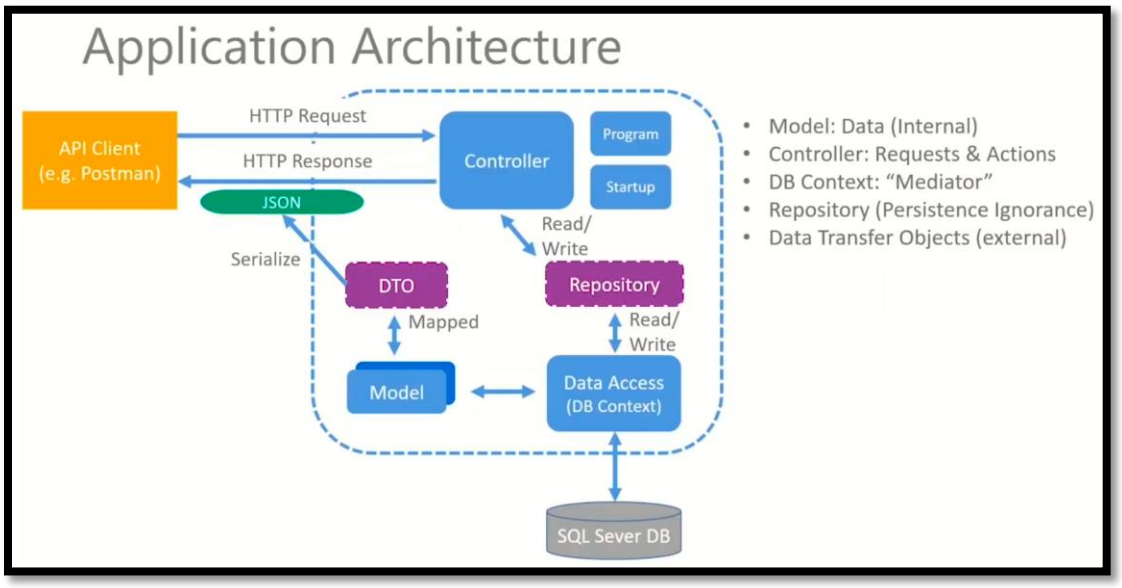
Learning ASP.NET MVC WEB API Core

In

Visual Studio Code = VSCode

Mr. Dariush Tasdighi

Document Version 1.4



Open Visual Studio Code

Update it!

Help → Check for Updates…

About → Version 1.48.1

CTRL + ` (Backtick) → Terminal (PowerShell)

D:\

MD SourceCodes

CD SourceCodes

MD LearningWebApiCore

CD LearningWebApiCore

Display current .NET Core Version:

dotnet --version

dotnet --info

Display All .NET Core Templates:

dotnet new

dotnet new webapi -n Commander

dir

cd Commander

dir

Solution (2)

code .

Solution (1) → It is better!

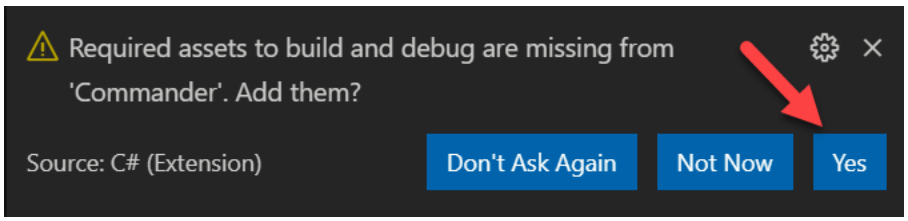
code -r .

With or Without Open → WeatherForecast.cs

Visual Studio Code Alert:

Required assets to build and debug are missing from 'Commander'. Add them?

