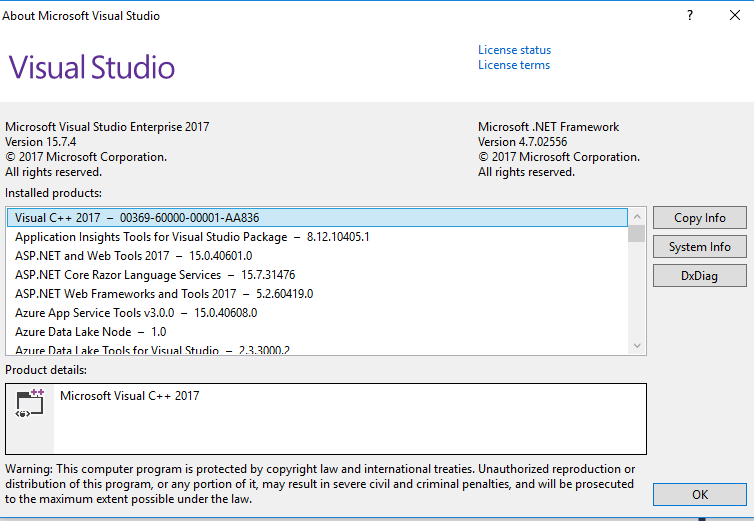
**Asp.net\_Core:**

****

.statndard framework

Base class library ()

CLR-Common Language Runtime,CTS-Common Type system,CLS-Common Language specification

Security- using assembly given strong name,in application security-using authentication/authorization

Disadvantages:

* Framework is heavy( base class Library-large set of assmbiles).
* It is not platform independent- ex: if create application it will work only windows not any other operating systems.
* Framework is free but it is not open source. –it will be developed by Microsoft/partner only same time if we taken open source we can get the libraries and make some changes and we can use into our application this is not possible in .net framework.
* Microservices not supported (open sources supports using node.js..etc also
* Advantages- low cost, easy deployment at any os, also support containerization)
* If you want to create any microservices

Asp.net core- support containerization,CLI tool like create react app, open source using github- we can download,large set of libraries,support Microservices.

Supports cloud so we can deploy application in azure cloud.

**Asp.net\_core\_components:**

* Base class library
* CLR
* Same garbage collector also same core CLR but engine modified.
* Console application,Window, wcf, wpf, mobile app, Asp.net Core application like web application (MVC application,Web API,Razor app,Angular SPA app, ReactJS SPA app), universal windows application,service application(micro services),

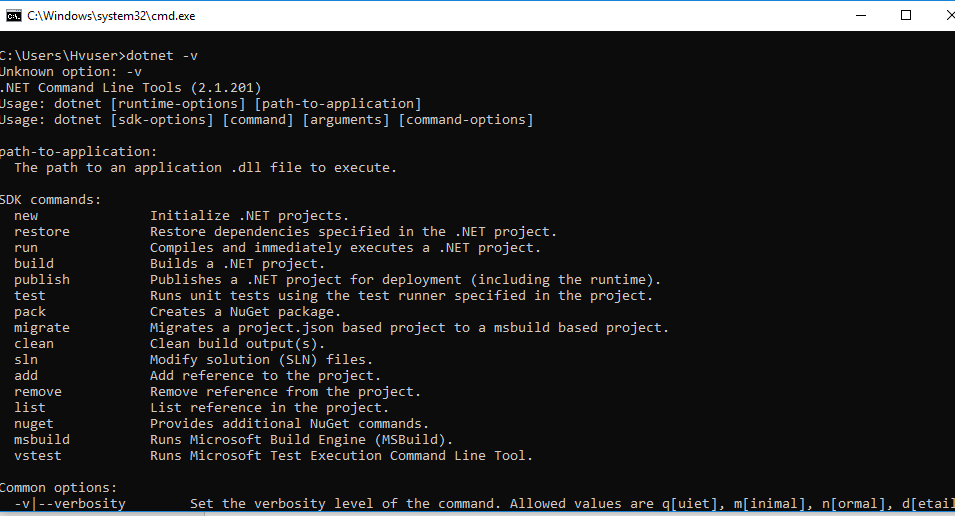
Difference Asp.net /Asp.net Core

.net – multiple config file we can use.

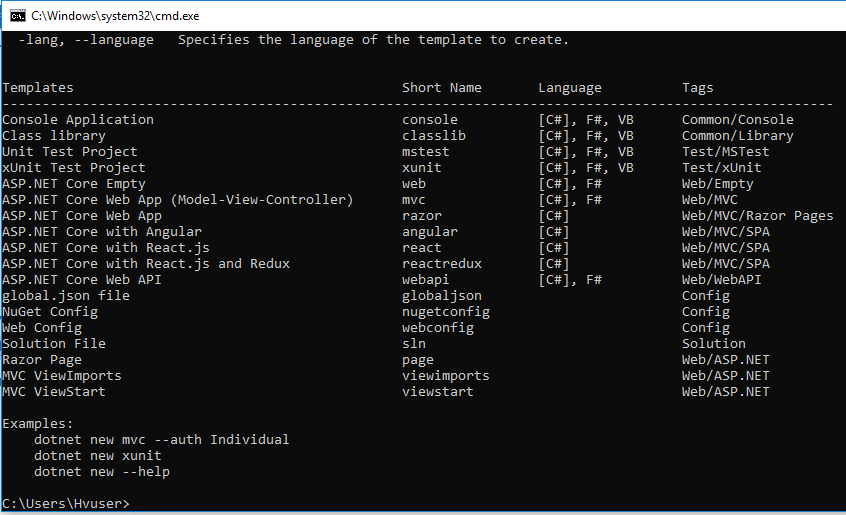
1. Different sources we can read the configuration🡪Through environment variables. You can read the arg from CMD, ini, Json, XML, Inmomory objects, Database.
2. Build in dependency Injection container- if you don’t want to use also you can replace with others(unity/inject third party containers).
3. Hosting- typically we can host web application in IIS in Asp.net but asp.net core we can host in self hosting , differrnt environment/servers,WebPack,wep Api,apache tomcat,nginx.
4. Start up file- where you can config your services using middleware (whatever using module in (http module here called middleware) you can create own middleware/built in middleware)
5. Some new templates(Razor pages we can use for MVC app)

Hosting diff,Building,config.

Need to install .net core 2.1.400🡪 check dotnet –v (to check in cmd version of .netcore)



Where ever you want to create folder and moved into project folder.



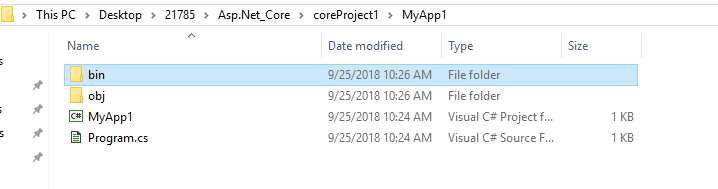
Use short name🡪 you can

dotnet new console -o MyApp1

cd MyApp

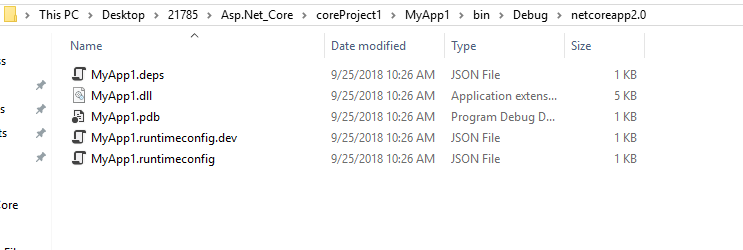
dotnet restore

dotnet build –to compile project

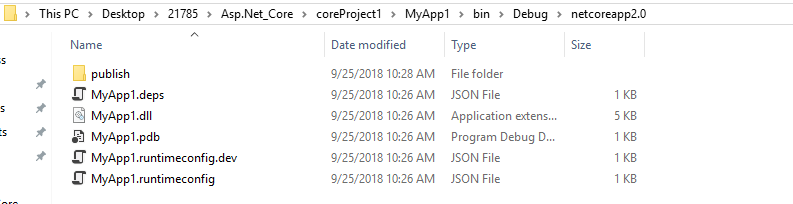


dotnet run 🡪 to run the application in CLI





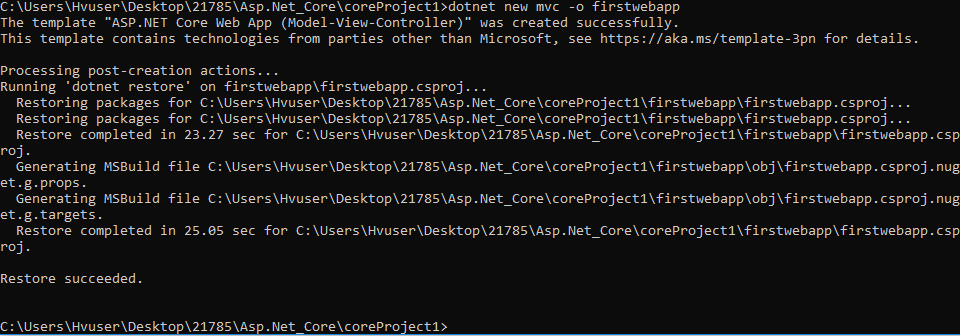
dotnet publish -- > create publish directory



Code .

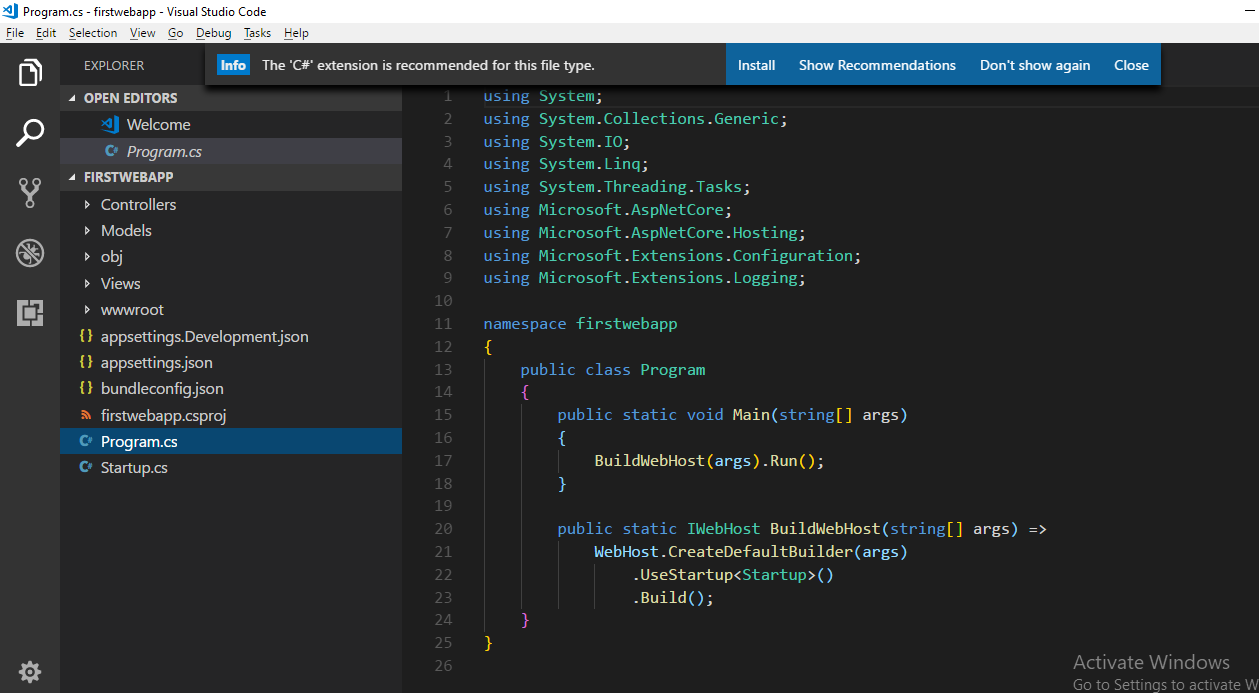


dotnet new mvc -o firstwebapp



Cd firstwebapp

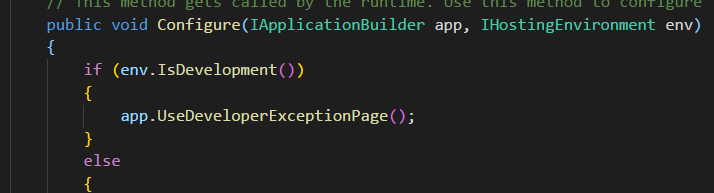
Then open into Visual studio code using open folder

