Softwares

.Net Framework – Supports websites, servers, desktop apps & more on windows (fast buildy)

.Net Core – cross platform, websites, servers & desktop apps
(modular, light weight & flexible)
→ able to use in different computers.

.Net – free, cross platform open source for web, mobile, desktop games etc

ASP.Net :– framework for web apps with .Net & C#
web apps, APIs, Realtime, Microservices
→ developer platform made up of tools, prog. lang. & libs to build web apps
– fast & scalable
– secure (XSS) – cross site scripting
(CSRF) – cross site req. forgery

ASP.Net MVC – decouple UI, (view) data (model) & Appl. logic (controller)

ASP.Net Web API – HTTP services with easy way that returns only data.

~~ASP.Net Web API MVC~~

ASP.Net ~~Web API~~ (dotnet)

→ Start up.cs — Startup & middleware configuration
↳ requests & responses

→ mywebApp/pages — contains web pages for app

→ myweb.csproj — libraries referenced

~~dotnet~~ .

```
dotnet new webapp -o mywebApp --no-https
~~dotnet~~ dotnet watch run
```

Razor pages

Software development framework

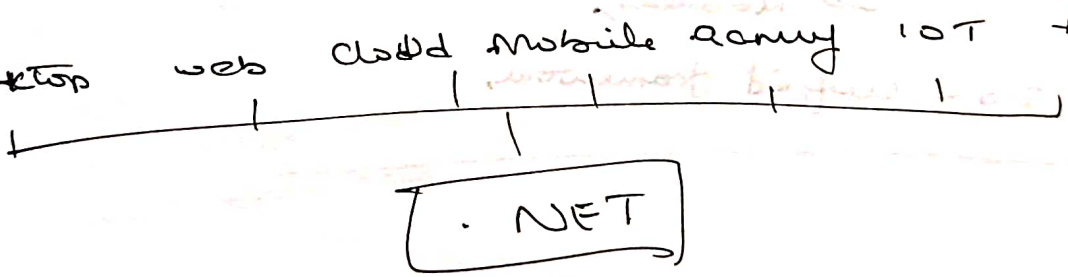
→ foundation for developers to build software applications
→ It includes libraries, packages, compiler etc
→ It improves efficiency of working on the important details of app instead of working on the fundamental parts.

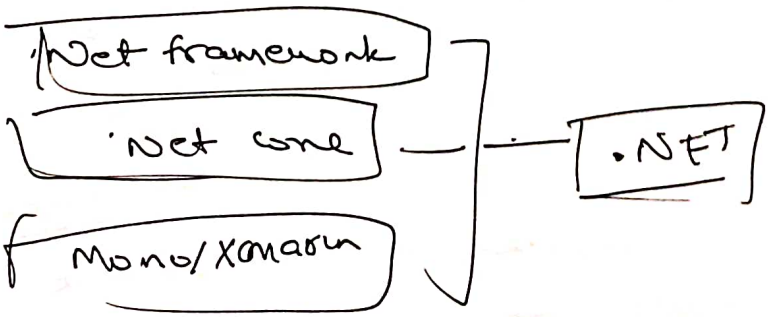
.Net

→ is a software framework that runs primarily on Microsoft windows
→ has a large library that supports interoperable (ability to exchange))
→ access several languages., platform to build.

→ Brief

.Net → the ecosystem — collection of diff software projects
CLR → JVM
C#, F#, VB → Languages
Nuget → npm etc
dotnet cli → entry pt, sdk, javac etc
&
Desktop web cloud Mobile gaming IOT AI





Ecosystem —— Languages — C#, F#, VB
runtime — CLR
libraries - Base Class lib, etc

.dll - dynamic link library
.exe - executable

.Net 5.0 — follow up .Net Core 3.1 & added functionality
— & removed core part

TFM — Target framework Moniker

C# — 9.0

---

dotnet — package manager

.Net framework → windows appl's.
→ WPF, ASP.Net forms,

.Net Core → cross platform software framework
→ Modular,

.Net 5.0 — unified framework

~~MVC~~ — MVVM (Model-View-ViewModel)

~~Startup.cs~~

Razor Pages — for simple pages to receive & display data.

