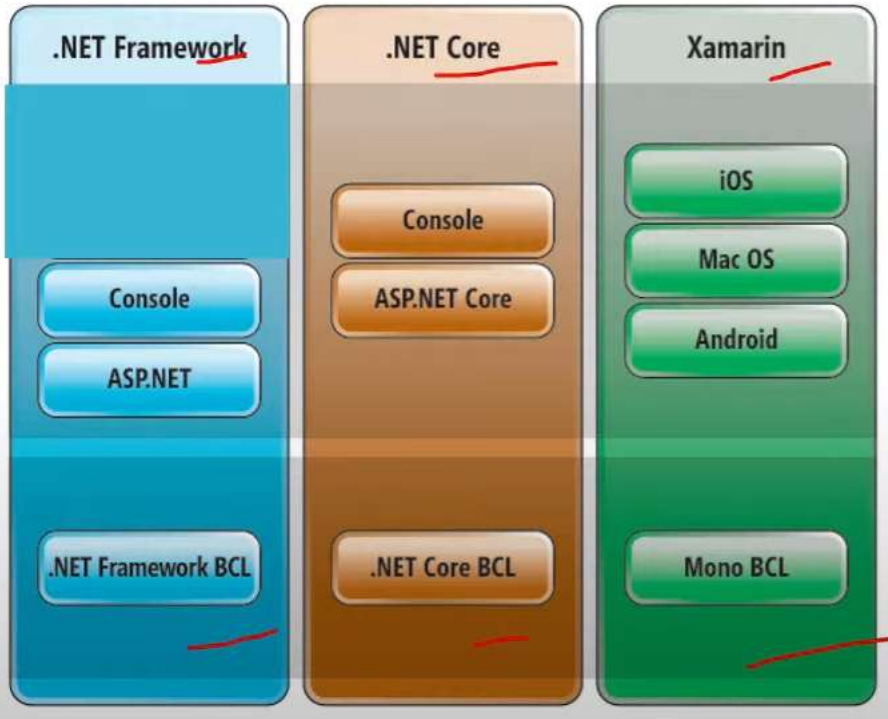
What is .NET Core?

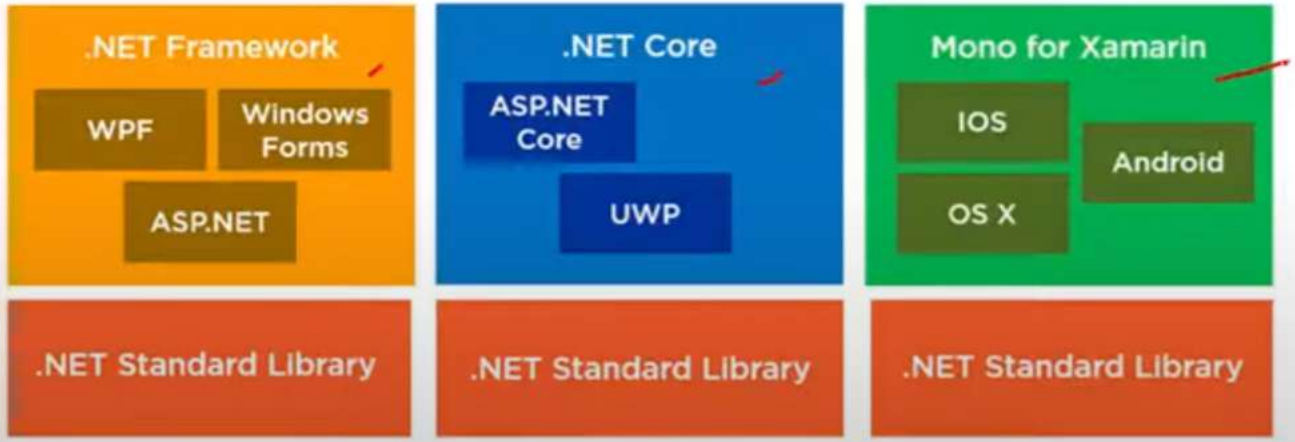
- .NET Core is completely a **NEW** framework, which is a **FREE** and **OPEN-SOURCE** platform developed and maintained by Microsoft.