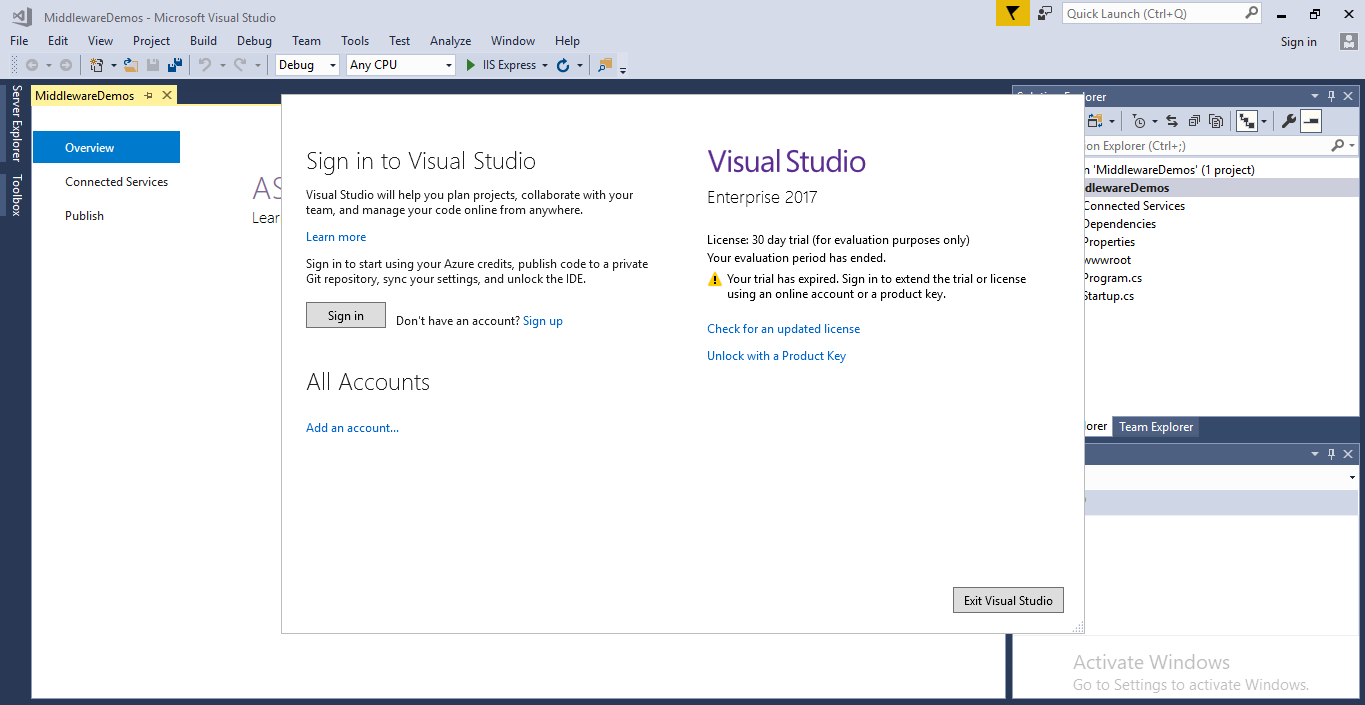


Startup-classs🡪taking configuration file mentioned middleware, it contained 2 main method.

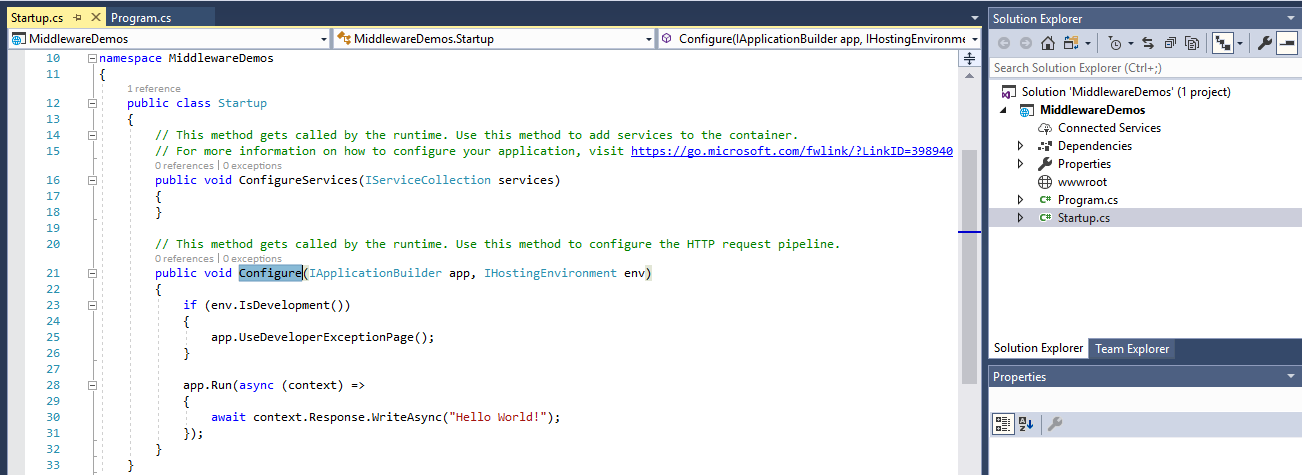
1. ConfigureServices()🡪1 servers input
2. Configure() 🡪2 parameter input additional also allowed. Here we can configure middleware.there is an environment variable default aspnetcore\_environment variable🡪 by default🡪IsDevelopmnet/Produciton/Staging we can give custom value also but that type need to check explicity needs to check

These 2 configure method execute only once in the application lifecycle.



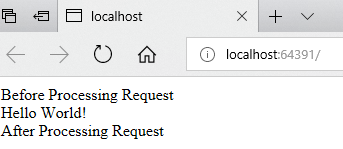


Login through office365 account/outlook /Hotmail acc.



* Three methods usually to configure the middleware

1. Use() – Use to configure take a set of argument to the req & response middleware use this.
2. Map() – different set of URL at that time you need to use Map only for diverting the flow ex: app.map(‘./about’,myDelegates)
3. Run() – To shotcircut the request flow Like to terminate the request flow then send the response back
4. public void Configure(IApplicationBuilder app, IHostingEnvironment env)
5. {
6. if (env.IsDevelopment())
7. {
8. app.UseDeveloperExceptionPage();
9. }
10. app.Use(async (context, next) =>
11. {
12. var contenttype = new Microsoft.Extensions.Primitives.StringValues("text/html");
13. context.Response.Headers.Add("content-Type",contenttype);
14. await next.Invoke();
15. });
16. app.Use(async (context, next) =>
17. {
18. await context.Response.WriteAsync("Before Processing Request<br/>");
19. await next.Invoke();
20. await context.Response.WriteAsync("After Processing Request<br/>");
21. });
22. app.Run(async (context) =>
23. {
24. await context.Response.WriteAsync("Hello World!<br/>");
25. });
26. }



Using map in configure

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.Use(async (context, next) =>

{

var contenttype = new Microsoft.Extensions.Primitives.StringValues("text/html");

context.Response.Headers.Add("content-Type",contenttype);

await next.Invoke();

});

app.Use(async (context, next) =>

{

await context.Response.WriteAsync("Before Processing Request<br/>");

await next.Invoke();

await context.Response.WriteAsync("After Processing Request<br/>");

});

app.Map("/about",MyRequestHandler);

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World!<br/>");

});

}

private void MyRequestHandler(IApplicationBuilder app)

{

app.Use(async(context,next) =>

{

await context.Response.WriteAsync("Before about<br/>");

await next.Invoke();

await context.Response.WriteAsync("After about<br/>");

});

app.Run(async(context) =>

{

await context.Response.WriteAsync("This is about handler");

});

}

// Extension method used to add the middleware to the HTTP request pipeline.

public static class DemoMiddlewareExtensions

{

public static IApplicationBuilder UseDemoMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DemoMiddleware>();

}

}

//app.UseMiddleware<DemoMiddleware>();

app.UseDemoMiddleware(); //after extension like above

below Example:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

namespace MiddlewareDemos.Middleware

{

// You may need to install the Microsoft.AspNetCore.Http.Abstractions package into your project

public class DemoMiddleware

{

private RequestDelegate \_next;

public DemoMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task Invoke(HttpContext context)

{

if (context.Request.Path == "/report")

{

if (context.Request.Protocol == "https")

{

await context.Response.WriteAsync("Accessing using HTTPS<br/>");

await \_next.Invoke(context);

}

else

{

await context.Response.WriteAsync("Cannot be accessed using HTTP, use HTTPS<br/>");

}

}

else

{

await \_next.Invoke(context);

}

}

}

// Extension method used to add the middleware to the HTTP request pipeline.

public static class DemoMiddlewareExtensions

{

public static IApplicationBuilder UseDemoMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DemoMiddleware>();

}

}

}

Calling in Startup Page inside theConfigure funciton

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.Use(async (context, next) =>

{

var contenttype = new Microsoft.Extensions.Primitives.StringValues("text/html");

context.Response.Headers.Add("content-Type",contenttype);

await next.Invoke();

});

app.Use(async (context, next) =>

{

await context.Response.WriteAsync("Before Processing Request<br/>");

await next.Invoke();

await context.Response.WriteAsync("After Processing Request<br/>");

});

app.Map("/about", AboutRequestHandler);

//app.UseMiddleware<DemoMiddleware>();

**app.UseDemoMiddleware();**

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World!<br/>");

});

}

How builtin MVC Middleware

Need to Add startup Page inside configure services function

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.Use(async (context, next) =>

{

var contenttype = new Microsoft.Extensions.Primitives.StringValues("text/html");

context.Response.Headers.Add("content-Type",contenttype);

await next.Invoke();

});

//app.Use(async (context, next) =>

//{

// await context.Response.WriteAsync("Before Processing Request<br/>");

// await next.Invoke();

// await context.Response.WriteAsync("After Processing Request<br/>");

//});

//app.UseMiddleware<DemoMiddleware>();

app.UseDemoMiddleware();

app.Map("/about", AboutRequestHandler);

//Configure MVC here

app.UseMvc(options=> {

options.MapRoute(

name:"Default",

template:"{Controller=Home}/{Action=Index}/{Id}"

);

});

app.Run(async (context) =>

{

await context.Response.WriteAsync("Hello World!<br/>");

});

}

Old mvc and new mvc

Controller /mvc and api same

Actionresult 🡪 I ActionResult in new MVC

Based on return type only will get diff it take from api/controller.

IOC- Inversion of control

Ioc is pettern and DI is a method to implement

Loosly couples modules

DI- is build in component

ZTe

DI-create interface class,implement calss,,

services.AddSingleton<IDataService,DataService>(); //any no.of req. time for same object user adn same instaces

services.AddTransient<IDataService, DataService>(); //same req itselfCreate different instances within that services

services.AddScoped <IDataService, DataService>(); //different req. but same object next time vist it changes but common for both.

AutoFac—Needs to install

Autofac.Extensions.DependencyInjection - Needs to install

Alexinea.Autofac.Extensions.DependencyInjection - Needs to install

26-09-2019:

Startup

Middleware

DI-Dependency Injection

Today🡪

.Net Configuration will do following method to store configure information.

1. Key Vault, INI, XML, JSON – File Provider
2. In memory
3. EF Database Source
4. Environment Variables
5. Command Line Arguments

In Source file 🡪 from Program.CS 🡪

.Net core 2.1 onwards ConfigureAppConfguration(context,config)—it accepts lambda expression.

Config.AddJsonFile(….)

.AddInmemeory(…)

.AddXMLFile(…)

.AddCommandArgs(…)

Old Core 1.0

Var config = new configurationBuider() //this is needs to be create instance

Var configuration=config.Build();

host.Configure(configuration);

Startup

ConfigurationService () 🡪 here need to configure services

Configure() 🡪 need to call middleware

getValue()

getSection()

getChildern()

configuration object you needs to register as a service of Iconfiguration

sc query dotnetcoerweb

sc start

C:\Users\Hvuser\source\repos\configureDemo2.01\configureDemo2.01

sc.exe create netcore21 binPath= "C:\Users\Hvuser\source\repos\configureDemo2.01\configureDemo2.01\bin\Debug\netcoreapp2.1\win10-x64\configureDemo21.exe"

sc query netcore21

sc start netcore21

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().RunAsService();

}

In Program.cs -- > main function needs to be set as **“RunAsServices”**

CreateWebHostBuilder(args).Build().**RunAsService();**

It will be working as a service otherwise that will be kestrel.

**Steps After create Asp.net Core 2.1.0 applcation selfHosting🡪 Need to be publish aslo config in command prompt as Adminstrator mode** ---------

sc create <name> binPath =” path to .exe”

sc start <name>

sc query <name>

sc stop <name>

sc delete <name>

**Asp.net MVC core**

Views

Home

Shared

Layout

ViewStart.cshtml

ViewImports.cshtml (new in.net core contains all import statement.)

Std.html does not have action and controller.

Ex:

@using(Html.BeginForm(“Grade”,”blog”,”Post”))

@Html.LabelFor(x=>Name)

<form asp-action=”Create” asp\_control=”Blog” method=”Post>

<label asp\_for=”FirstName”></Label>

Old

<label for=”Name”>Fitst Name</Label>

Whatever coming in asp**- is** called Tag helper.

<**environment** include=”Development”>

<Link href=”~/local/loda..” >

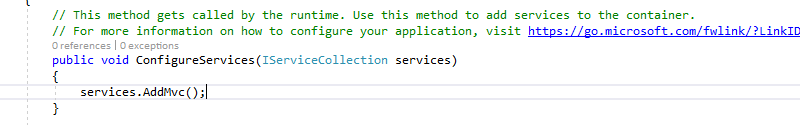
</**environment**>

<**environment** exclude=”Development”>

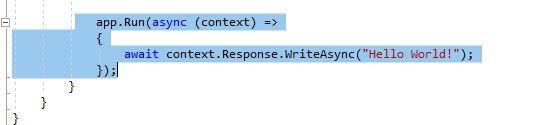
<Link href=”~/local/loda..” >

</**environment**>

Now mostly html helper class now converted into TagHelper.

.

