**Web Api:-**

**Section:- 1 Introduction**

**What Is Api:-**

Api is the link can provide data an any format like:- web, android and ios.

* Api can Work on request and response.

**Request:-**

The Request can send to server. And send a data like;:- Verb , Header And content.

**Verb:-** the verb can send a data request like get, post, delete and put.

**Header:-** Header can pass the data like:- Content type, length and authorization.

**Content;-** content can pass the value like:- html,css,xml,and json also that the blogs.

**Response:-**

The Response can send reply to sender and send data like:- status code, Header And Content.

**Create Project:-**

Create A Project in Visual Studio Code.

**Step:-** Visual Studio Code => Create New Project => Choose Asp .Net Core Api (C#) => Add a Name => Select the package of net 7.

**Controller:-** Controller File

**Model:-** .CS File

**Repository:-** .cs File

**IRepository:-** Interface File

**View:-** Can add razor view. And extension is .cshtml File.

**Section:- 2 First End Point**

**Process:-**

**Step:-1** Create A Villa Api Controller To add

**Controller Name in route :-**

**[Route(“Api/[Controller]”)]** => The data will call the controller the data will access the controller name auto catch but when the name can change the data will not catch.

* But name can be specified the controller so that when ever the change the controller name the data can access the easily.
* So that we can prefer the route name can specified easily.

**[ApiController]** => Api Controller Can make all validation first when code run.

* ApiController can work before the other validation.
* They Can validate the **DataAnnotation.**

**Section:- 3 CRUD Functionality**

Process:-

Step:-1 Create Model Of data.

Step:-2 After The Create Model Add Make The Controller And Add A Method.

Step:-3 After Adding The GET, PUT, DELETE Method And Add A Patch Method.

Patch:- For the Work Of Patch Install The package on **Villa\_Api** Project.

* **Microsoft.AspNetCore.JsonPatch**
* **Microsoft.AspNetCore.Mvc.NewtonsoftJson**

After The Adding Package Line Add In Program.CS File.

**builder.Services.AddControllers(option => {**

**//option.ReturnHttpNotAcceptable=true;**

**}).AddNewtonsoftJson().AddXmlDataContractSerializerFormatters();**

A Patch Method Mostly Return **Return NoContent();.**

* **JsonPathch :-** ( Can Add a replace item And add in body Of api item Patch method.)
* **PostMan:-** Post man Are Use For api check Get Detail.

**Function Of PostMan:-**

* Add a Method (GET,POST,PUT,DELETE,PATCH) And A URL.
* HEADER:- Header Can Spred in Key And Value. There add new authorization etc…
* BODY:- Body are used the patch method.

**Step:-** 4 Difference:-

**1.Pathch And Put**

**Pathch:-** patch can update the particular property.

**Put:-** Put can update the entire object.

**2.ActionMethod And IActionMethod:-**

**ActionMethod:-** This method can describe the return type first.

**IActionMethod:-** This Method Can not need to describe return method first.

**3.Apicontroller And Validation**

**ApiController:-** Api Controller validate the dataanotation with first when project with start.

**Validation:-**Validation can Validate the and make process with data annotation but apicontroller make earlier.

**Section:- 4 Dependency Injection And DBContext**

Process:-

Step:-1 Changes in Controller According

* Add logger line in controller and add in AllGet and individual Get Method.
* The Application Json Show can error then comment out a in newtonsoftjson file in http line can comment .

Step:-2 Download The Package

* Serilog.AspNetCore
* Serilog .Sinks.File

Step:-3 After The Install The Package Add a file in Program .cs File

* Log.Logger = new LoggerConfiguration().MinimumLevel.DFebug().Write.File(“log/villalogs.txt”,rollingInterval: RollingInterval.Day).CreateLogger();
* Bulider.Host.UseSerilog();
* After they have special logger but we have use default logger so we can remove this line.
* Unistall The Both Package.

Step:-4 After the all logger sentence can remove in controller.

Step:-5 Create A Custom Login (Create A New Folder Name Of Logging)

* ILogging – Interface File
* Logging – Class File
* (Add logger information for see the custom logger and after than remove it.
* This logger file gor logger information is not need to custom logger so that can’t be add.(
* // builder.Services.AddSingleton<ILogging,LoggingV2>(); ))

Step:-6 Create a Database Model On DATA Folder => ApplicationDbContext.CS

Install The New Package For Database Model.