What is .NET Standard ?

- .NET Standard is not a framework.

- .NET Standard defines a set of rules which if any base class library of any framework will follow then the framework will be called .NET Standard compliant.



Q What is the role of Startup.cs file?

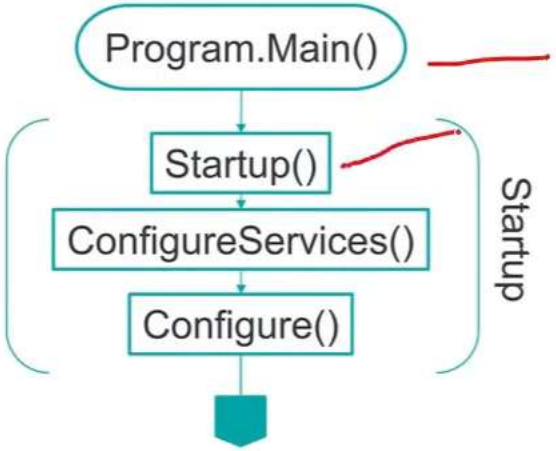
.NET Core

```
2 references
public class Startup
{
    0 references
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    1 reference
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime.
    // Use this method to add services to the container.
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllersWithViews();
    }

    // This method gets called by the runtime.
    // Use this method to configure the HTTP request pipeline.
    0 references
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days.
            // You may want to change this for production scenarios, see https://ak
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
        });
    }
}
```

1. Startup constructor will set the configuration related things with IConfiguration interface.

2. ConfigureServices method configures the SERVICES which are required by the application.

3. CONFIGURE method defines the application REQUEST HANDLING PIPELINE as a series of middleware components.



Q

What are the advantages of .NET Core over .NET framework? V. IMP.

.NET Core

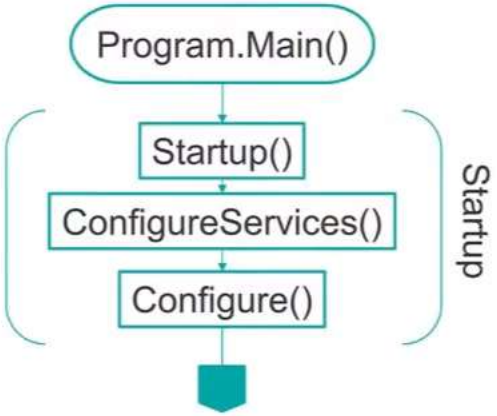
CROSS PLATFORM	OPEN SOURCE	HOSTING	BUILT-IN DEPENDENCY INJECTION	SUPPORT MULTIPLE IDE
• Windows • Linux • MacOS	• Free to use • Modify • Distribute	• Kestrel • IIS • Nginx	• Loosely Coupled Design • Reusability • Testability	• Visual Studio • Visual Studio for Mac • Visual Studio Code
.NET Framework only supports Windows	.NET Framework is paid	.NET Framework only support IIS Hosting	.NET framework don't have built in dependency injection	.NET framework only support Visual Studio IDE

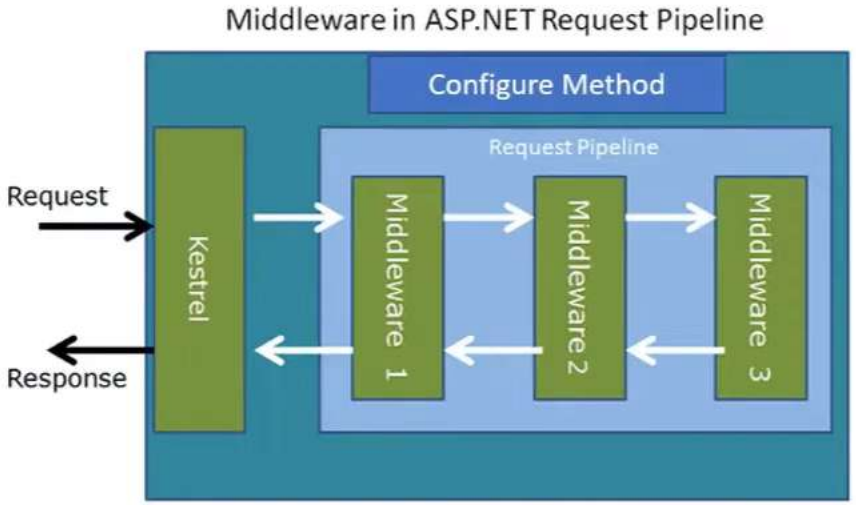
Q What is the role of Configure method?