Remove this



Add this

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

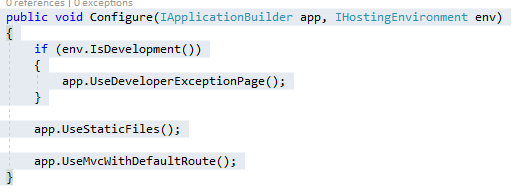
app.UseDeveloperExceptionPage();

}

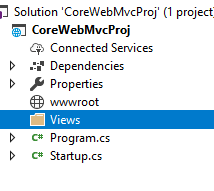
app.UseStaticFiles();

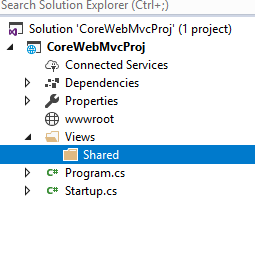
app.UseMvcWithDefaultRoute();

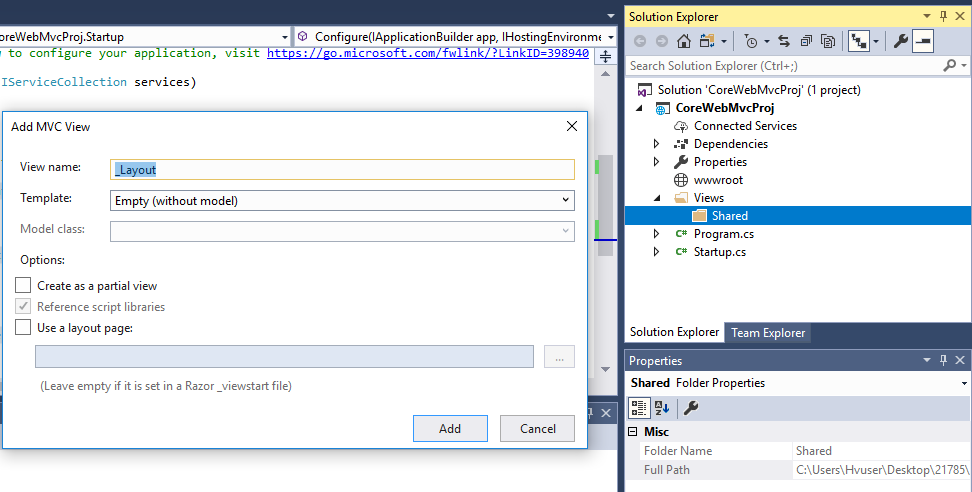
}

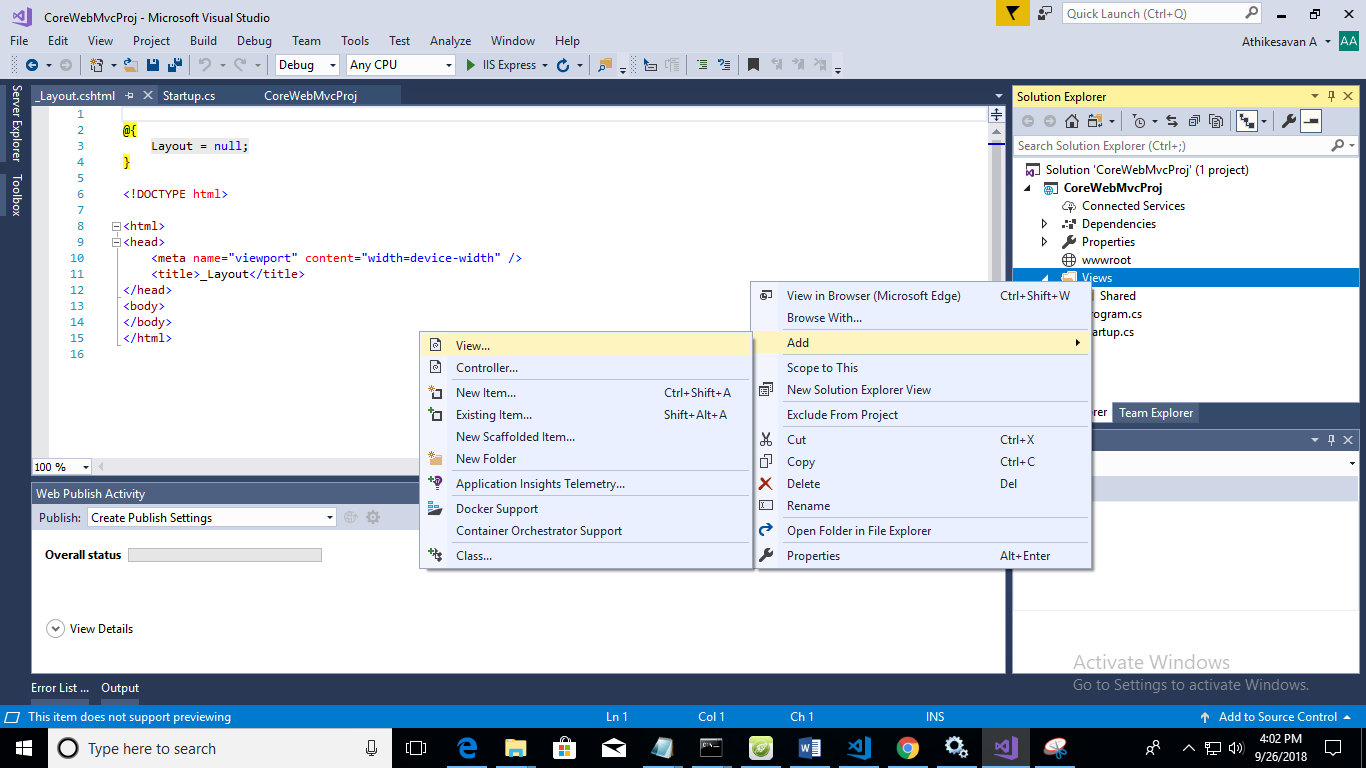


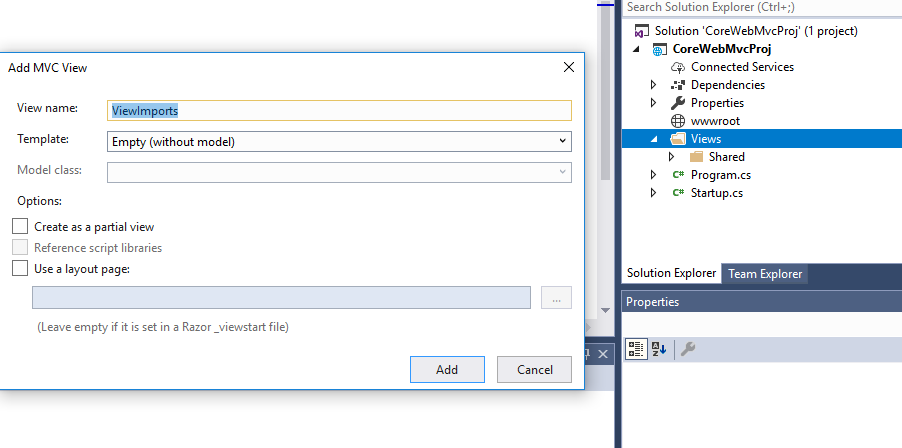
Always Static file configuration given abve only

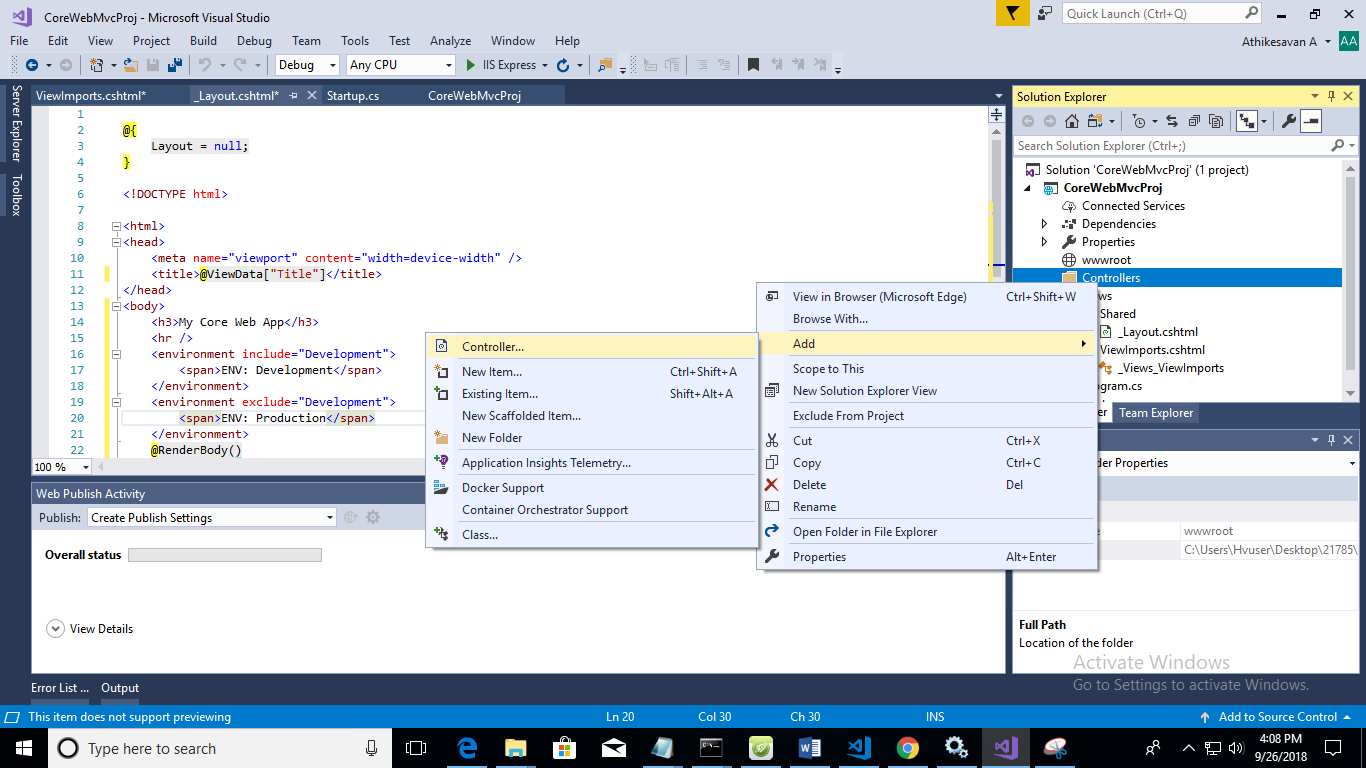






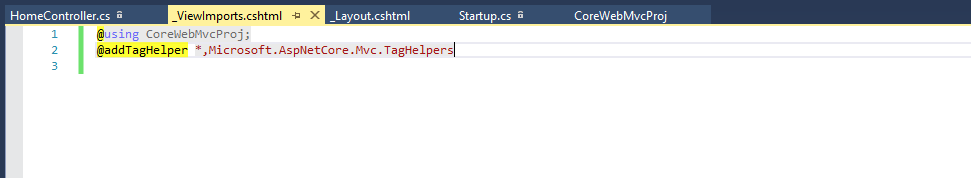


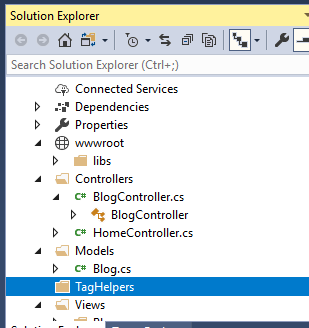




BootStrap 3.3.7

Jquery

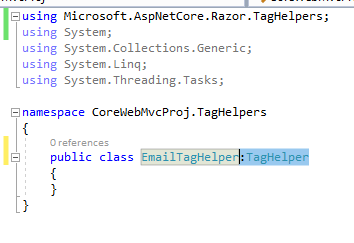




Add new Taghelper.cs class file

Them import taghelper

using Microsoft.AspNetCore.Razor.TagHelpers;



namespace CoreWebMvcProj.TagHelpers

{

[HtmlTargetElement("email",TagStructure =TagStructure.NormalOrSelfClosing)]

public class EmailTagHelper:TagHelper

{

public override void Process(TagHelperContext context, TagHelperOutput output)

{

output.TagName = "a";

output.Attributes.SetAttribute("href", "@ch.com");

output.Content.SetContent("Click me");

}

}

}



Cookies -do not store

Session

TempData

Querystring

HiddenControl

caching

DependencyInjection through Singleton

HttpContext.Items

Cookies -do not store sensitive data,1 cookie can have 4 mb size max as standard to all browser, more cookies as large amt of data browser will be slow.

Session – you have to enable a service.Also needs to be enable distribured Storege. cookieless session is not allowed here because session id is stored in cookies, default time is 20 mins if you want you can change it the service.it will integer & String only using json object we can serialize method convert into Json string and using setSession to store this.