Yes



Describe Some Files:

Program.cs

Startup.cs and 3 Functions

```
namespace Commander
{
    1 reference
    public class Startup
    {
        0 references
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        1 reference
        public IConfiguration Configuration { get; }

        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
        }

        0 references
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseHttpsRedirection();

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

Commander.csproj

appsettings.json

Appsettings.Development.json

Properties → launchSettings.json

Edit startup.cs:

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

0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    // app.UseHttpsRedirection();

    app.UseRouting();

    // app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Build:

…\Commander> dotnet build

Run:

…\Commander> dotnet run

…\Commander> dotnet watch run

Browser:

http://localhost:5000/ → Error

http://localhost:5000/WeatherForecast → OK

http://localhost:5000/WeatherForcast/Get → Error

Delete Files:

WeatherForecast.cs

Controllers Folder → WeatherForecastController.cs

Create Folder: Models

Create Command Model → Command.cs

int Id

string HowTo

string Line

string Platform

Create Folder: Data

Create Repository →

Temp Repository:

ThisIsNotAGoodRepository → Mock Repository که در واقع یک

In Controllers Folder, Create CommandsController

CommandsController.cs

Model: Command → Controller → In Polar & Pascal → **Commands**Controller

Run Application:

dotnet run

dotnet watch run

Create Folder: Data

Create IRepository & Repository →

ICommandRepository

```
Command GetById(int id);
System.Collections.Generic.IEnumerable<Models.Command> GetAll();
```

CommandRepository

```
Command GetById(int id);
System.Collections.Generic.IEnumerable<Models.Command> GetAll();
```

Modify Startup.cs File:

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

    // ServiceCollection Lifetimes:
    // 1. AddSingleton
    //      Same for every request
    // 2. AddScoped
    //      Created once per client request
    // 3. Transient
    //      New instance created every time
    services.AddScoped<Data.ICommandRepository, Data.CommandRepository>();
}
```

Using Entity Framework Core Code First

Go to Terminal (CTRL + `)

…\Commander>

dotnet add package Microsoft.EntityFrameworkCore

Commander.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="3.1.6" />
  </ItemGroup>

</Project>
```

…\Commander>

dotnet add package Microsoft.EntityFrameworkCore.Design

Commander.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="3.1.6" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="3.1.6">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
  </ItemGroup>

</Project>
```

…\Commander>

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

Commander.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="3.1.6" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="3.1.6">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="3.1.6" />
  </ItemGroup>

</Project>
```

…\Commander>

dotnet ef

در صورتی که با خطا مواجه شدیم، باید ابتدا دستور ذیل را اجرا نماییم

dotnet tool install --global dotnet-ef

در پوشه Data فایلی به نام DatabaseContext ایجاد می‌کنیم:

Data > DatabaseContext.cs > ...

```
1   namespace Data
2   {
        2 references
3       public class DatabaseContext : Microsoft.EntityFrameworkCore.DbContext
4       {
            0 references
5           public DatabaseContext
6               (Microsoft.EntityFrameworkCore.DbContextOptions<DatabaseContext> options) : base(options)
7           {
8
9           }
10
            0 references
11          public Microsoft.EntityFrameworkCore.DbSet<Models.Command> Commands { get; set; }
12      }
13  }
```

Modify appsettings.json:

Add Connection Strings:

appsettings.json > ...

```
1   {
2     "Logging": {
3       "LogLevel": {
4         "Default": "Information",
5         "Microsoft": "Warning",
6         "Microsoft.Hosting.Lifetime": "Information"
7       }
8     },
9     "AllowedHosts": "*",
10    "ConnectionStrings": {
11      "CommanderConnectionString": "Server=.;Initial Catalog=CommanderDB;User ID=CommanderUser;Password=1234512345"
12    }
13  }
```

Modify Startup.cs

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    // For Using option.UseSqlServer --> using Microsoft.EntityFrameworkCore;
    services.AddDbContext<Data.DatabaseContext>(option =>
    {
        option.UseSqlServer(Configuration.GetConnectionString("CommanderConnectionString"));
    });

    services.AddControllers();
```

Create Database with Migrations:

برای انجام Migrations ابتدا یک بار پروژه را Compile می‌کنیم که هم فایل dll ایجاد شده و هم اطمینان حاصل کنیم که پروژه هیچ خطایی نداشته باشد!

dotnet build

Go to Terminal (CTRL + `)

…\Commander>

dotnet ef migrations add InitialMigration

نکته: در زمان اجرای دستورات Migrations باید پروژه Stop باشد!