.NET Core

- Configure method will configure the request pipeline.

```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```





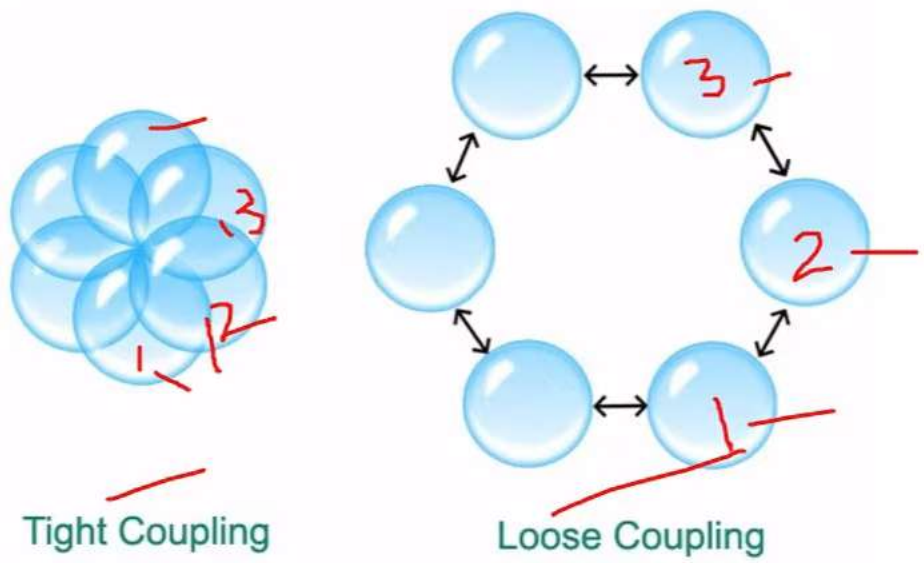
What is Dependency Injection in ASP.NET Core?

V. IMP.

.NET Core

- Dependency Injection (DI) is a software **design pattern** that allows us to develop **loosely coupled application**.

- This is a process in which we are injecting the object of a class into another class that depends on that object.



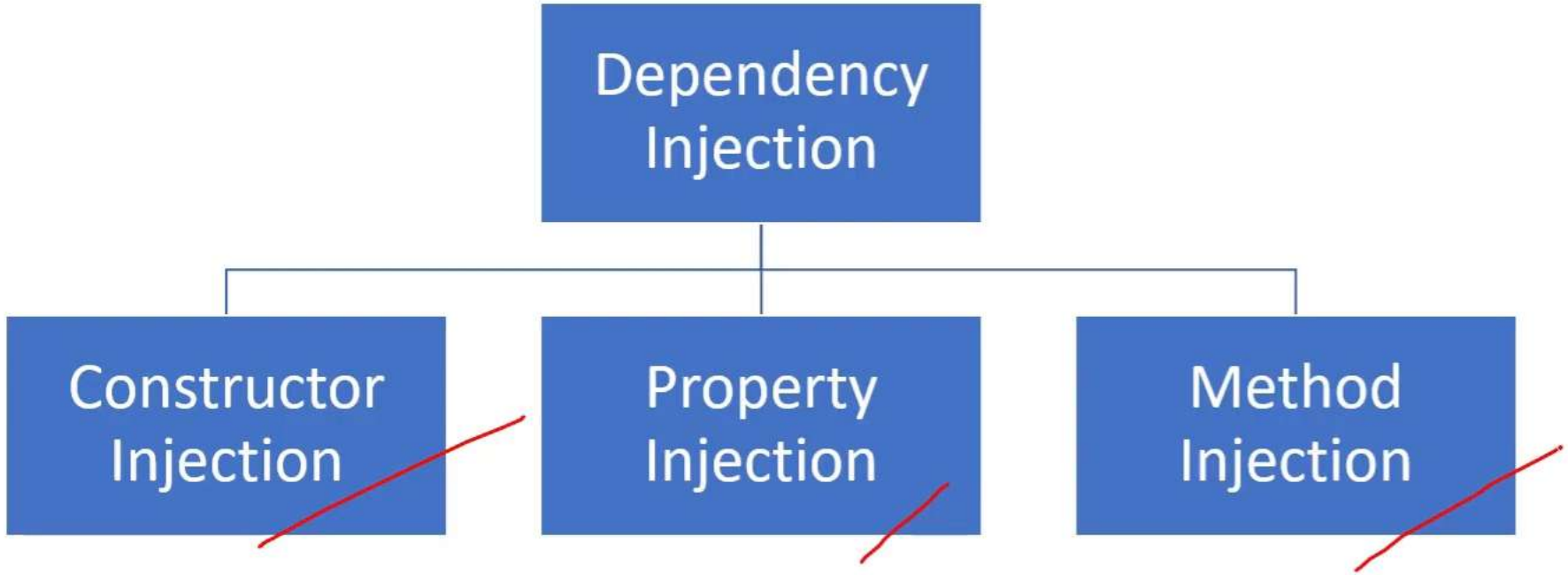