* Microsoft.EntityFrameworkCore.runtimeCompilation
* Microsoft.EntityFrameworkCore.Tools
* Microsoft.EntityFrameworkCore.SqlServer
* When Sql server add upper two package add automatically due to relation

Step:-7 Changes In Controller According

Step:-8 Add **NoTraking**:-

**Section:- 5 DTO And AutoMapper**

Process:-

Step:-1 Chages According to DTO And Automapper

* Create a model (villaCreateDTO And VillaUpdateDTO )
* Changes In Controller

Step:-2 Create a Async Mehod And Changes In Controller According To Async Method.

Step:-3 Install The New Package In **Villa\_Api** And They Are used In Application.

* **AutoMapper**
* **AutoMapper.Extensions.Microsoft.DependencyInjection**

Step:-4 Create A Mapping Config File And Make Replationship between the models.

Step:- Add File In program.cs

* builder.Services.AddAutoMapper(typeof(MappingConfig));
* And After that add in controller file.

**Section:- 6 Repository**

Api can not intrect with database directly so that we can add Repository.

Step:-1 Crete A Repository Folder And inside this folder can add a IRepositoery Folder.

Step:- 2 Interface Add some item

* Get all villa list
* Get indivisual villa
* Create
* Remove
* Save
* In the get method add a **bool tracked=true;** it used for we add no traking for that.

Step:-3 Add A repository And make some changes in implement. And Async Method.

Step:-4 Add A **ApiResponce** Model for make the response of api when work .

* This model can not be Add in data base
* This model Property are use in repositor.

Step:-5 Add Try And Catch Method

* Try:- This Method can analyze the data and find the error.
* Catch:- They Can work the error can cathch and throw the data base and save them.

Step:- 6 When we Create a model in the id we can add a below line:-

* **DatsbaseGeneratedOption.None** =>

Step:-7 When we crete new model and add repository so that add file in Program.cs file:-

* Builder.Services.AddScopped<IVillaRepository, VillaRepository>();

**Section:- 7 Consuming API**

Process:-

Step:-1 Create A MVC Project in

Step:-2 Make Some Changes In Launch Setting And add a location of api.

Step:-3 Create A Project Of **Utility** And A Type Of **Class Library** And Make A Class In **SD**

Step:-4 Create A Folder Of Service And Make The Peta Folder Of IServices.

Step:-5 Create New Class Of Mapping Config

* Make The New File Program.cs File
* Add Services In Program.Cs File

Step:-6 Create A New File Views And Controller.

Step:-7 When We Make A Migration So Restart The app And Again Try The Migration.

Step:-8 Create A View Model.

**Section:- 8 API Security**

Step:-1 Add Models For Login And registration

* Add a model Of LoginRequest, LoginResponce , RegistrationRequestDto And LocalUsers.
* After that add a localUsers in ApplicatonDbContext.

Step:-2 Add IUserReposotory And UserRepository.

* Add a login And Register request and response in IUserReposotory.

Step:-3 implement User Repository – register

The register data implement and specified here.

ApplicationUser user = new()

{

UserName = registerationRequestDTO.UserName,

Email = registerationRequestDTO.UserName,

NormalizedEmail = registerationRequestDTO.UserName.ToUpper(),

Name = registerationRequestDTO.Name

};

Step:-4 implement the secretkey string in appsetting.json

"ApiSettings": {

"Secret": "THIS IS USED TO SIGN AND VERIFY JWT TOKENS, REPLACE IT WITH YOUR OWN SECRET"

}

Step:-5 Generate Token on Sucessful Login

* Name And role save as a claim identity. And make the code expire in seven day.
* var roles = await \_userManager.GetRolesAsync(user);
* var tokenHandler = new JwtSecurityTokenHandler();
* var key = Encoding.ASCII.GetBytes(secretKey);
* var tokenDescriptor = new SecurityTokenDescriptor
* {
* Subject = new ClaimsIdentity(new Claim[]
* {
* new Claim(ClaimTypes.Name, user.Id.ToString()),
* new Claim(ClaimTypes.Role, roles.FirstOrDefault())
* }),
* Expires = DateTime.UtcNow.AddDays(7),
* SigningCredentials = new(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
* };
* var token = tokenHandler.CreateToken(tokenDescriptor);
* LoginResponseDTO loginResponseDTO = new LoginResponseDTO()
* {
* Token = tokenHandler.WriteToken(token),
* User = \_mapper.Map<UserDTO>(user),
* Role = roles.FirstOrDefault(),
* };
* return loginResponseDTO;
* }

Step:-6 Add a controller

* In this Controller add a repository responce.