با این دستور، یک پوشه به نام Migrations و تعدادی فایل در داخل آن ایجاد می‌شود!

دقت کنید که با اجرای دستور فوق، بانک اطلاعاتی ایجاد نمی‌شود!!!

به کدهای Migrations توجه می‌کنیم! در صورتی که مورد علاقه ما نبود

با استفاده از دستور ذیل، Migrations را حذف می‌کنیم:

dotnet ef migrations remove

Modify Command Model:

Command.cs

```
Models > Command.cs > ...
1   namespace Models
2   {
        22 references
3       public class Command : object
4       {
            2 references
5           public Command() : base()
6           {
7           }
8
9           [System.ComponentModel.DataAnnotations.Key]
            4 references
10          public int Id { get; set; }
11
12          [System.ComponentModel.DataAnnotations.Required]
            2 references
13          public string Line { get; set; }
14
15          [System.ComponentModel.DataAnnotations.Required]
16          [System.ComponentModel.DataAnnotations.MaxLength(length: 250)]
            2 references
17          public string HowTo { get; set; }
18
19          [System.ComponentModel.DataAnnotations.Required]
            2 references
20          public string Platform { get; set; }
21      }
22  }
```

Go to Terminal (CTRL + `)

…\Commander>

dotnet build

dotnet ef migrations add InitialMigration

برای ایجاد بانک اطلاعاتی:

dotnet ef database update

بعد از ایجاد بانک اطلاعاتی، برای تست دو رکورد در جدول Commands به شرح ذیل ایجاد می‌کنیم:

Id: 1

HowTo: How to create migrations

Line: dotnet ef migrations add <Name>

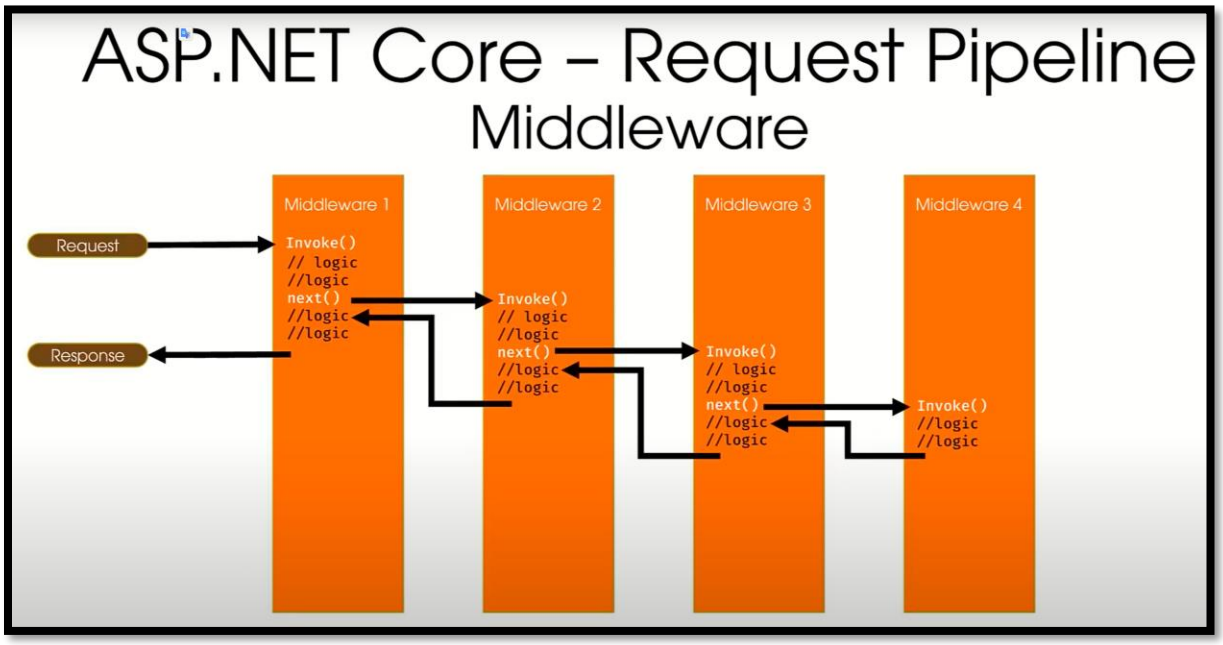
Platform: EF Core

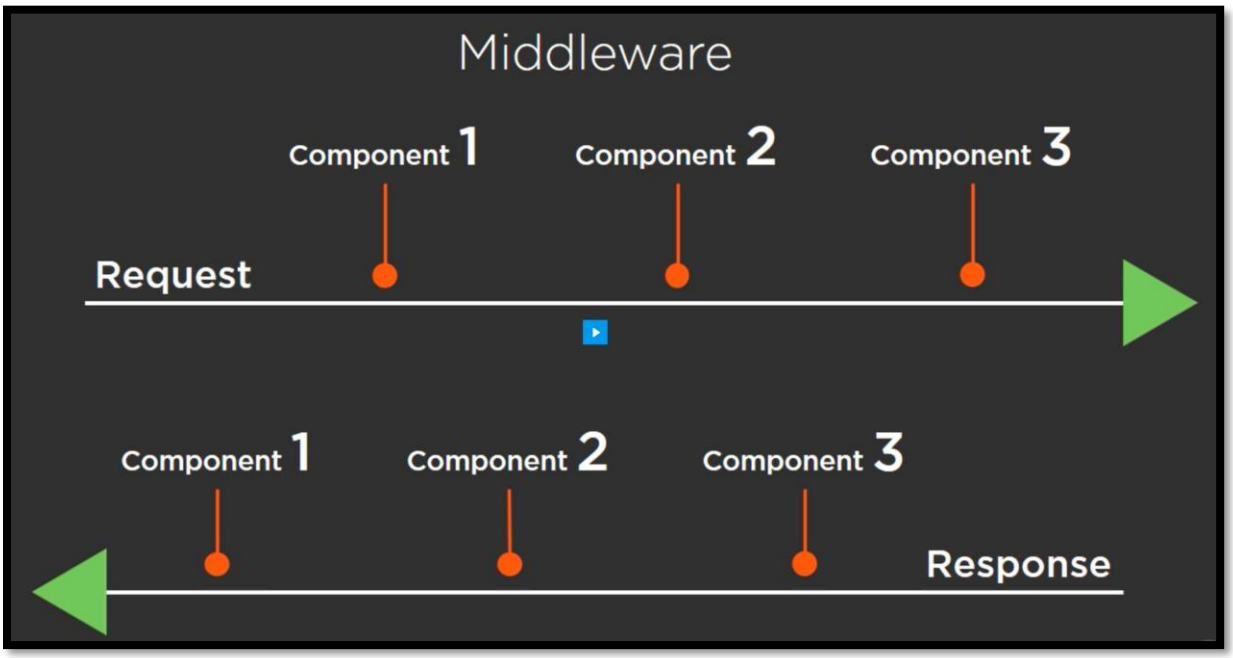
Id: 2

HowTo: How to run migrations

Line: dotnet ef database update

Platform: EF Core


ASP.NET Core – Request Pipeline
Middleware
Middleware 1
Middleware 2
Middleware 3
Middleware 4
Request
Response
Invoke()
// logic
//logic
next()
//logic
//logic


Middleware
Component 1
Component 2
Component 3
Request
Component 1
Component 2
Component 3
Response