```
public class A
{
    public static void Main(string[] args)
    {
        B b = new B();
        int a = b.MethodB();
        Console.WriteLine(a);
    }
}
```

```
public class B
{
    public int MethodB()
    {
        return 100;
    }
}
```

Q

What are the types of Dependency Injection?

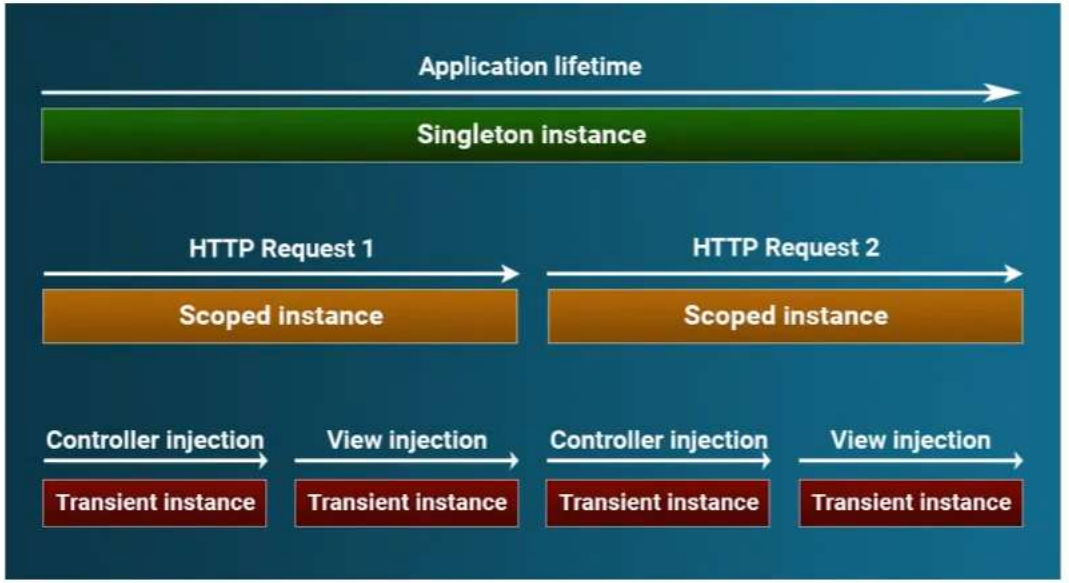
.NET Core



Q What are the types of Service Lifetimes of an object/ instance in ASP.NET Core?

.NET Core

- Service lifetimes describe for how long the instance of any class will persist.



Types of Service Lifetimes

1. AddSingleton
2. AddScoped
3. AddTransient

SUBSCRIBE

How to use Dependency Injection in Views in ASP.NET Core?

- A service can be injected into a view using the @inject directive.

