ASP.NET Core 2.x

Consist of

1. Web Forms
   1. Http Request Processing for Views
2. MVC
   1. Http Request processing for Controllers
3. WEB API
   1. Http Request Processing for Controllers+Data

Dotnetcore SDK

1. Common Libraries
   1. Collection
   2. Thread
   3. Xml

ASP.NET Core App

1. Security
   1. OAuth
   2. OpenIdConnect
   3. JWT
   4. External Auth
      1. Google
      2. Microsoft
      3. Facebook
      4. Twitter
2. Razor
3. API
4. CORS
5. Hosting
   1. InProc
      1. HttpSys 🡪 IIS Express
      2. IISHosting

IServiceCollection, the interface that manages all Service Objects for app

1. Cookies
2. DI Objects
3. Security 🡪 Authentication and Authorization with Policies aka RBS
4. CORS Policies
5. MVC

Use EF Core Migrations for Generating DB and Tables

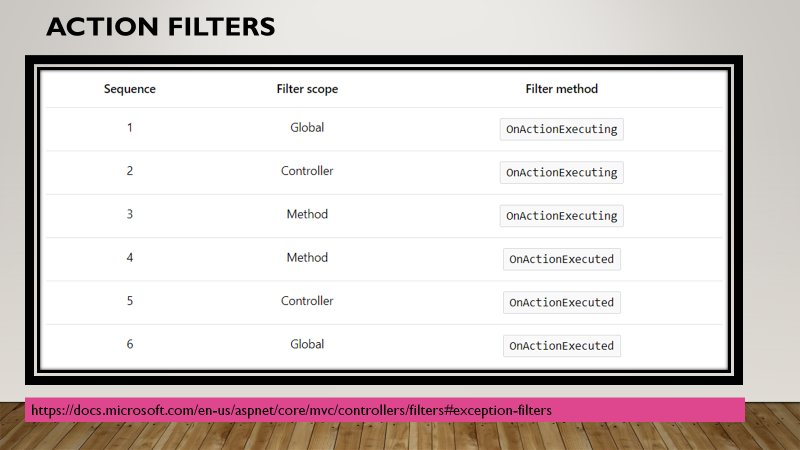
Dotnet ef migrations add <MIGRATIOn-NAME> --context <FULL-QUALIFY-NAMESPACE-DbContext>

Dotnet ef database update -c <FULL-QUALIFY-NAMESPACE-DbContext>

The DI Registration features for Repositories in ASP.NET Core App

1. Transient 🡪 Stateless Object Instance, this object cannot be shared. The instance is created for each request 🡪 PerCall
2. Scoped 🡪 The Object will be created for every new request (Session). The object can be shared across requests in the current session. Stateful.
   1. Logger Objects
   2. Authentication Object
3. Singleton 🡪 Created only Once and the life is at application level
   1. DbContext
   2. Caching
   3. CORS

ASP.NET Core WEB API Programming

1. Custom Validations
   1. Data Annotations
   2. **Process Based Validations using Exceptions**
      1. **Try..catch block**
2. Filters
   1. Additional Service Objects for ASP.NET Core, those are used for
      1. MVC
      2. WEB API
   2. Filter objects are provided to Post MVC Context i.e. When the ConfigureServices() loads MVC Request processing
   3. Filter Type
      1. Action Filters for any custom Logic e.g. Logging
      2. Exception Filters
      3. Result Filters 🡪 For MVC Controllers
   4. 
3. **Middlewares**
   1. Create a class that accepts **RequestDelegate** as dependency
      1. The RequestDelegate is a delegate that represent an execution of HttpContext.
      2. The HttpContext is generated means the Request is received by **dotnet.exe** and its has loaded application dll and all of its dependencies.
   2. Write an **InvokeAsync()** method, that accepts **HttpContext** as Mandatory Parameter
      1. This method contains the logic that middleware wants to execute.
      2. E.g. if the error middleware, then this method will use try..catch block, in try it will simple invoke the RequestDelegate and in catch block implement logic for handling errors. This method is responsible for writing the RESPONSE.
   3. Create an **extension class** with static method having first parameter as **this IApplicationBuilder app**
      1. In this method write following
         1. app.UseMiddleWare<MiddlewareClass>();
         2. The UseMiddleware() method will load Middlware class, inject RequestDelegate and Invoke the InvokeAsync() method for the current HttpContext
4. Security
   1. UserManager<IdentityUser>
      1. Manages CRUD Operations in IdentityUser Entity Class and Map it with AspNetUsers Table in Database
   2. RoleManager<IdentityRole>
      1. Manages CRUD Operations in IdentityRole Entity Class and Map it with AspNetRoles Table in Database
   3. SignInManager
      1. Class manages The user Authentication by accepting IdentityUser Object as parameter
   4. ClaimIdentity
      1. The class to Verify the current Identity User’s claim to verify Authentication based on Token
5. Deployment

Install-Package Swashbuckle.AspNetCore

<https://dotnet.microsoft.com/download/dotnet-core/2.2>

ASP.NET Core/.NET Core: [Runtime & Hosting Bundle](https://dotnet.microsoft.com/download/thank-you/dotnet-runtime-2.2.3-windows-hosting-bundle-installer)

For Hosting on IIS