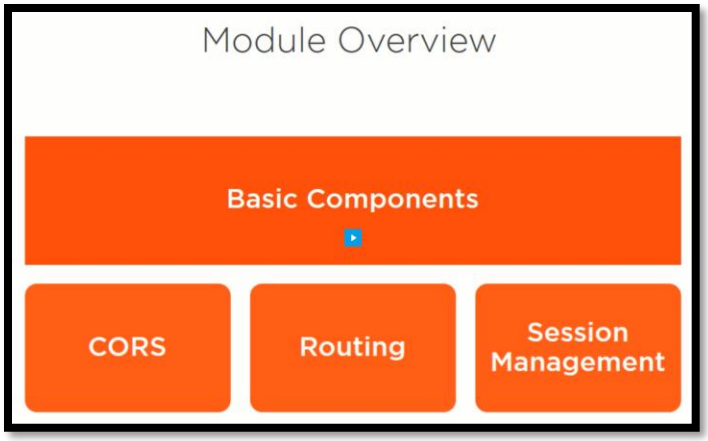
- UseDeveloperExceptionPage
- UseDatabaseErrorPage
- UseExceptionHandler
- UseHsts
- UseHttpsRedirection
- UseStaticFiles
- UseAuthentication
- UseSession
- UseMvc

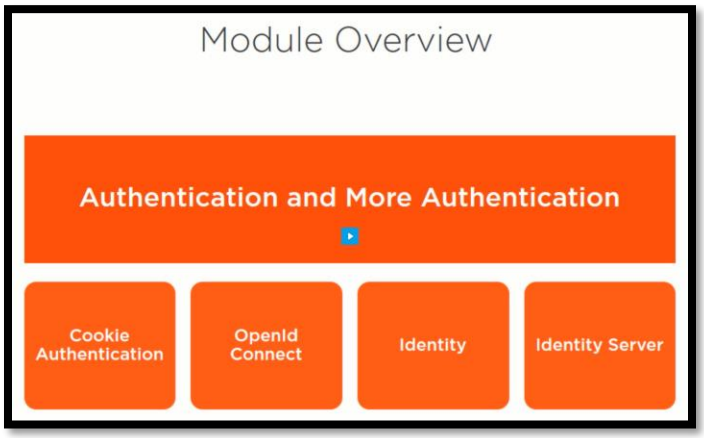
- `Run()`
  - Delegates Terminate the Request Pipeline
- `Use()`
  - Multiple Request Delegates can be chained
- Short-circuiting
  - When a delegate does not call the "next" delegate

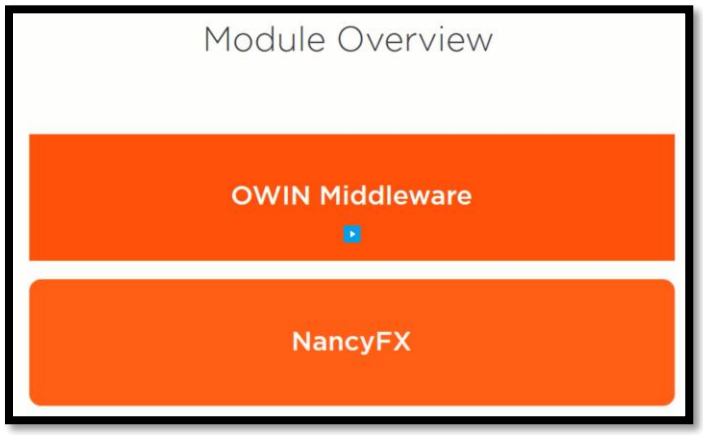
Pre-requisites for Custom Middleware class

- Public constructor with `RequestDelegate` argument
- Public method named `Invoke` or `InvokeAsync`
  - First argument Must be `HttpContext`
  - Must return as `Task`
  - Additional arguments can be injected

*** Created **once** per application lifetime – at startup ***


Module Overview
Basic Components
CORS
Routing
Session Management


Module Overview
Authentication and More Authentication
Cookie Authentication
OpenId Connect
Identity
Identity Server


Module Overview
OWIN Middleware
NancyFX