**ASP.NET Core - Intermediate**

# 1 | ASP.NET Core Internals

## a | ASP.NET Core Internals

1. Create New Project MyMVA Web Application.
2. Set authentication to Individual user accounts (gives ability for users to login to the application. We can manage which logged in user can access which part of the application)
3. Till 20:00 is Basics of Startup class.
4. In Program.cs
   1. CreateDefault builder will load files like appsettings, loats its valie in configuration.
   2. It also loads environment variables.
5. In Home Page, index file.
   1. At top add @inject Microsoft.Extensions.Configuration.IConfiguration Configuration
      1. This makes available configuration to this page.
   2. Clear all divs.
   3. Add header tag with <h1>This is my Machine Name : @Configuration["computername"]</h1>
      1. This will find computername in configuration.
6. In appsettings.json
   1. Add Appsetting. "MyTwitterKey" : "TwitterSecretKey",
7. In Home Page, index file.
   1. Add header tag with <h1>This is my Machine Name : @Configuration["MyTwitterKey "]</h1>
8. Run and Check
9. This is in Development Env, but in Production Env we don’t want the same value.
10. We will override it with Env Variable.
11. Right click Project and select Properties, go to Debug Tab.
12. In Environment variables.
    1. The key ENVIRONMENT is set to development, if this is not set the by default it is production Env.
    2. Add New Key “MyTwitterKey” to value “TwitterSecretKeyFromEnvVar”
13. Run and check.
    1. Now we will get “MyTwitterKey” from Env Variable.
14. Appsettings.json is not the best secure way to store variables. By mistake it may be commited by developers. And it becomes public.
15. To avoid this we can use user Secrets
16. Right click Project select manage User Secret.
    1. We get a file secrets.json, this is not a encrypted file, it is not in source control.
    2. If a key exist in secret.json and in Env variable., then Env Variable will be selected.
    3. If it only exist in Scerets.json it will be picked.
17. Run and Check.
18. Now we will change log level in Env Variable.
    1. Add Value “LoggingLogLevelDefault” and key “Trace”
19. Run and check, now we will get trace level debugging. (just like in Application Insights)

# 1 | Tag Helpers

## a | Tag Helpers

1. Create New Project WebApplication24 Web Application., Set authentication to Individual user accounts.
2. In Register.cshtml
   1. The form uses tag helper, we see tag helper for div, label, input, span,etc.
3. In \_Layout.cshtml.
   1. We see tags environment, these are not HTML5 tags, these are higher level tag helpers.
      1. These generates a normal html, depending on development Env.
4. These Tag Helpers become active in \_ViewImports.
   1. @addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers
   2. \*, this is the assembly, which is pulled.
   3. Else we define Namespace.( without \*,)
5. We will Learn how to make custom Tag Helper.
6. Add New Class RepeatetagHelper , inherit it from TagHelper
7. Overide ProcessAsync Method in Parent class.
   1. Add Forloop, for Count.
   2. Add output.Content.AppendHtml(await output.GetChildContentAsync(useCachedResult: false));
8. Go to \_ViewImports and add RepeatetagHelper.
9. Add Repeat Tag to Index Page, with count 6, add p tag with content
10. Run and Check.
    1. The p tag will be repeated Count Times.
11. We will check tag helpers created by other developers, and we can use it.
    1. DamianEdwards.
    2. DavidPaquette (dpaquette)
    3. Check their Github, Examples and Source.

# 1 | Entity Framwork Core

## a | Entity Framwork Core

1. POCO : Plain Old CLR Classes