```
4 references
public interface IStudent
{
    2 references
    public int GetStudentCount();
}
```

```
1 reference
public class MathStudent:IStudent
{
    2 references
    public int GetStudentCount()
    {
        return 100;
    }
}
```

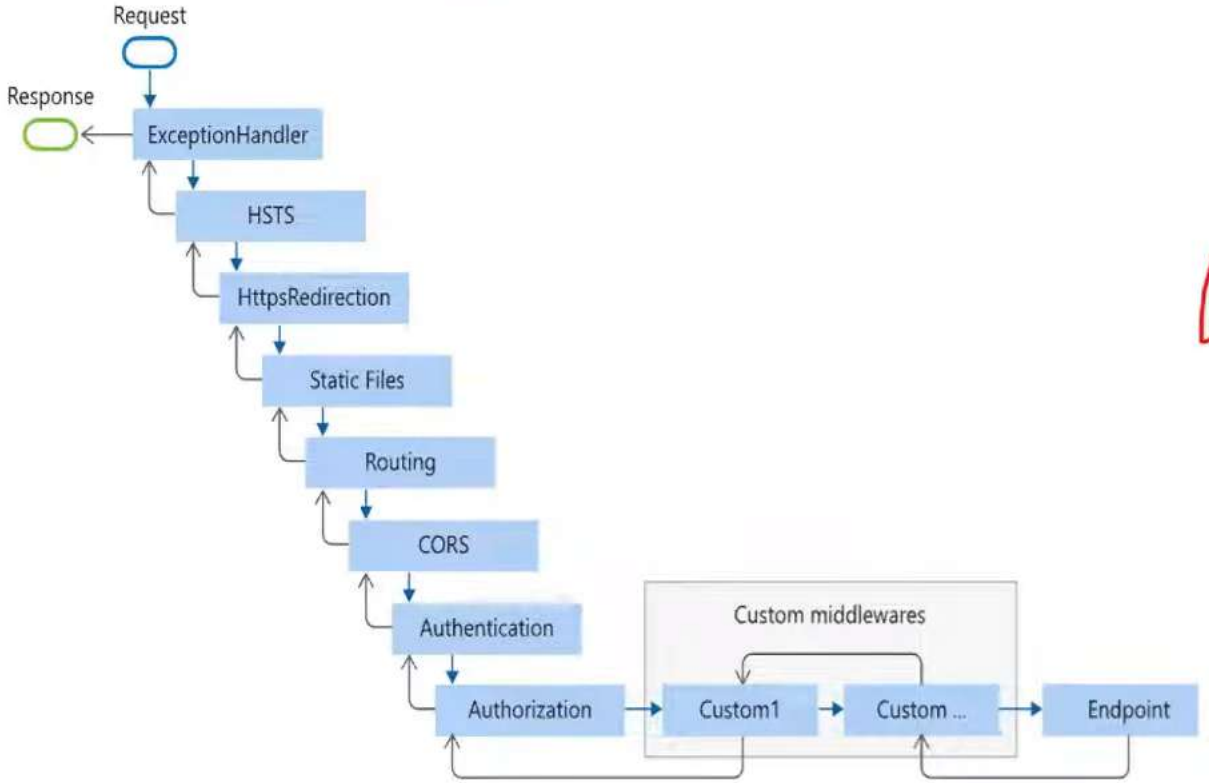
```
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddScoped<IStudent, MathStudent>();
}
```

```
@{
    ViewData["Title"] = "Home Page";
}
@inject WebApplication1.IStudent _student

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    Total Student: @_student.GetStudentCount();
</div>
```

Q What is Middleware in ASP.NET Core? V. IMP.

.NET Core

- A middleware a component that is executed on EVERY REQUEST in the ASP.NET Core application.
- We can set up the middleware in ASP.NET using the CONFIGURE method of our STARTUP class.