TempData

Querystring

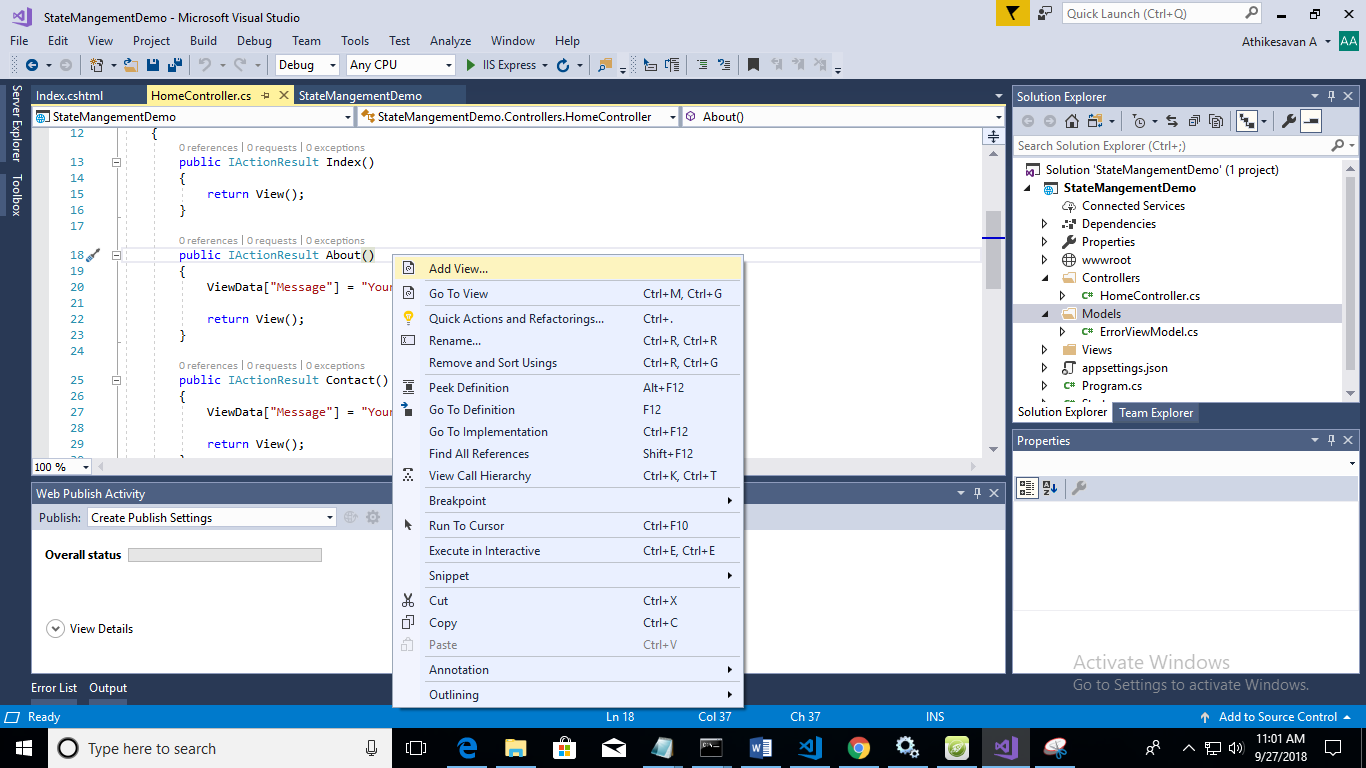
HiddenControl

caching

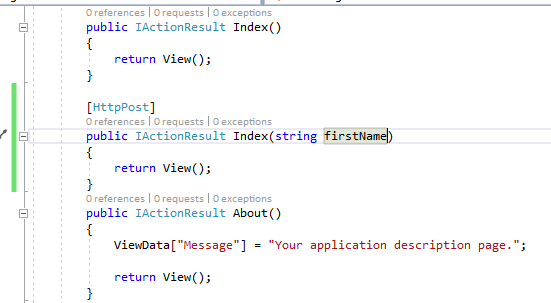
DependencyInjection through Singleton

HttpContext.Items

Now are going to work how to use Cookie and session.







**Asdf**

**Asdf**

Sd

Add this into controller

[HttpPost]

public IActionResult Index(string firstName)

{

Response.Cookies.Append("myname", firstName, new CookieOptions

{

MaxAge=TimeSpan.FromMinutes(100),

});

return View();

}

public IActionResult About()

{

ViewData["username"] = Request.Cookies["myname"];

Response.Cookies.Delete("myname");

return View();

}

Startup

public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => false;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => false;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddDistributedMemoryCache();

services.AddSession(options =>

{

options.Cookie = new CookieBuilder

{

Name = "SESSION\_COOKIE",

Expiration = TimeSpan.FromMinutes(10)

};

options.IdleTimeout = TimeSpan.FromMinutes(2);

});

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCookiePolicy();

app.UseSession();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

}

CAONTROLLER

[HttpPost]

public IActionResult Index(string firstName)

{

//Cookies

//Response.Cookies.Append("myname", firstName, new CookieOptions

//{

// MaxAge=TimeSpan.FromMinutes(100),

//});

//Session

HttpContext.Session.SetString("myname",firstName);

return View();

}

public IActionResult About()

{

//ViewData["username"] = Request.Cookies["myname"];

//Response.Cookies.Delete("myname");

ViewData["username"] = HttpContext.Session.GetString("myname");

HttpContext.Session.Clear();

return View();

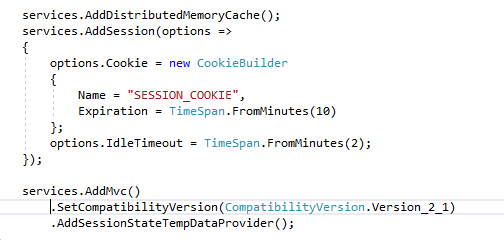
}

TempData/Caching/DI

TempData🡪its similar than mvc – diff. is TempDataProvider.

Session based TempData

For session based need to configure in Startup page as below inside ConfigureServices ()



public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => false;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddDistributedMemoryCache();

services.AddSession(options =>

{

options.Cookie = new CookieBuilder

{

Name = "SESSION\_COOKIE",

Expiration = TimeSpan.FromMinutes(10)

};

options.IdleTimeout = TimeSpan.FromMinutes(2);

});

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1)

.AddSessionStateTempDataProvider();

}

DI-Create singleton object for this needs to be create –Its not ause

Caching: 🡪 inside controller Response caching.

In memory caching-Absolute expiry, Sliding Expiry

Distributed Cache—multiple instance application is thare that time use this, using redes,Sql

Response Cache 🡪 implement in 2 ways either attribure(like output cache use response cache here—attribure cache can be used individual page also.),Response caching Middle ware used for entire application.

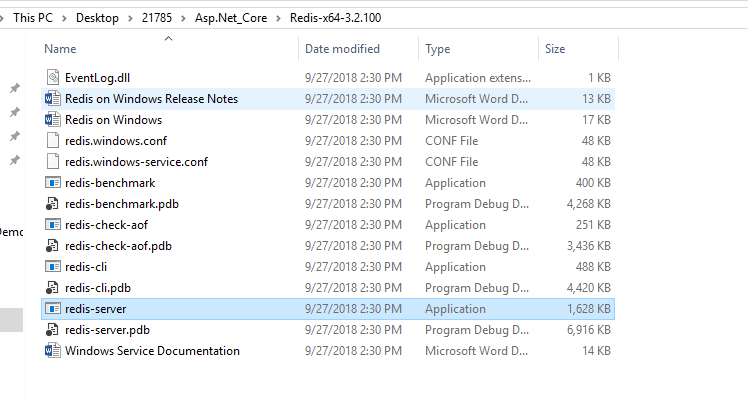
For caching Redis we need to install this extendsion from Nuget package manager.

Microsoft.Extensions.Caching.Redis

And Also download🡪 for Caching redis

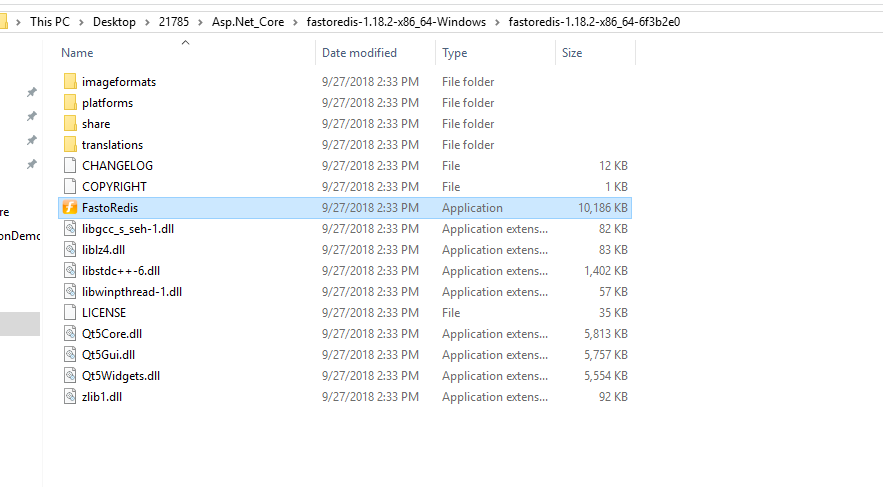
<https://github.com/MicrosoftArchive/redis/releases>

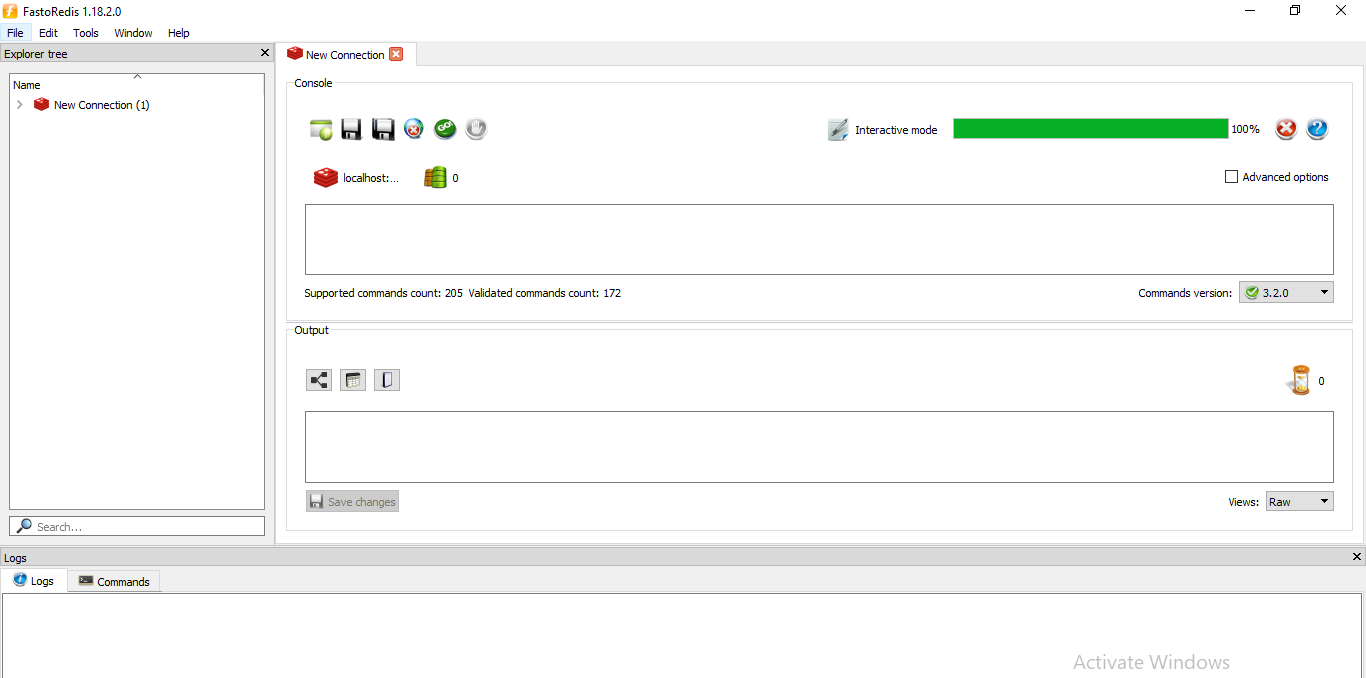




For UI need to download otherthing

<https://fastoredis.com/anonim_users_downloads>





Then go to visual studio

Startup pages

Inside configservices

//services.AddDistributedMemoryCache();

services.AddMemoryCache();//Imemory Cache

services.AddDistributedRedisCache(options =>

{

options.InstanceName = Configuration.GetValue<string>("redis:name");

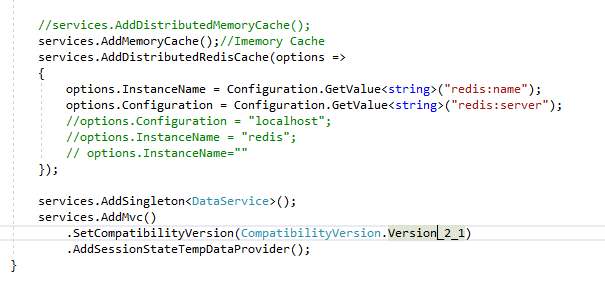
options.Configuration = Configuration.GetValue<string>("redis:server");

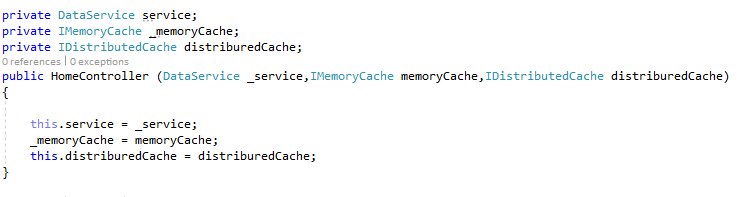
//options.Configuration = "localhost";

//options.InstanceName = "redis";

// options.InstanceName=""

});





[HttpPost]

public IActionResult Index(string firstName)

{

//Cookies

//Response.Cookies.Append("myname", firstName, new CookieOptions

//{

// MaxAge=TimeSpan.FromMinutes(10),

//});

////Session

//HttpContext.Session.SetString("myname",firstName);

//HttpContext.Session.CommitAsync();

//Tempdata

//TempData["myname"] = firstName;

//TempData.Keep();

//DI

//service.SetName(firstName);

//Memory Caching

//MemoryCacheEntryOptions options = new MemoryCacheEntryOptions()

//{

// SlidingExpiration = TimeSpan.FromMinutes(10)

//};

//\_memoryCache.Set("myname", firstName);

//Distributed redis

DistributedCacheEntryOptions distributedOptions = new DistributedCacheEntryOptions()

{

SlidingExpiration = TimeSpan.FromMinutes(10)

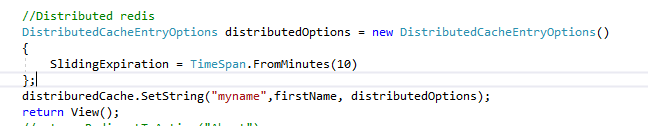
};

