**ASP.NET Core – Beginner**

# 1 | Getting Started

## a | Getting Started with ASP.NET Core

1. From dot.net download .Net Core SDK From Downloads.
2. From docs.microsoft.com/en-us has Docs for all Mocrosoft products.
3. From visualstudio.com download Visual studio.

## b | Introduction to the .NET CLI & Templating

1. How to check if .Net Core is Installed
   1. In CommandPromt type dotnet-version , this will show version.
   2. where dotnet will show were it is installed.
   3. dotnet—help will show help.
   4. dotnet new will show all the template available, their language.
2. In CommandPrompt
   1. Goto Folder where you want to make new Project
   2. Create new folder myconsoleapp
   3. dotnet new console will create console Template
   4. dir will show contents of project folder
      1. it has myconsoleapp.csproj file and Program.cs File
   5. type myconsoleapp.csproj will show file contents
   6. dotnet run will compile and run the project
   7. code . will open the current project in VS Code
3. In VS Code
   1. Press Ctrl + P will open a pallet. Various commands can be given.
   2. In Program.cs Change Hello world, and press Ctrl + ~ to run it on a terminal in VS code
   3. Put Debug Point, Goto Debug Tab, and run, the debug point will be hit.

## c | Documentation & Community

1. Goto docs.Microsoft.com/ select .NET. this has all Documentations

# 2 | ASP.NET Web Application

## a | Console to Web Application

1. In Cmd Prompt type dotnet new web –o mywebapp , this will create folder and web project . Open it in VS Code
2. Run and Check.
3. In Program.cs
   1. Has Main Method, which calls CreateWebHostBuilder.
   2. In CreateWebHostBuilder Method
      1. CreateDefaultBuilder is called.
      2. And UseStartup<Startup>(); is called, this calls Startup Class.
4. In Startup.cs
   1. ConfigureServices Mehod, Prepares Services
   2. Configure Method is used to configure the service

## b | Dynamically Rendered HTML

1. In ConfigureServices Add services.AddMvc();, this gets the service ready for use.
2. In Configure add app.UseMvc();
3. Run And Check.
4. Add New Folder Pages to project and index.cshtml file to it.
5. Add @page and a header tag
6. Run and Check.
7. Add page01.cshtml file , Add @page and a header tag
8. Run and Check. /page01
9. Add Folder page02 in Pages, add file index.cshtml , add @page and a header tag
10. Run and check, /page02
11. Till now we had static html., lets trl dynamic html.
12. In page02 app <p> tag with @DateTime.Now

## c | Configuration

1. Here we add app settings
2. Add appsettings.json file in Proj
3. Add Message Json string.
4. Add Startup Constructor with IConfigurartion as parameter
5. Add IConfiguretion Property.In constructor assign parameter to it.
6. In index.cshtml file
   1. Add @using Microsoft.Extensions.Configuration
   2. Add @inject IConfiguration Configuration
   3. Add header @Configuration["Message"]
7. Run And Check.

# 3 | Razor Pages and CRUD

## a | Razor Pages

1. Open VS 2017
2. Add New .Net Web Core Project.
3. Check project Structure.

## b | CRUD

1. In Startup file
   1. Add using Microsoft.EntityFrameworkCore;
   2. Add to ConfigureServices Method services.AddDbContext<AppDbContext>(options => options.UseInMemoryDatabase("name"));

1. Add AppDbContext File.
   1. Add Constructor, with DbContextOptions as parameter, pass it to base class
   2. Add public DbSet<Customer> Customers { get; set; }
2. Add Customer.cs Class.
   1. Add Property Id, Name.
   2. Add Requred and Lengnt of 10.
3. In Pages Folder add razor Page Create
   1. Add Form to enter Name.
4. In Create pagemodel.
   1. Add private readonly AppDbContext \_db;
      1. Private because only I can access this context.
      2. Readonly because I want to assign it only once.
   2. Add constructor with AppDbContext as parameter, assign it to \_db property
   3. Remove OnGet Method
   4. Add OnPostAsync Method. public async Task<IActionResult> OnPostAsync()
      1. Check ModelState Validity, if not return page.
      2. Add \_db.Customers.Add(Customer); , adds posted Customer to db.
      3. Add await \_db.SaveChangesAsync(); , saves changes to db asynchronously
      4. Add return RedirectToPage("/"); , redirects to index page
5. Run and Check
6. In Create.cshtml
   1. InForm add <span asp-validation-for="Customer.Name"></span> for particular field validation
   2. <div asp-validation-summary="All"></div> for Full form Validation
7. Run and check, add string longer than , we will get error.
8. In Create.cshtml.cs in OnPostAsync Method, redirect page to “/Index”
9. Run and Check
10. In Index.cshtml.cs
    1. Add AppDbContext as private variable, constructor which takes AppDbContext and passes to private variable.
    2. Add public IList<Customer> Customers { get; private set; }
    3. Add async Method OnGetAsync
       1. Add Customers = await \_db.Customers.AsNoTracking().ToListAsync();
       2. AsNoTracking,is good for readonly scenarios.Donot track, makes it faster
11. In Index.cshtml
    1. Add html for table to show customers.
    2. Add Anchor tag to edit customer
    3. Delete button to delete customer.
    4. Add Create link.
12. In Index.cshtml.cs
    1. Add OnPostDeleteAsync method..
       1. Here find customer by id, if found Delete it, redirect to same page.
13. Add edit razor page.
    1. In cshtml file, here we take a parameter as input(Id), add @page “{id:int}”
14. In edit.cshtml.cs
    1. Intialize model with AppDbContext, constructor
15. In edit.cs
    1. Add form to Edit customer
16. In edit.cshtml.cs
    1. Write OnGetAsync method.
    2. Write OnPostAsync method
17. Run and check

# 3 | Logging and Diagnostics

## a | Logging and Diagnostics

1. In Create.cshtml.cs
   1. Add property Message., add DataAnnotatin TempData.
   2. Add Message = $"Customer {Customer.Name} added !"; to OnPostAsync Method before the return.
2. In Index.cshtml.cs
   1. Add property Message., add DataAnnotatin TempData.
3. In Index.cshtml
   1. Add <h3>@Model.Message</h3> at top of form.
4. Run and check.
5. We will get feedback that customer is created in Index.
6. In Startup.cs
   1. In Configure services method
      1. Add services.AddLogging(); adds logging service
   2. In Configure method
      1. In Parameters add ILoggerFactory loggerfactory.
      2. Add loggerFactory.AddConsole(); Send log to Console
      3. Add loggerFactory.AddDebug();
7. In Create.cshtml.cs
   1. Add private ILogger<CreateModel> \_log; and pass it to constructor and assign it to this private variable.
   2. Add \_log.LogCritical(msg); to OnPostAsync
8. Run application from command line.
   1. Go to that folder and write dotnet run.
9. In Console we will see log in red(sincs we put LogCritical), when we create a customer.
10. Rightclick Project, add Application Insights. Select add SDK locally.
11. Rightclick on tool bar, and check Application Insights.
12. Run and check
    1. We will see as we browse through app, create customer the number on the application Insight bulb increase,
    2. Double click the bulb. We will see application Insights Search window
    3. Browse and apply Filters on left side tab.