Startup.cs

Startup
- Configuration
- Configure (app, env)
  - if development
  - Https redirect
  - use static files
  - use Routing
  - use Authorization
  - use Endpoints

@Html

@page
<...>
<... @Model ...>

filename.cshtml

@page
@using namespace_name
@model model_name
<>
< >
Model - .
</>

filename.cshtml.cs

namespace namespace_name {
public class model_name : PageModel
}

.cshtml – html markup & c# code with Razor Syntax

.cshtml.cs – c# code for page events

wwwroot – contains assets like HTML, JS, & CSS files

appsettings.json – configdata & connection strings

Program.cs – entrypoint for the system

Startup.cs – code to configure app behaviour.

→ app settings.json

→ Program.cs

→ Startup.cs

---

appsetting.json

→ logging

↳ log level

↳ default – ~~microsoft~~ information

↳ microsoft – warning

↳ .

– AllowedHosts

– ConnectionStrings: {

—— context: ——

}

Program.cs (AspNetCore, .Hosting, Extensions)

↳ CreateHostBuilder(args)

→ IHostBuilder

~~IHostBu~~ ~~Host.CreateDefault~~

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

Startup.cs

{ AspNetCore, .Builder, .Hosting, .Extension, .configuration

Dependency Injection, Hosting, Config, Framework Core,

→ Configuration

→ ConfigureServices

→ AddDbContext

→ Configure (AppBuilder App, IWebHostEnv)

Program

→ ~~Host~~.CreateDefaultBuilder(args).
.ConfigureWebHostDefaults(webBuilder =>

→ CreateDefaultBuilder
→ Chained ConfigurationProvider – source to app config.
[appconfig] - custom prop & behaviour of App.
→ ~~appsettings.json~~ → Configuration value
→ Content root – gets current directory
↳ env. variable & cli arguments
↳ loads app config from appsettings.json.
↳ logging provider
↳ console
↳ debug
↳ event source
↳ event log

ConfigureWebHostDefaults
↳ host config
↳ kestrel
↳ adds middleware
↳ IIS Integration

Framework provided Services

→ IHost Appl^n lifetime — logging
→ IHost lifetime
→ IHostEnvironment — appname, env name, content root path

Host Configuration

Configure HostConfiguration on IHostBuilder

ASPNETCORE_ENVIRONMENT ⇒ environment

ConfigureHostConfiguration (config Host ⇒)

→ SetBasePath
→ addjsonfile (Hostsettings)
→ Add Env variable
→ add command line

App config

AppName — key, type, default (entry point), Env variable —.

Content root — key, type, set value, Default - app assembly

Env name — key, type default: production, Env. Env var

WebApps

CaptureStartupErrors
UseStartup <Startup>.()

URLS

← UseURLs (———)

webroot

.UseWebRoot ("public")

(Web API)

→ UseDeveloperExceptionPage

→ use Swagger

→ addSwaggerGen — Author, licence, descript→

→ use HttpsRedirect

→ use Routing

→ use Authorization

→ use Endpoints

Dot fold Configuration

① Connected Services

→ lets app. developers connect their apps to service providers that runs in the cloud

② Dependencies

(i) Analyzers - find potential issues & fix them (Suggests)

(a) @ Microsoft. ASPNetCore.

→ Analyzers - Nuget package

- Analyze the errors generated in the code editing.

→ Component Analyzers - Roslyn analyzers

- for quality, style, maintainability, design & other issues

→ mvc.analyzers.

- powerful, pattern based dynamic websites

(b) Microsoft. CodeAnalysis. CSharp. NetAnalyzers &

- inspect C# code
- security, performance & design & other issues

Eg platform compatibility errors

(ii) Frameworks

- the dependencies of code.

Eg System, System. IEnumerable etc

③ Properties..