```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

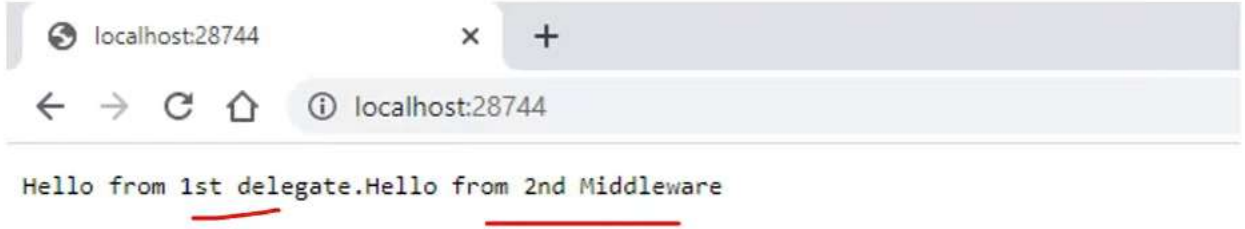
SUBSCRIBE

Q What is Run(), Use() and Map() method? V. IMP.

.NET Core

- Use method will execute next middleware or line in sequence.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.Use(async (context,next) =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
                await next();
            });
            app.Run(async (context) =>
            {
                await context.Response.WriteAsync("Hello from 2nd Middleware");
            });
        }
```




```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

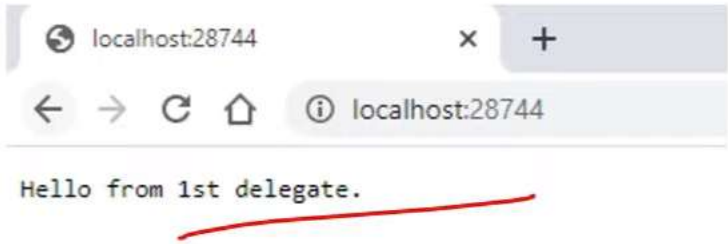
SUBSCRIBE

Q What is Run(), Use() and Map() method? V. IMP.

.NET Core

- Run method will TERMINATE the chain.
- No other middleware method will run after this.
- Should be placed at the end of any pipeline.

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

            app.Run(async context =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
            });
            app.Run(async (context) =>
            {
                await context.Response.WriteAsync("Hello from 2nd Middleware");
            });

        }
```



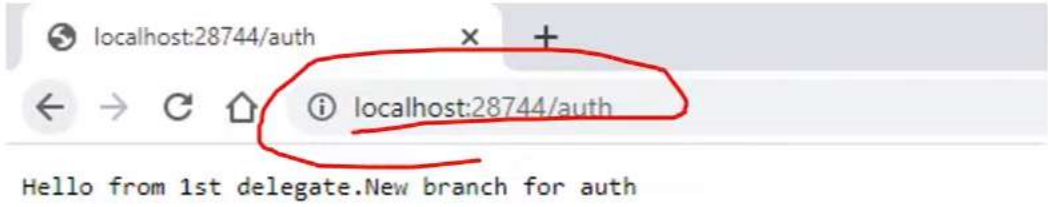
Q What is Run(), Use() and Map() method? V. IMP.

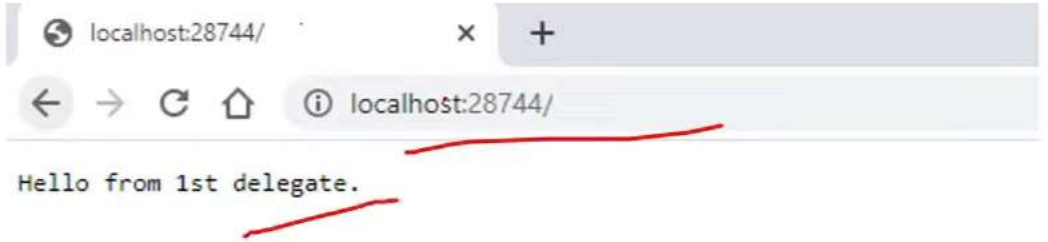
.NET Core

- The **Map** extension **method** is used to match request delegates based on a request's URL path.

localhost:28744/auth

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
      {
          app.Use(async (context, next) =>
          {
              await context.Response.WriteAsync("Hello from 1st delegate.");
              await next();
          });

          app.Map("/auth", a =>
          {
              a.Run(async (context) =>
              {
                  await context.Response.WriteAsync("New branch for auth");
              });
          });

       }
```