Step:-7 In The UserRepository add the token and user the when login request send.

return new LoginResponseDTO()

{

Token = "",

User = null

};

Step:-8 Add user repository in program.cs

* builder.Services.AddScoped<IUserRepository, UserRepository>();

Step:-9 Secure api EndPoint

* Add Authentication Add in VillaApiController.

Step:-10 Authentication In Action

* Doenload the package gor jwt token
* After that the add authentocan line in Program.c File

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x => {

x.RequireHttpsMetadata = false;

x.SaveToken = true;

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key)),

ValidateIssuer = false,

ValidateAudience = false

};

});

* After that add pipeline in program .cs

app.UseAuthentication();

Step:-11 swagger And bearer in Action

* **Bearer** Are used to before paste token in authentication.

builder.Services.AddSwaggerGen(options => {

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description =

"JWT Authorization header using the Bearer scheme. \r\n\r\n " +

"Enter 'Bearer' [space] and then your token in the text input below.\r\n\r\n" +

"Example: \"Bearer 12345abcdef\"",

Name = "Authorization",

In = ParameterLocation.Header,

Scheme = "Bearer"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement()

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

},

Scheme = "oauth2",

Name = "Bearer",

In = ParameterLocation.Header

},

new List<string>()

}

});

}

**Section:- 9 Consuming Secured API**

Process:-

Step:-1 create a model (Login(Request/Response)DTO , RegistrationRequestDTO And UserDTO)

Step:-2 Create a AuthService and IAuthService implement.

* After The implement the of service add a service in program.cs file
* builder.Services.AddHttpClient<IAuthService, AuthService>();
* builder.Services.AddScoped<IAuthService, AuthService>();
* Auth Service can make the call of api and access.

Step:-3 Add A Auth Controller And the implement the code.

Step:-4 Add A view About the loginRequest, Register And Authservice.

Step:-5 Add A tocken property in SD,CS File.

* public static string SessionToken = "JWTToken";
* After the Add it apply tocken access method in Controller Method.
* And all of later add tocken session in program.cs file

**builder.Services.AddDistributedMemoryCache();**

**builder.Services.AddSession(options => {**

**options.IdleTimeout = TimeSpan.FromMinutes(100);**

**options.Cookie.HttpOnly = true;**

**options.Cookie.IsEssential = true; });**

* After add this Line Add A pipe line.
* **app.UseSession();**
* In this session loggin token add in controller in login method.

Step:-6 Add session token in send in login session through the layout

* Add in upper side of layout format.

@using MagicVilla\_Utility

@inject Microsoft.AspNetCore.Http.IHttpContextAccessor httpAcc

* Add in the layout of navbar section.
* All are the get accces use from the SD class.

@if (httpAcc.HttpContext.Session.GetString(SD.SessionToken) != null &&

httpAcc.HttpContext.Session.GetString(SD.SessionToken).ToString().Length > 0)

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Auth" **asp-action**="Logout">Logout</**a**>

</li>

}

else

{

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Auth" **asp-action**="Login">Login</**a**>

</li>

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-controller**="Auth" **asp-action**="Register">Register</**a**>

</li>

}

Step:-7 Add Authentication In Web project

* Make Changes in Auth Controller of role And claim identity.

var identity = new ClaimsIdentity(CookieAuthenticationDefaults.AuthenticationScheme);

identity.AddClaim(new Claim(ClaimTypes.Name, model.User.UserName));

identity.AddClaim(new Claim(ClaimTypes.Role, model.User.Role));

var principal = new ClaimsPrincipal(identity);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal);

Step:-8 Add Admin Authorization in villa controller And villNumberController.

**[Authorize(Roles = "admin")]**

Step:-9 Inside A Program.Cs File Add Authuntication and path

**builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)**

**.AddCookie(options =>**

**{**

**options.Cookie.HttpOnly = true;**

**options.ExpireTimeSpan = TimeSpan.FromMinutes(30);**

**options.LoginPath = "/Auth/Login";**

**options.AccessDeniedPath = "/Auth/AccessDenied";**

**options.SlidingExpiration = true;**

**});**

After that add a pipeline. => **app.UseAuthentication();**

Step:-9 Add Admin Authorization In Villacontroller And VillaNumberController. In vill spi project

Step:-10 Add Below line in **villa\_web Project** in all controller

* **HttpContext.Session.GetString(SD.SessionToken)**

Step:-11 Add A token property in APIRequest

* public string Token { get; set; }

Step:-12 Add A authorization token line add in BaseService

if (!string.IsNullOrEmpty(apiRequest.Token))

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiRequest.Token);

}

Step:-13 Add below file add in all service of vill\_web project.