- provides with project configuration

launchSettings.json

→ used only on local development machine

→ not deployed

→ contains profile settings

Default : IIS Express - webServer

↳ light weight, works well with http & .NetCore projects

④ JSON in ~~.net~~ ASP.Net

→ Global.json - applies to all projects in solution

- (project list & sdk version)

- src - actual app^. & test for testing

→

→ appsettings.json — define app related settings like connection string & logging settings or key for web.config

→ project.json - project setting & server side dependencies

→ Launchsettings.json - project specific setting associated with each profile. VS configure to launch the application (IIS, Kestrel, weblistener)

→ bower.json - maintain & manage packages of HTML, CSS etc

→ package.json —

⑤ wwwroot. — web root folder

— serves static files

— ~~ASP.NET~~ .AspNetCore.StaticFiles

⑥ Pages — serve as path for pages

↳ underscore will not work if app name

— they are not meant to shown directly as a path.

- not intended to be browsed.

- @page prefix

⑦ Program.

CreateHostBuilder(args).Build().Run(); → initialize the host

Host.CreateDefaultBuilder(args).

ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

IHostBuilder — properties

— central location for sharing state b/w components during host startup

IDictionary<object, object> Properties;

→ Create Default Builder.

① Set contentRoot path
② Load IConfiguration from DOTNET_ environment var
③ Set IConfig from args (Command line)
④ IConfig from appsettings.json & app.env.json
⑤ IConfig - user secrets
⑥ IConfig - env vars
⑦ Load app - app from command line args
⑧ ILoggerFactory config (Console, debug, EventSource, EventLog)
⑨ Scope validation on DI config when env is Development

→ IConfiguration:

IConfiguration[Song]

Configuration
represents a configuration file that is applicable to a particular computer, app or resource.

→ ConfigureWebHostDefaults — properties (obj, Obj> c)

Configures with IHostBuilder with defaults for hosting a web app.

ConfigureAppConfiguration()
Startup()
Build()

→ Extension methods - without creating derived type

add methods

Host Config (ASPNETCORE_), Kestrel,
HostFiltering Middleware, ForwardedHeaders middleware)
Environment (IHostEnvironment)

~~IHostWebBuilder~~ IWebHostBuilder – Build configure app, services etc

~~webBuilder.UseStartup~~ (– A Builder for IWebHost

~~IWebHostBuilder~~ IWebHost – represent a configured web host

↳ server features – collection of feature by server

↳ services – IServiceProvider

UseStartup – IWebHostBuilder & strings for assembly name of Startup)

- Specify the assembly containing the startup type to be used by webHost

IHostBuilder

~~Startup~~. Build() – actions to initialize the Host. – return IHost

IHost – IDisposable(– remove ~~unmanaged~~ managed resources)

– Services – program config & services

Run() — Runs an application & block the the calling thread until host shutdown

② Startup.

Configuration ← configuration (host config
app config
logging provider
scope & dependency validation)

loads the services

IServiceCollection – count, IsReadOnly, Item

MVC service collection Extensions.

Builder has services

→ Configure

→ IApplicationBuilder – Defines a class that provides the mechanism to configure an application pipeline.

→ useHttpRedirection — adds middleware for redirecting HTTP to HTTPS

→ useStaticFiles – Enables static file serving for current request path

→ useRouting – adds a RoutingEndpoint middleware. (match endpoints

→ useAuthorization – adds Authorization middleware to specified app(Builder & enables Authorization capabilities

btw — app.useRouting() & app.useEndpoints())

→ executable route endpoint

→ useEndpoints – Endpoint middleware. to specify App's route and Endpoint datasource instances. built from configured IEndpointRouteBuilder. the epoint middleware executes the request

IEndPointRouteBuilder – DataSources – ICollection<EndpointDataSource> endpoint Data Sources config in Route

– ServiceProvider – get the set the IServiceProvider used to resolve services for routes

Endpoint – Name, MetaData, Request Delegate

Name / signature display name

MetaData / Collection metadata

Request Delegate delegate used to execute

EndpointMetadataCollection - list of metadata the - of endpoints types added in order.

- builds empty..

Endpoint

builder - to add middleware &

action<T> - configure to provide IEndpointMetadata

Endpoint

Map get method to define the endpoint.

→ selected by matching the url

→ executed by running the delegate