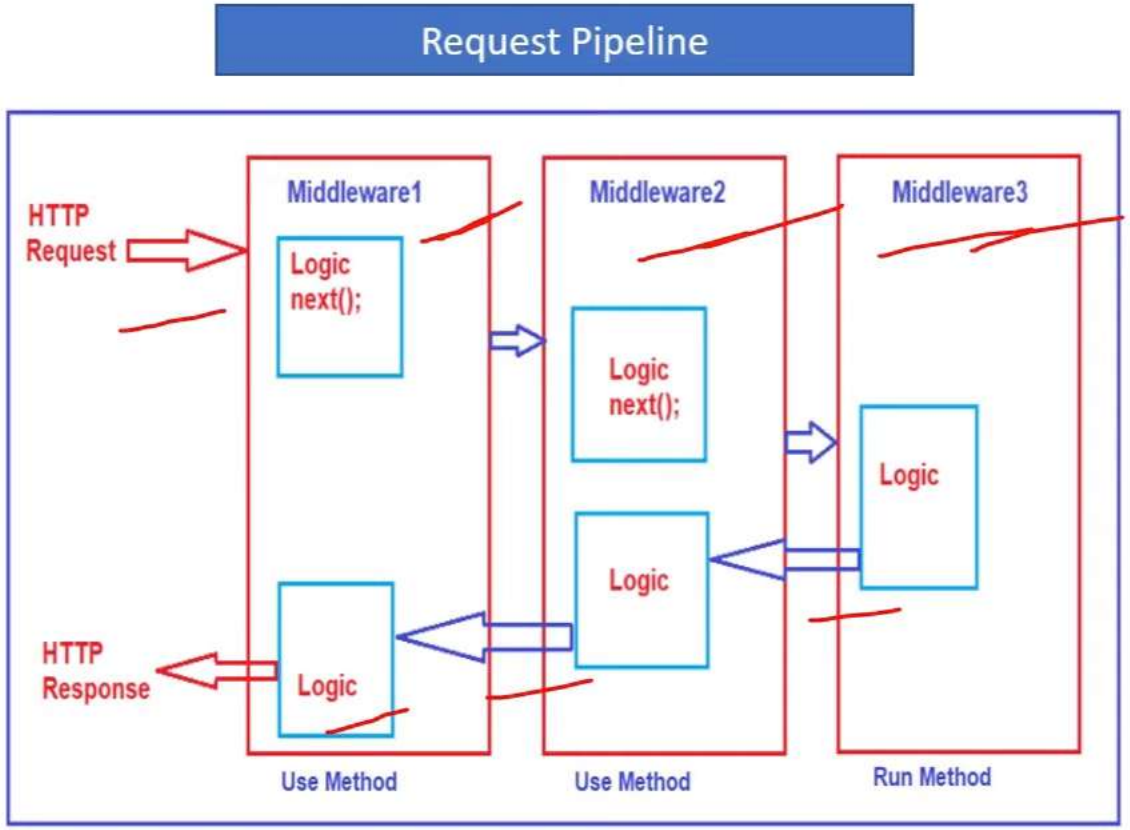
SUBSCRIBE

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.Use(async (context, next) =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
                await next();
            });

            app.Map("/auth", a =>
            {
                a.Run(async (context) =>
                {
                    await context.Response.WriteAsync("New branch for auth");
                });
            });
        }
```

Q What is Request Delegate in .NET Core?

.NET Core

- Request delegates are used to build the request pipeline.
- Request delegates handle request pipeline by using RUN, MAP and USE extension methods.



Q What is Routing? Explain attribute routing in ASP.NET Core?

.NET Core

- Routing is used to handle incoming HTTP requests based on the URL.

http://localhost:52190/Home/Index