* string token
* After A api Location Add A Below File:-
* Token = token.

**Section:- 10 Versioning In API**

Change the after the new update in api we might to update the api version.

**Process:-**

**Step:-1**

Install New package the change the versioning:-

* **Microsoft.AspNetCore.MVC.Versoining**
* **Microsoft.AspNetCore.MVC.Versoining.ApiExplore**

Step:- 2

After the install the package the add a line I **api project**:-

**builder.Services.AddApiVersioning(); =>** after the this line of code the api version cannot the run so that we make the change the globally endpoint. And change line in below code:-(this version of project in two type minor and mazor => that can change by the Small number in below condition)

builder.Services.AddApiVersioning(options => {

options.AssumeDefaultVersionWhenUnspecified = true;

options.DefaultApiVersion = new ApiVersion(1, 0);

});

Step:-3

After this line of code api version is not specify so that make changes make in controller:-

Add This Add I top of the controller:- **[ApiVersion(“1.0”)]**

**Step:- 4**

But the controller have the multiple versions of api and they have in different action method so than we can make some and describe the multiple version By:- **[MapToVersion(“1.0”)]**

But Route Can not aceepet the multiple version in different action method **[Route(“api/VillaNumberApi”)]** to change with **[Route(“api/v{version:apiVersion}/VillaNumberApi”)]**

Step:-5

After the change of route in controller add a line to program.cs fille to save groupnameformat.

**builder.Services.AddVersionedApiExplorer(options =>**

**{**

**options.GroupNameFormat = "'v'VVV";**

**options.SubstituteApiVersionInUrl = true;**

**});**

* In this format of groupname Small-v is value and Upperrcase-VVV is version name.
* After that you can not see the swager error.
* For the upper version method api had a multiple end point.

Step:-6 Multiple version Access

* When the multiple version can add I controller add a default version add in controller.
* After that Add this line in program.cs file in ApiVersioning inside.
* **Option.ReportApiVersions = true;**
* Other the line add in program.cs file in ApiVersionInUrl inside add for the make url acces.
* **Option.SubstituteApiVersionInUrl = true;**
* Upper line add for the multiple version iss not in controller so that it cannot saw error in Swagger And not display the multiple Version.

Step:-7 Swagger Documentation for V1 and multiple Swagger documentation.

* For the clean of swagger are used in program.cs file **App.UseSwagger();** are display swagger in working the upper side.
* And a other than search bar can specified the particular version Filter.
* After That The Swagger location line are in program.cs file are Add.
* This file are add under

**app.UseSwaggerUI(options => {**

**options.SwaggerEndpoint("/swagger/v1/swagger.json", "Magic\_VillaV1");**

**options.SwaggerEndpoint("/swagger/v2/swagger.json", "Magic\_VillaV2");**

**});**

* After That below dark line are add for the security and license.

builder.Services.AddSwaggerGen(options => {

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description =

"JWT Authorization header using the Bearer scheme. \r\n\r\n " +

"Enter 'Bearer' [space] and then your token in the text input below.\r\n\r\n" +

"Example: \"Bearer 12345abcdef\"",

Name = "Authorization",

In = ParameterLocation.Header,

Scheme = "Bearer"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement()

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

},

Scheme = "oauth2",

Name = "Bearer",

In = ParameterLocation.Header

},

new List<string>()

}

});

options.SwaggerDoc("v1", new OpenApiInfo

{

**Version = "v1.0",**

**Title = "Magic Villa V1",**

**Description = "API to manage Villa",**

**TermsOfService = new Uri("https://example.com/terms"),**

Contact = new OpenApiContact

{

Name = "Dotnetmastery",

Url = new Uri("https://dotnetmastery.com")

},

License = new OpenApiLicense

{

Name = "Example License",

Url = new Uri("https://example.com/license")

}

});

options.SwaggerDoc("v2", new OpenApiInfo

{

**Version = "v2.0",**

**Title = "Magic Villa V2",**

**Description = "API to manage Villa",**

**TermsOfService = new Uri("https://example.com/terms"),**

Contact = new OpenApiContact

{

Name = "Dotnetmastery",

Url = new Uri("https://dotnetmastery.com")

},

License = new OpenApiLicense

{

Name = "Example License",

Url = new Uri("https://example.com/license")

}

});

});

Step:-8

* **[ApiVersion(“1.0”,Deprecated = true)]** => They will display the depreciated in header.But that also is not that explicit.
* **[ApiVersionNeutral]** => that is common api version use no matter that what version are used.