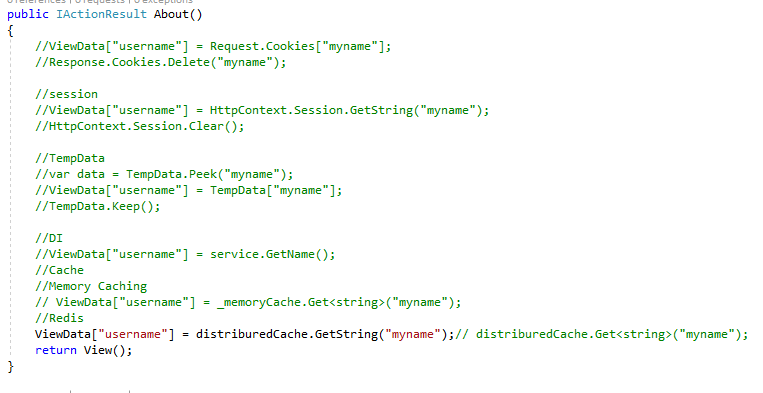
distriburedCache.SetString("myname",firstName, distributedOptions);

return View();

//return RedirectToAction("About");

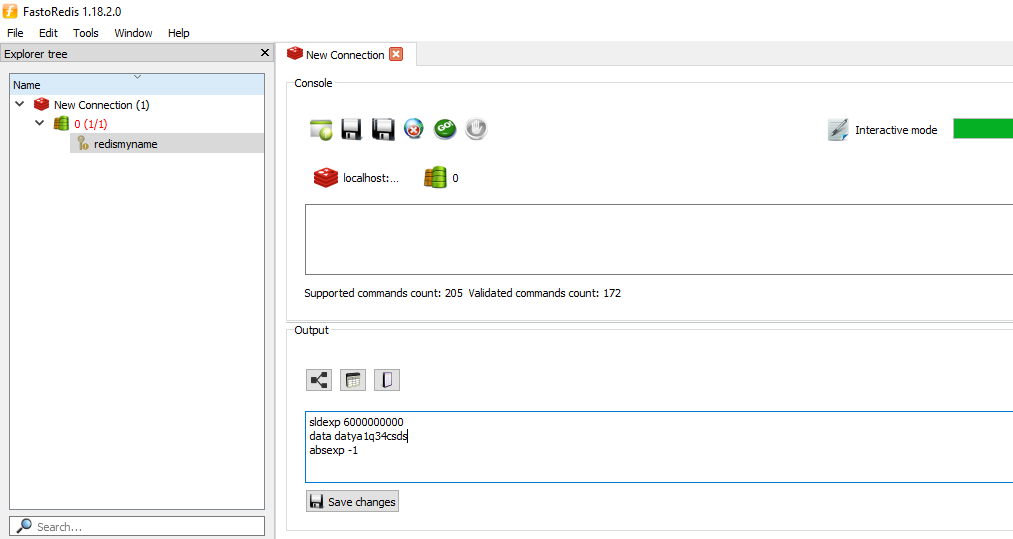
}





These are all nneds to be done in homecontroller./.





Statemanagement over

Entity Framework

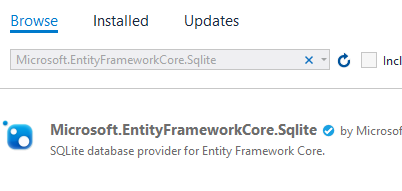
dotnet ef migration add<name>

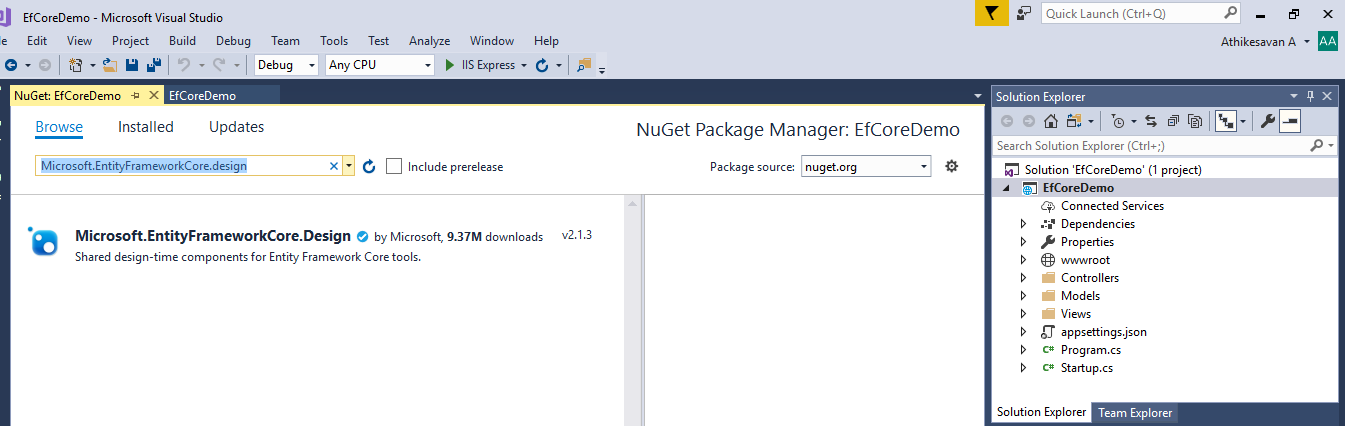
dotnet ef update database

addDbContext

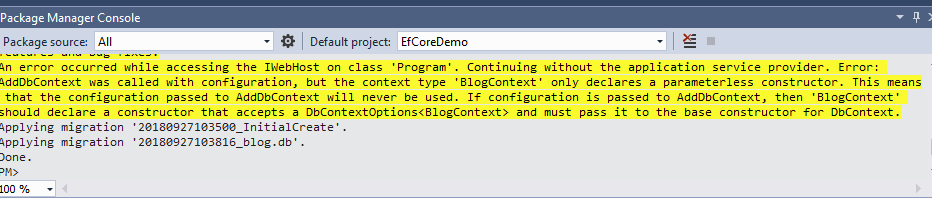
For this 2 packages needs to be added

Using this create – model class🡪db contextclass🡪create Database.





dotnet ef migrations add "InitialCreate"



**Model**

Model 🡪

1. Binding
   1. Binding- Attributes using this we can bind data
      1. [bind]
      2. [FromBody]
      3. [FromRoute]
      4. [FromQuery]
      5. [FromService]
   2. Body
   3. RouteParameter
      1. /blogs/edit/12
      2. {controller}/{action}/{id}
   4. Querystring
      1. /blogs/list?from=<date>& to=<date>
      2. {controller}/{edit}/{from}/{id}
      3. Blogs/list/12-15-18//15-5-18

Ex:

Public IActionResult Edit([FromRoute] int id,)

([FromBody] Blog Model,)

([FromService] IBlogService svc,)

2.Validation

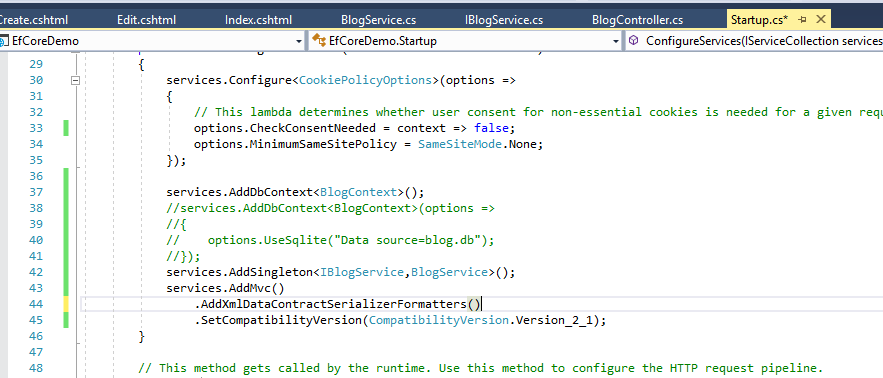
1. Data Annotation
2. Server Side Code
3. Self Validation🡪IValidatableObject (wherever you use this code it reduce repetable actions to validate like email domail as not to be public only company email allowed instead of wirting multiple code you can do once in Model using this interface.)
4. Client Validation 🡪 ex:data-minlength,date-required
5. Remote Validation.-->It will works as client side validation,whenever type values immediately it will

For XML you need to add in Startup under Configserverice

services.AddMvc()

.AddXmlDataContractSerializerFormatters()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);



Asp.net

MVC || WebApi 🡪ServiceOriented Applciation

Razor Page || SPA

App

Web

Ma

RESTfull Server🡪creatr obj for rest and call using function in normall way.

RESTfull API having methods like HTTP Methods🡪POST,GET,PUT,DELETE

And respective action will be done like Create,Fetch all,Fetch single data,Update,Delete

Also using status code

But ->RESTfull its accesable http:..url so that function

Ex:POST🡪app/products – Create data

GET🡪app/products—fetching all data

GET🡪app/products/{id}—fetching Single data

PUT🡪app/products/{id}—Updating single data

DELETE🡪app/products/{id}—Deleting Singledata

Statuus:

200-ok

201-Created

404—Not

500—Internl Error

401—Authentication

403—bad request

Now every application is running using internet like windows,mobile,web.. then they can use RESTfull service..

FullstackApplciation🡪from frontend to full backend also implementd

Front end –Angular/React

Backend—RESTfull services

Database/sql/nosql.

WebAPI:

1. Specfic Types
2. IActionResult
3. ActionResult<T>

Ex:

Public sting GetDate()

{

Return “hello”;

}

[Prodcues]

API

IRepoditoty<T> ….. Type – T🡪 why🡪It can be any type of tables classe like Product,Order,Customer,Blog

While creating these class I need to specify it can be a entity class/base class/Interface also id is common for all class

Ex: while creating repostity

I Repositoty<T> were T:EntityBase

1. DateRepository
   1. Add
   2. Update
   3. Delete
   4. GetAll
   5. GetById
   6. GetByDate

Entity framework configuration

These 2 needs to be installed from the Nuget package manges

Microsoft.EntityFrameworkCore.SqlServer

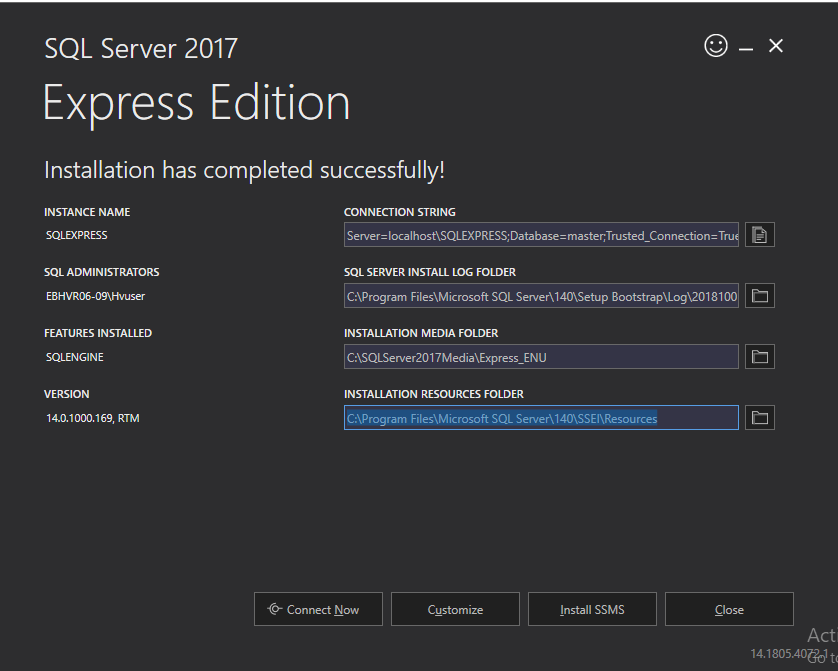
Microsoft.EntityFrameworkCore.Design

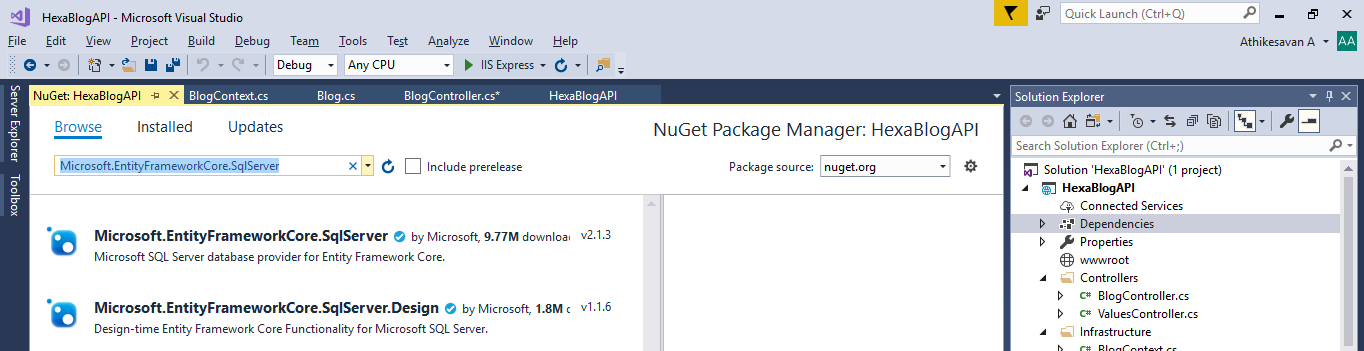
Server=localhost\SQLEXPRESS;Database=master;Trusted\_Connection=True;

C:\Program Files\Microsoft SQL Server\140\Setup Bootstrap\Log\20181001\_112023

C:\SQLServer2017Media\Express\_ENU

C:\Program Files\Microsoft SQL Server\140\SSEI\Resources





Cors policy🡪cross domain policy this connection is allowed /not.