```
public class HomeController : Controller
{
    public ViewResult Index()
    {
        return View();
    }
}
```

- Attribute based routing is the ability to manipulate the behavior of URL by Route Attribute.

Example
http://localhost:60995/Home/NewIndex

```
[Route("")]
[Route("NewIndex")]
0 references
public IActionResult Index()
{
    return View();
}
```

SUBSCRIBE

Q

What are the main JSON files available in ASP.NET Core?

.NET Core

- **launchsettings.json** – You can set the things here which are needed when the application is launching or starting.

  For example, set development or production environment in this file.

- **appsettings.json** – Configuration settings like database connection string can be set in this file.

  Like web.config in ASP.NET.

- **project.json** - ASP.NET Core uses Project.JSON file for storing all project level configuration settings.

  For example, the nugget packages you have installed in the project.

- **global.json** - You can define the solution level settings in global.json file.

  For example, your application name and version.

Q

What is the difference between Appsetting.Json and Launchsetting.Json file?

.NET Core

- Appsetting.Json store the configurations which are required when your application is running. For example, database connection string is used when application is running.

- But Launchsetting.Json store the configurations which are required to start the application. For example, things like application URL are stored here.

Q

What are the various techniques to save configuration settings in ASP.NET Core?

.NET Core

- Appsettings.json (Default) (Mostly Used)
- Azure Key Vault (Mostly used and it's the best if you are using Azure)
- Environment variables
- In-memory .NET objects
- Command Line Arguments
- Custom Providers

Q

What is In-Memory caching & Distributed Caching?
When to use In-Memory caching and when to use Distributed caching?

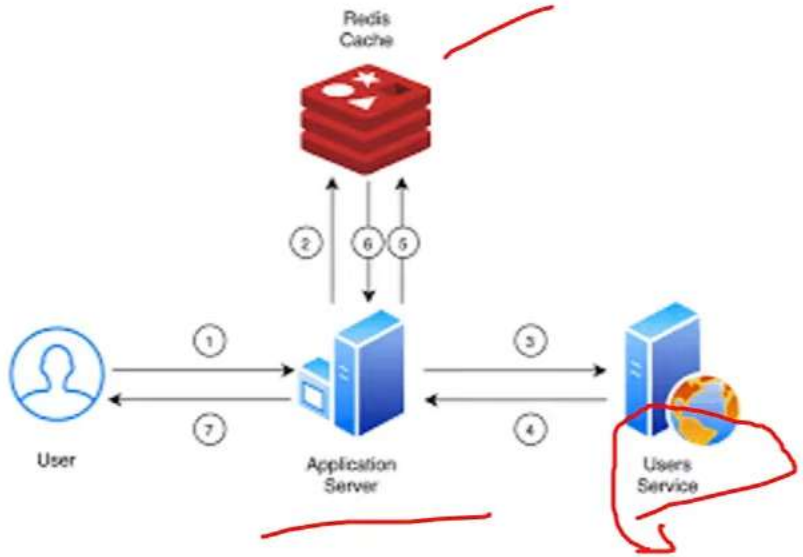
.NET Core

In-Memory Caching	Distributed Caching
1. It's the normal way of caching. In this cache is stored in the memory of a **single server** which is hosting the application.	Distributed caching is when you want to handle caching outside of your application. A different server is used for store cached data.
2. It can be implemented with the IMemoryCache Interface in ASP.NET Core.	It can be implemented with the help of Redis Cache.

- Redis is an open-source, highly replicated, performant, non-relational kind of database and caching server.

- When to use which caching?

In normal cases where the application size is small, use in-memory cache.
But where application is very big or it's a microservices based architecture, then use distributed caching.



SUBSCRIBE