Step:-9

* **ApiResponse!=null =>**

**Section:- 11 Caching, Filter And Pagination**

**Catchiing:-**

* Catching are deficult for some application. Api end point that typically sends the same data and you recive multiple request for the same point.
* Add this below line to mactive trhe catch method. And this line add in **program.cs** file.
* **builder.Services.AddResponceCaching();**
* Add below line to get response cache and by default add **Duration** And this line add in **controller.**
* **[ResponceCache(Duration =30)]**
* **[ResponceCache(Duration =30,Location =ResponceCacheLocation.None,NoStore = true)]**
* **NoStore = true =>** are used for the error the not store in datbasse.
* **Location =ResponceCacheLocation.None=>** The location can add a the path the cache can run.
* For the cache of profile we can add file in program.cs file.

Builder,Services.AddControllers(option => option.CacheProfiles.Add(“Default30”,

New CacheProfile()

{

Duration = 30

**});**

* After the Add this upper line remove the duration in controller and make the name replacement of Duration.

**Filter:-** filter are used the data will fielter by the particular action method like ooccuption.

* [FromQuery] Are used to a try the filter by the field.
* After the upper command add the string? Search and add In the if condition.

**Pagination:-** Pagination are used in the data arrange in one by one page.

* **Int pagesize=3, int pageNumber = 1 =>** add in repository and other than first are the page size and other are the page number.
* After the add the upper line add below line in repository.

**if (pageSize > 0)**

**{**

**if (pageSize > 100)**

**{**

**pageSize = 100;**

**}**

**}**

**//skip0.take(5)**

**//page number - 2 || page size -5**

**//skip(5\*(1)) take(5)**

**Query = Query.Skip(pageSize \* (pageNumber - 1)).Take(pageSize);**

* In repository get methos are add other line after the this method.
* **pageSize:pageSize,pageNumber:pageNumber**
* **Add pagination in header for that add model and line in repository.**
* **Response.Header.Add(‘X-Pagination”, JsonSerializer.Serialoize(pagination));**

**Section:- 12 .Net Identity**

Process:-

Step:-1 Create Model ApplicationUser Model And Add identity in ApplicationDBContext.

* DbContext Replace With IdentityDbContext<ApplicationUser>
* Add table Name And add below line on seed data for add indetity table.
* base.OnModelCreating(modelBuilder);

Step:-2 After the add upper line we have add below line in program.cs file

builder.Services.AddIdentity<ApplicationUser, IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

Step:-3 Add maping file in villa\_api project.

CreateMap<ApplicationUser, UserDTO>().ReverseMap();

Step:-4 Add login Identity

* Create A UserDTO model.
* Add role and user refrence pass in loginResponceDto File.

public UserDTO User { get; set; }

public string Role { get; set; }

* After all that changes in user controller change according to mapping and userDto model.
* And role of user and add user manager.

Step:-5 Add Register Identity in UserRepository

* Changes in userrrepository And IUserRepository AS Aspect to localUser To ApplicationUserr And LocalUser to Userdto.
* After the Add user management and add role for admin and customer. Like:-

try

{

var result = await \_userManager.CreateAsync(user, registerationRequestDTO.Password);

if (result.Succeeded)

{

if (!\_roleManager.RoleExistsAsync("admin").GetAwaiter().GetResult())

{

await \_roleManager.CreateAsync(new IdentityRole("admin"));

await \_roleManager.CreateAsync(new IdentityRole("customer"));

}

await \_userManager.AddToRoleAsync(user, "admin");

var userToReturn = \_db.ApplicationUsers

.FirstOrDefault(u => u.UserName == registerationRequestDTO.UserName);

return \_mapper.Map<UserDTO>(userToReturn);

}

}

catch (Exception e)

{

}

Step:- 6 Add Role Manager Add in UserRepository.

* private readonly RoleManager<IdentityRole> \_roleManager;

Step:-7 Download the package

* System.IdentityModel.Tokens.Jwt
* Remove role in LoginResponce mmodel and userRepository file.
* And other hand the remove role in UserDTO and And id change int to string.

Step:-8 Add token handler line in claim identity where acces their controller.

//var handler = new JwtSecurityTokenHandler();

//var jwt = handler.ReadJwtToken(model.Token);

**Section:-13 Deployment**