**Exception Handling**

**.---------------------------------**

**Swagger**

**Its an API documentation method.in webservice case will use wsdl but here asp.netcore use Swagger.**

Types:

1. Swagger
   1. **Swashbuckle Asp.netCore**
      1. **Swashbuckle.Asp.netCore.Swagger**
      2. **Swashbuckle.Asp.netCore.SwaggerGen**
      3. **Swashbuckle.Asp.netCore.SwaggerUI🡪creating doc UI API including Testing like postman**

* Swagger Form🡪 WT Token auth.

1.Create image gallery API

2.implement concept swagger

3.Exception Handling

4.Repository Pattern

Upload Image

Swagger

Exception

API,Repository

APP UI-Using React

Download Image

Identity

**Security**

Security—

Authorization:Bearer<token>

Some token is passed from client to server.

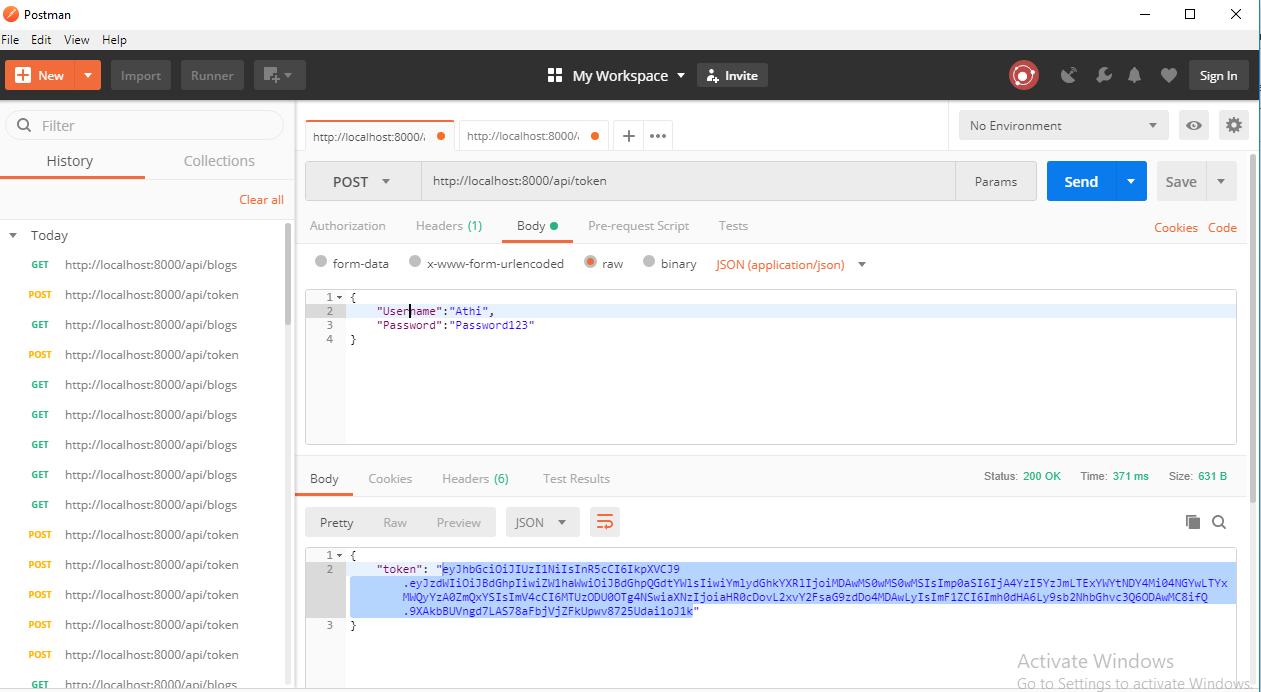
Scope-which resource want to protect

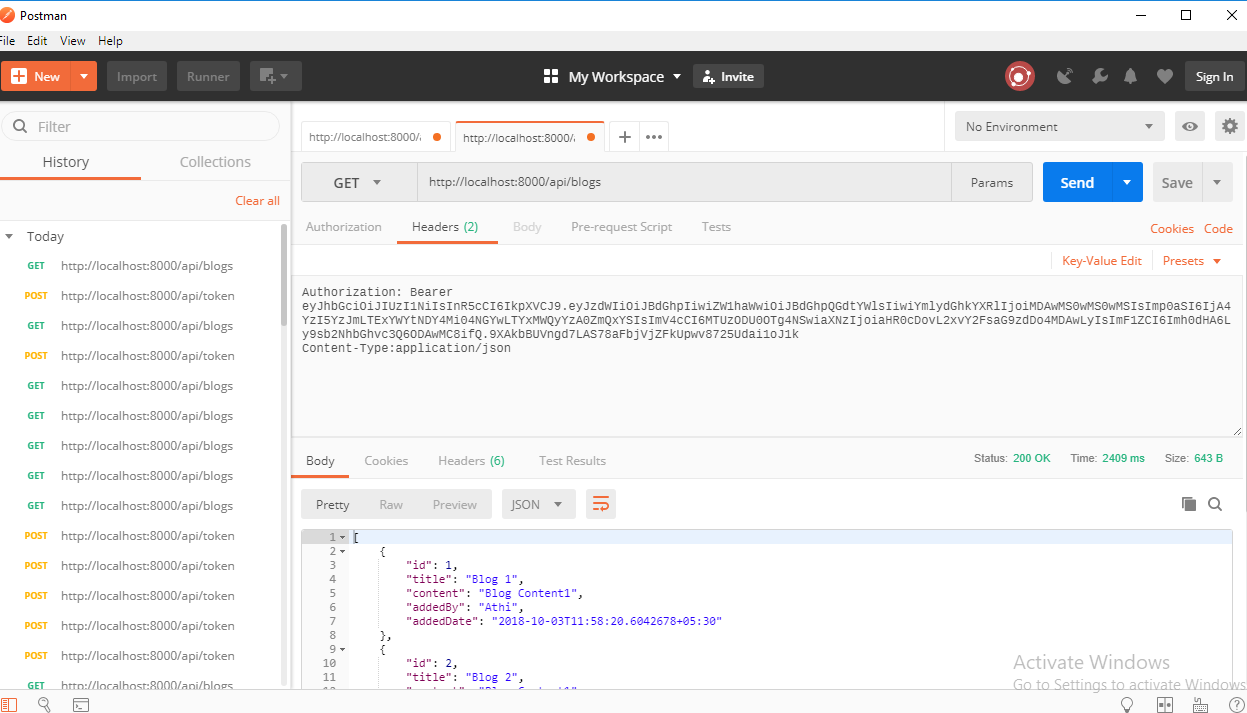
Access token- contains claims information also can contain identity token also.

Identity token – only check whether the user is valid / not

Referesh token - When you generate token for shot period when you want to relogin also contains user information .sent to identifyand create

This is used to access for





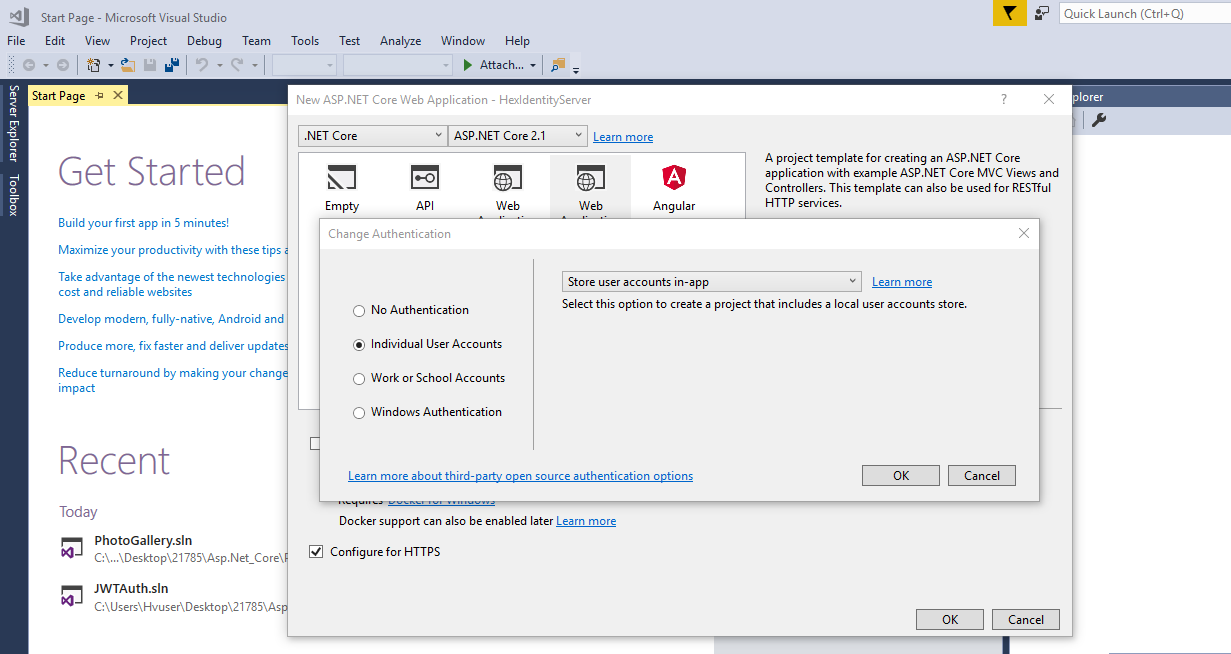
Identity

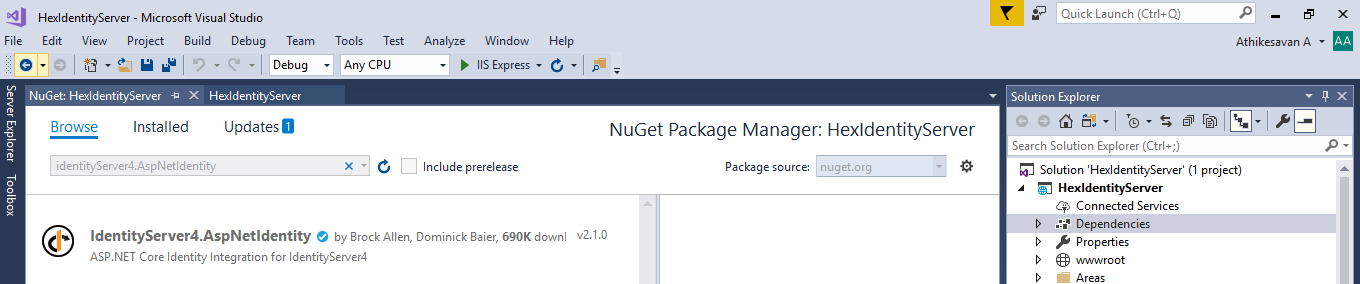
1. Initial inMemory config
2. Same changes into entityFramework Configuration

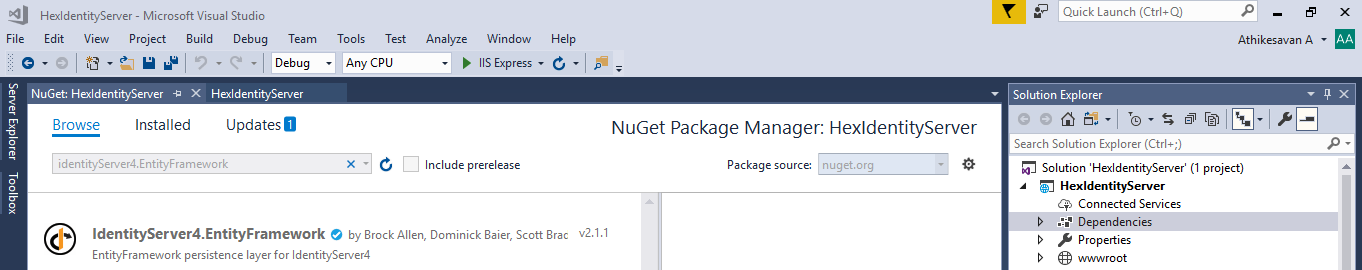
Create MVC App

Then Login in into Identity server

While login it loads login page







HexSPAClient – Applciaiton Creation steps

1. index.html
2. callback.html
3. Oidc-client.html
4. app.js

<https://github.com/sonusathyadas>

🡪 <https://github.com/sonusathyadas/DotNetCore_IdentityServer>

Download projects from the above link and restore database in [**HexIdentityServer**](https://github.com/sonusathyadas/DotNetCore_IdentityServer/tree/master/HexIdentityServer) in this project using below comment

1. Build application.
2. Change connection string in appsettings.json file

"DefaultConnection":"Data Source=localhost\\SQLEXPRESS;Initial Catalog=HexIdentityDb;Trusted\_Connection=True;"

1. cmd: **Update-Database**

dotnet ef migrations add InitialPersistedGrantDbMigration -c PersistedGrantDbContext -o Data/Migrations/IdentityServer/PersisterGrantDb

dotnet ef migrations add InitialConfigurationDbMigration -c ConfigurationDbContext -o Data/Migrations/IdentityServer/ConfigurationDb

Language Tools

Deployment

Scaling 🡪Particulart alone we can able to

Security

Microservices🡪 Its and individual piece of application

* Which can be deployed any platform
* Which can be individually scalable
* Which can be individually securable
* Which can be individually developed different language.

There is vertical division of module.

Whenever create micro Services it can be

REST-Synchronous communication

Messaging- Asynchronous communication.

You can deeply your individual application using web app

Azure Service Fabric--. You can develop ---

Microservices deployment is easy using Docker.

Every Microservices can have a

Docker is an container- having an imge on multiple layer .--> act as hub. To upload files as image into using “push” and download “Pull” and login command

Prerequisites to user docker:

Command –

docker pull microsoft/dotnet:2.1-sdk

docker pull microsoft/dotnet:2.1-aspnetcore-runtime

docker build –t <imagename>

docker build –t blogapi

docker build –t blogapi . //here . means current working directory

somename:tag

blogapi:1.0

Ex:

docker pull sonusathyadas/products/api

docker tag

to reduce the size of the containers.. for the same tag while we ..

docker images

docker ps-🡪 it shows currently running mode

docker ps –s 🡪 it includes everyprocess not only running –like stoped running

docker rmi <imgeid> 🡪 to remove docker image first 4 charcters

docker stop <containerId> 🡪 if you want to stop the process

docker rm<containerId>

some times rm 🡪 removal layer failed like dependency for this we can forcefully remove

docker rm <containerId> -f

Docker File:

FROM <basicImage> -- this is an command

LABEL <commits> -- this is an method

RUN dotnet restore -- this is an command it will create new image and new container will be create

RUN dotnet build -- this is an command

RUN dotnet publish -- this is an command

To avaoid this multiple command to avoid using “&&” to minimize no.of intermediate layer

**RUN dotnet restore && dotnet build && dotnet publish**

**ENV Key** Value

**WORKDIR** <folder path>

COPY src dest

RUN <cmd>🡪run is excuted at that time of building

CMD <cmd—what command needs to be execute> 🡪 when you run the application at that time

Ex: CMD . dotnet run dll

EXPOSE 8080 🡪by default port no.. from the local machine localhost 8080 itwill forward to containers portno 8080

ENTRYPOINT 🡪its similar to CMD specifies the commands and arguments also need to specifies the CMD also.

Ex: ENTRYPOINT cmd 🡪 only the commands

Cmd -- > having arguments.

Static html page using ngnix – how to do that.

**To Crete docker use Visual studio Code**

<h2>Sample App2</h2>

<h2>Sample App2</h2>

**Needs to add dockerfile and inside sample code**

FROM httpd

COPY ./public/ /usr/local/apache2/htdocs/

EXPOSE 80

**Save and run**

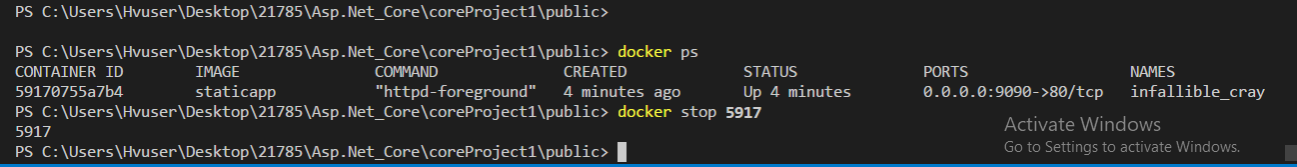
**docker build -t staticapp .**

**docker images**

**docker ps**

**docker run -p 9090:80 staticapp**

**docker stop 5197**



**docker rm 5197**

**docker ps**

**docker ps -a**

**Add🡪 Docker Explorer 🡪 install in Visual studio code**

**Create docker file inside the root folder**

FROM microsoft/dotnet:2.1-sdk AS build

WORKDIR /src

COPY "\*.csproj" "./"

RUN dotnet restore

#copy remaining files

COPY . .

RUN dotnet build -c Release -o /app

RUN dotnet publish -c Release -o /app

#Create image using runtime base image

FROM microsoft/dotnet:2.1-2.1-aspnetcore-runtime AS final

WORKDIR /app

EXPOSE 80

COPY --from=build /app .

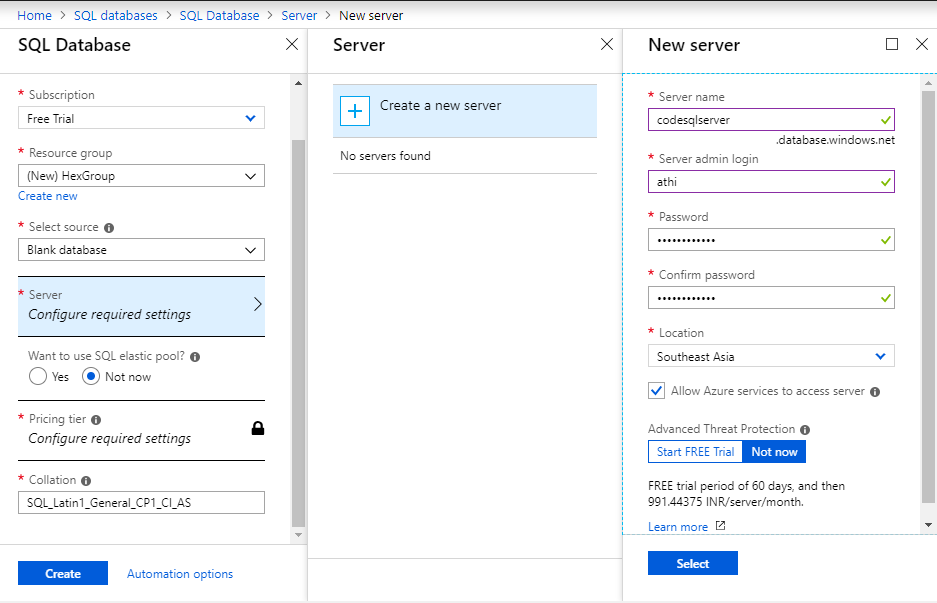
ENTRYPOINT [ "dotnet","HexaBlogAPI.dll" ]

**Azure portal Database connection create**

**Portal.azure.com**

**codesqlserver**

**Password@123**



**Azure-** It is public cloud

Types:

1. Public Cloud
2. Private Cloud
3. Hybrid Cloud

**Cloud Services- 3 types of cloud services available**

1. **Iaas – Infrastructure as a Service you create virtual machine, create evironemnt,setting up application. –Admin**
2. **Paas – Platform as a Service-- Developer**
3. **Saas – Applciation /infrastructure Software as a Service- application also given by them. This is typically used by End user. But azure is not providing Saas.**

**Azure – one of service of Microsoft Iaas /Paas**

**Microsoft – Provide Saas**

**Azure-from Microsoft more from if you want to deoply application you have 6-7 options for container alone itself.**

**Aws – their main focus is Iaas services**

**Azure 🡪 login -🡪** [**http://portal.azure.com**](http://portal.azure.com)

**ARM-** Azure Resource Manager (new service deployment-Primary concept is **Resource Group**, it is new here **RBAC**-Role Based Access Control based on user Role we can give permission, Resource Template also called **ARM Template**) –

its an new deployment model as below

1. Portal – easy to use UI for beginner but its slow depends on network speed may vary.
2. Command Line Method (CLI) Admin/dev they are not waiting so using CLI its fast.
   1. Powershell 🡪 you need to install Azure module for this CLI, Windows user- prefer to go powershell.
   2. X-platform CLI (cross platform Interface 1.0 developed in node.js,2.0-developed using python now it will be work in Linux, Mac also Windows).
3. ARM Template
4. SDK – Developers using SDK like dotnet, java, ruby, python. Etc..
5. REST Full Services- ultimate all Azure services are RESTFull services (its possible to create/delete/manage but users not prefer this everything is REST but directly we can use its very difficult)

**Whenever create service mandatory 🡪 Name, Resource Group, Location remaining will depends on needs.**

**ASM-** Azure Service Management (classic is old model now like Virtual Machine in this model we had an issue with publishing applications-group wise billing is not able to taken, **RBAC, ARM Template** - also not available here,)

**MyHexaAppPlan**

**API – apps similar like web app. Both are giving same functionalities only differences types of service changing.**

**Azure – Storage Account – Storage-Blob,Table,Queues,FileShare the key provided by Azure we can access all these. It will be provide administrator.**

SAS- Shared Access Signature

1. **Storage Account**
   1. **Can store upto 200 storage Accounts each 500 TB \* 200**
   2. **Storage**
      1. Blob [Page blob but page is not read randomly it is streamable data[exe, Iso file], Block blob-Each block can have fixed size file like (audio,Video), Appendable data – If its written we can modify.also every blog stored inside the container like logic folder not an actual folder] .
         1. Page blob
         2. Block blob
         3. Appendable blob
      2. Bolb does have an access policy.3 types of policy availabe
         1. Blob – if you know the url u can access the file but you cannot list names of the files in the container. ( http://storage1.blobcore.windows.net.images/img1.jpg)
         2. Container if you give the container name and list the files

ex; ( http://storage1.blobcore.windows.net.images)

* + - 1. Private – no access ( <http://storage1.blobcore.windows.net.images>) still if you want to access the file using Keys & SAS
    1. Table
       1. NoSQL
       2. Key
       3. Entity(256 key)
       4. Rowket, Pattermnky
    2. Queue (App- App communication)
    3. FileShare

**Storgare/media**

**Hot – frequently used**

**Cool**

**Redundancy**

**LRS - Locally redudant storage (high availability)**

**ZRS- Zone Redundant Storage=>this is available also this is more secure than LRS**

**Geo Redundant which minimum 500 m**

**RA-GR**

**Powershell**

Login-AzureRmAccount

New-AzureRmResourceGroup -Name "SampleGroup" -Location "Southeast Asia"

New-AzureRmStorageAccount -Name "bststoragenew" `

-Location "Southeast Asia" `

-ResourceGroupName "SampleGroup" `

-SkuName Standard\_LRS `

-Kind StorageV2 `

-AccessTier Hot

**Command**

**az login**

**az group list**

**az group list –o table**

**In 🡪 appsettings.json**

**“StorageConnectionString”:”** **DefaultEndpointsProtocol=https;AccountName=bstdatastorgae;AccountKey=Cwsg/hNHrkQVz8fyljBoynAZMgyO1iRXAC1SiyQ2wLPy85YJoxeKIcvcWfJmKkOaRWHWOWFvkZPFRk3bqLczIw==;EndpointSuffix=core.windows.net”**

**08-Oct-18**

**Monday**

**Key – it’s an admin key if share with anyone they can able to access everything.**

1. **SAS [Stored Access Policy] – to overcome key we can give SAS token, the difficult is access revoke part.**
   1. **We can create SAS with some set of polices, also we can give an name.**
   2. **Instead of generating SAS token we can create SAS that is stored inside the database.**

[**https://github.com/sonusathyadas/TrainingMaterials**](https://github.com/sonusathyadas/TrainingMaterials)

**Cosmosdb- it nosql kind of database that can be access by different API’s like**

1. **CosmosDB**
   1. **NoSQL**
   2. **API**
      1. **SQL (DocumentDB)**
      2. **MongoDB**
      3. **Casendra**
      4. **Table**
      5. **Gremline(graphs)**
   3. **SLA on Multiplt Parameters**
   4. **Geo-Replication**

**unstructured hireachal data**

**Server=tcp:codesqlserver.database.windows.net,1433;Initial Catalog=Blogdb;**

**Persist Security Info=False;User ID=athi;Password=Password@123;MultipleActiveResultSets=False;Encrypt=True;**

**TrustServerCertificate=False;Connection Timeout=30;**

**Needs to be installed-**

**Microsoft.Azure.DocumentDB.Core**

**docker build -t imagegallery:v1 .**

**docker run -p 9090:80 imagegallery:v1**

**Different Services available in Azure:**

1. **WebApp for container code and Images)**
2. **Azure Container Instance(ACI)**
3. **Azure Container Services for Kubernate (Aks for Azure Kubernate Services)**
4. **Service Fabric**

**ACR – Azure Container Repository – Its paid and private**

**Docker Hub- Its free**

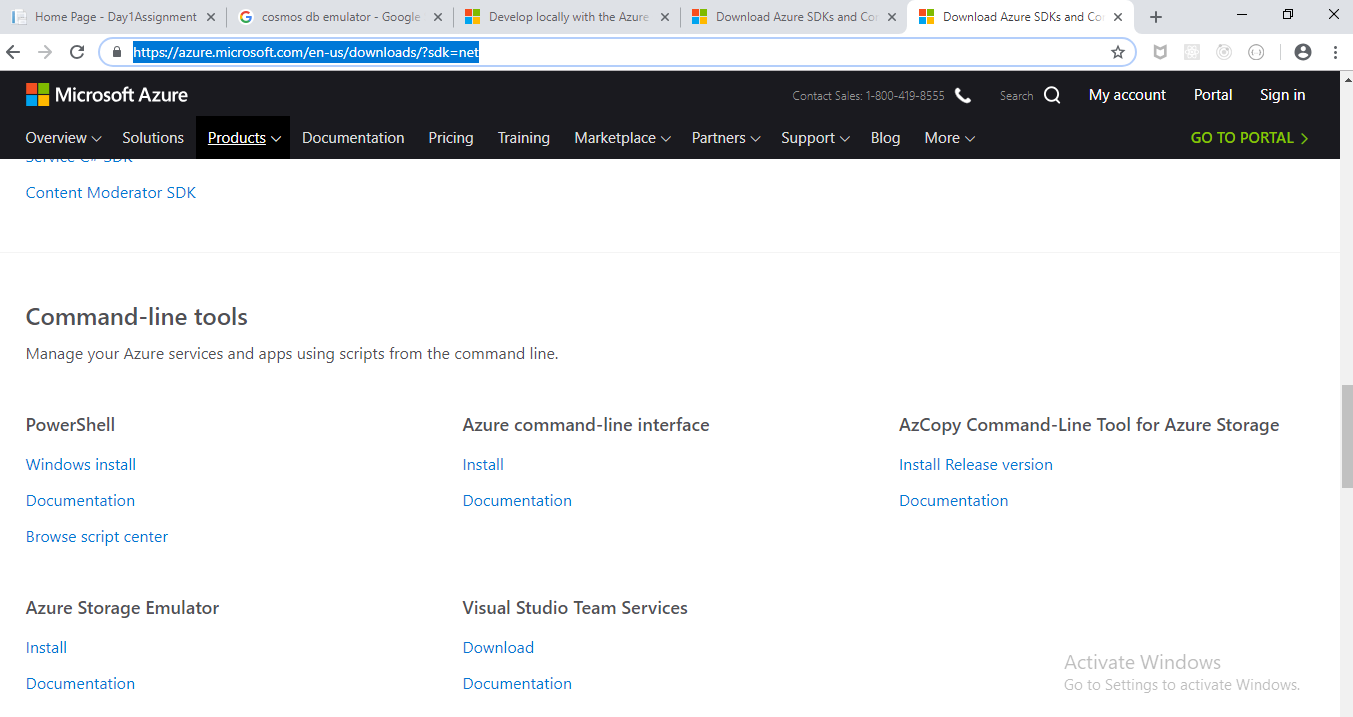
**Storage Emulator**

**If you want to run the application locally you need to download this emulator.**

**CosmosDB Emulator 🡪** [**https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator**](https://docs.microsoft.com/en-us/azure/cosmos-db/local-emulator)

**Azure Emulator**

[**https://azure.microsoft.com/en-us/downloads/?sdk=net**](https://azure.microsoft.com/en-us/downloads/?sdk=net)



**In that under Azure Storage Emulator🡪 click Install 🡪open local**

**AzureStorageEmulator.exe init : Initialize the emulator database and configuration.**

**AzureStorageEmulator.exe start : Start the emulator.**

**AzureStorageEmulator.exe stop : Stop the emulator.**

**AzureStorageEmulator.exe status : Get current emulator status.**

**AzureStorageEmulator.exe clear : Delete all data in the emulator.**

**AzureStorageEmulator.exe help [command] : Show general or command-specific help.**

**09-Oct-2018**

**Tuesday-9.30am-5.30pm**

**Function & Logic App**

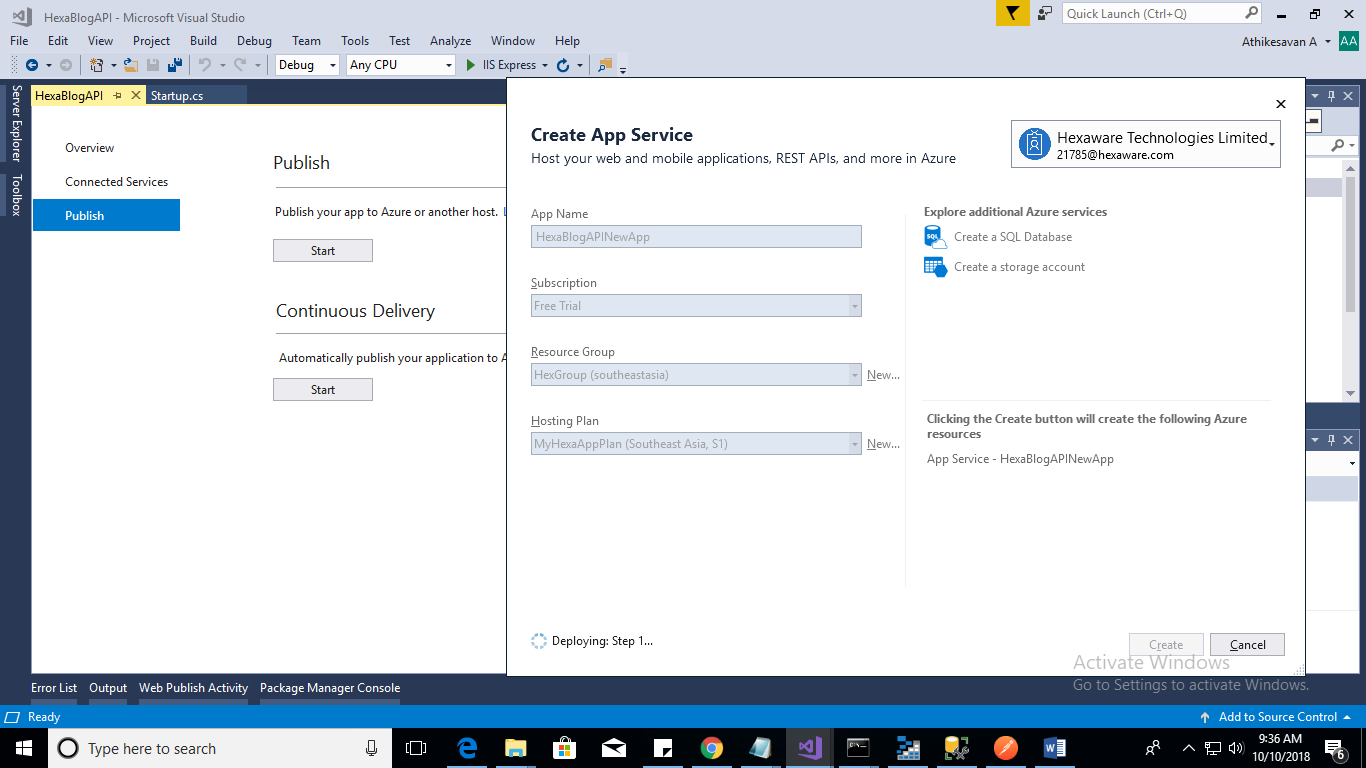
1. Piece of code
2. Execute Independently
3. Faas
4. Event Driven(Triggerd on event)
5. Focus on code
6. Duration – 5 mins(max 10mins)
7. Plan
   1. Consumption plan – it’s better for not continues triggering with in short period of time. Use this its taking more cost.(no infrastructure not created dynamically created infrastructure like Vm based on running time only cost need to pay)
   2. AppService plan – there is no time limit (so if its more than 5 mins better use this)- continues triggering use this its taking less cost. (you need purchase infrastructure vm then create function any how you are running function / not need to pay amount for that vm)
8. **Language (C#, JS, JavaBeans, F#....)**
9. **Trigger** – it’s nothing but sequence of functions based on actions which its firing function🡪only one function can have 1 trigger. if you want to trigger2/more need to be create 2/more functions
10. **Binding-** Establish connection to different data source.
11. Trigger
    1. HTTP Trigger
       1. Authorization level
       2. Function
       3. Admin
12. Timer trigger
    * 1. Signing up for a free Azure account unlocks all Functions capabilities without worrying about time limits!

Schedule Enter a cron expression of the format '{second} {minute} {hour} {day} {month} {day of week}' to specify the schedule.

If simple and single action use Funcitons

If its multiple steps use Logic App (every stage input previous stage take previous stageoutput )

API Management- **API Management services is called API Gateway**



Download swagger add it into

Once the client get the key ocim key you have to add the header 🡪 it will authenticate by gateway

Mocking – use mock response

How to reset input policy -- >limi call rate key 🡪under unbound tag

How to reset Outbound policy - > change header also🡪 under outbound key.

1. Docker🡪 how to create imges to deploy this using
   1. Single container application deoploy
      1. ACI🡪 Azure Container Instance – serverless -> it contains only single container applciation
      2. WebApp🡪 Dedicated paln Single container application
   2. Multi container application deploy in a single service in that cases use some clustered environment also called as MultiContainer Orchestrator
      1. ACS🡪 you have explicitly create the master and slaves , multiple Orchatrator choice
      2. AKS🡪 2benefits , its an managed master like no need to create azure will create master for us, only need to create other node

Only one Orchestrator is kubernates – this is only supports windows , docker support linux

Its an service running from master based on our input using YAML instruction will deploy the applications. If any node goes down automatically will be created new node.it continuously monitor. YAML🡪API-3,UI-1,ID-2.

Cost effective free master given by azure.

It’s a managed master

* + 1. Service Fabric 🡪

1. Components
   1. Master
   2. Nodes –it’s an virtual machine
   3. Pod – kubernet is a service, inside the node machine ->1 pod contains 1 containers, rare case more than 1 container. its starting and stoping will by done by Pod.inside node deploy container its like a n agent
   4. Replication Controller(ReplicaSet)🡪maintaining no.of.cluster if anything goes down it will automatically create another one.
   5. Services 🡪3 instances of same blog api is running if you want to communicate from anything it will use like Load balancers here called as Services.every container have the same API.
      1. This will identify what are all the services for backend also based on **Label**
   6. Deployments🡪its nothing but YAML- file , It’s contains how many containers and services

Kubernaters

DC/OS

Docker Sworm

Cluster Master

Managing the cluster

Coordinatinf all activities in cluster

Scheduling applications

Maintaining

Kubernatebootcamp.gitbub.io/kubernates-bootcamp/1-2html

az group create --name AKSGroup --location Southeastasia

az aks create --resource-group=AKSGroup --name=bstcluster --dns-name-prefix=hexbytestream --generate-ssh-keys --node-count=2

group create --name AKSGroup --location Southeastasia

SQL Database

Add Firewall exception

ACR

login server : aksregistray/azurecr.io

password: d2af48c2-c2c7-4800-a018-45ba5bb329af

App Id: 780c68ce-861c-4289-b089-4b0de0f070cf

working

--------------Step3 -------------------------------------

ACR\_REGISTRY\_ID=$(az acr show --name aksregistray --query id --output tsv)

echo $ACR\_REGISTRY\_ID

--------------password -------------------------------------

SP\_PASSWD=$(az ad sp create-for-rbac --name myacrspsks --scopes $ACR\_REGISTRY\_ID --role owner --query password --output tsv)

echo $SP\_PASSWD

------------App Id

SP\_APP\_ID=$(az ad sp show --id http://myacrspsks --query appId --output tsv)

----------- apply login server\_app\_d\_password -- using visual studio code

docker login aksregistray/azurecr.io -u 780c68ce-861c-4289-b089-4b0de0f070cf -p 780c68ce-861c-4289-b089-4b0de0f070cf

docker login aksregistray/azurecr.io -u 780c68ce-861c-4289-b089-4b0de0f070cf -p 780c68ce-861c-4289-b089-4b0de0f070cf

-----------------------

docker build -t aksregistray.azurecr.io/blogapi .

docker build -t aksregistray.azurecr.io/identityserver .

docker build -t aksregistray.azurecr.io/blogclient .

docker push aksregistray.azurecr.io/blogapi

docker push aksregistray.azurecr.io/identityserver

docker push aksregistray.azurecr.io/blogclient

---------------- if getting issue az login

az aks show --name bstcluster --resource-group AKSGroup --output table

----------------

az aks install-cli

------------------

az aks get-credentials --resource-group=AKSGroup --name=bstcluster

----------------- to see the cluster in your UI need to run the below command

az aks browse -g AKSGroup -n bstcluster

---------------------

kubectl create secret docker-registry regcred --docker-server=aksregistray/azurecr.io --docker-username=780c68ce-861c-4289-b089-4b0de0f070cf --docker-password=d2af48c2-c2c7-4800-a018-45ba5bb329af --docker-email=21785@hexaware.com

Service Fabric:

ACI- is not microservices

Web app - is not microservices

If its go for micrservices is

AKS 🡪 application is containeraised manner (more than 1 container can communicate each other, so its possible to create microservices)

Service Fabric (here primary focus is running container(it can be monolithic or Microservices) Microservices this came before container, in azure most of the cloud implements microservices like office 365 it communicating totally different using Remoting is primary communication method () REST and queues is secondary,)

Disadvantages🡪 all application already developed and deployed as on premesis services, so we need to rewrite entire program also way of coding style is different so that they don’t want to use existing applications.

So recently introduce support for running container for service Fabric cluster (it should be container).

Both are cluster environment.

Service Fabric-

Everything is master/slaves(Nodes) it can be Orchestration.

It can be run /deployable can created Azure Private/Onpremsies cloud but it requires 16 gb ram.

1. Service Fabric applications running in Azure:
   1. Skypefor business
   2. Azure Document DB (Cosmosdb)
   3. Azure Core Infrastructure
   4. Bing Cortana
   5. Azure SQL Databse
   6. Power BI

Services 🡪 Management

Communicaiton

Types of microservies

StateFul microserivices – mainting user data

Staeless microserivices

**Main Services: -- > Running in Service Fabric**

1. Cluster -> managing cluster(deploy,remove,healing->(REST [Http=19080],Powershell/FabricClient [Tcp=19000]))
2. FailOver Manager-Rebalances service instances as node come/go
3. Naming – Registry mapping service instances 🡪 endpoints
4. Fault Analysis – Let’s you inject faults to test your services
5. Image Store – Contains your app packages (not on oneBox)
6. Upgrade – Upgrade SF on nodes(Azure only)

These are all service Fabric architecture for administrator level

Application

Service service service

Code Config Data Code Config Data Code Config Data

Application Type - > it is a categorization of an application and consists of a bundle of service types

Service Type 🡪 is a categorization services

**Manifest file:**

Classes (or “types”) of applciations and services are describerd through XML files (**application manifest** and **service manifests**)

The manifests are the templates against which applications can be instantiated .

You can do all this in automated way using Repository once upload modified code from your local.(you can do either manual / automated using VSTS renamed as Azure DevOps)

CI 🡪 Continuous Integration

CD 🡪 Continuous Deployment

Flight Project---

React JS 🡪

Need to get into to the mainfolder of Pacakage.json🡪 **npm start**