https://www.youtube.com/watch?v=\_uZYOgzYheU

#. http basic

#. Understanding request object

#. Examine response object

#. Willapi controller inherit controlbase

\*. Apicontroller attribute:

#. Return end point with some status code

#. To document api we have decorator [producesresponsetype]

#. Adding new data to villa store

#. Postman accept type etc

#. Using serilog .net core 1:37:00

#. Custom logging with DI depend inject.

#.real world scenario add database to our application

**#. Link** application dbcontext to use the connection string and register in program.cs

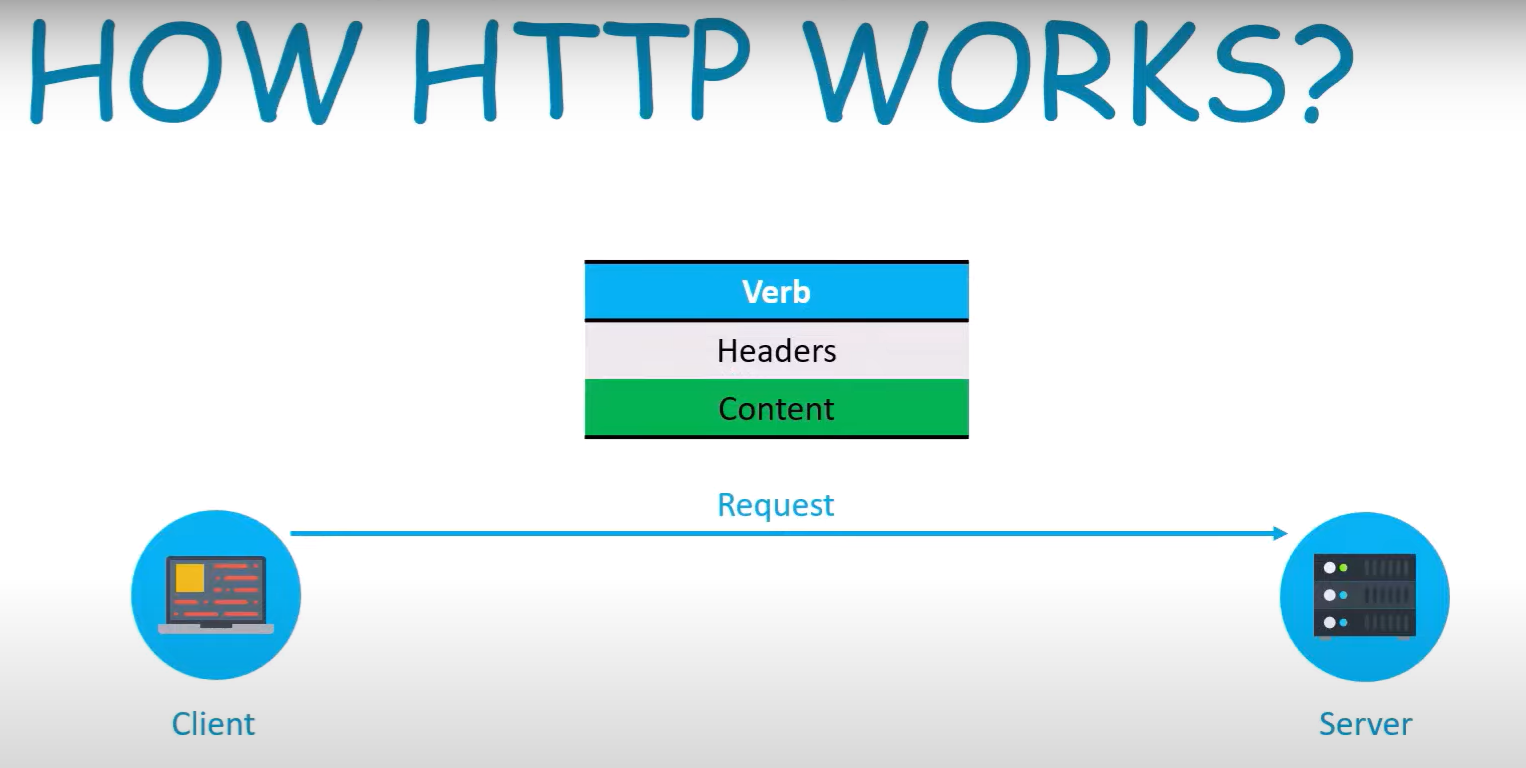
#. In code first approach we have to do migration so it entityframework changes reflects in db

#. How to see some record in db table with migration

#. Using applicationdbcontext in our villaapicontroller earlier we use villastore for saving data

#. http basic

Time: 7:26



When you search in google some thing in google.com it’s going to request to server this request is text document

It sends text doc to google server this text doc contains 3 infos

Verb: what action to take

Headers: has info of request itself

Content: which is optional field

**Request from client as text** doc



Example above is sent to the server: we want to create some thing on the server

http verb is post, header we have info: content length and content here is 9 byte

finally we send magic api as content

magic api is the string which we want to create

all data will be inside the content

when server receives text doc: it receives in the format which it can process or it might rejected because some thing might not valid with the request

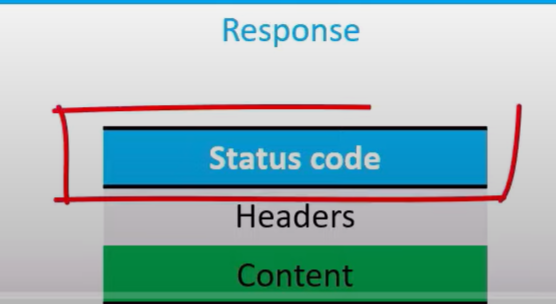
in both the cases it’ll send a response back to the client

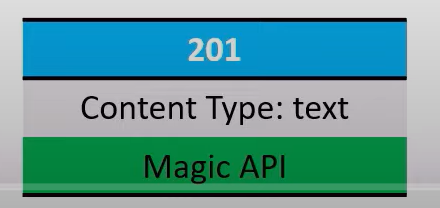
in that response we’ll have 3 items again we’ll have status code header and content

when client makes request server processes that and sends back a response

response is also a piece of data

he told **response in server**:





We can think of response also as text document

In response status code: basically defines the request made was it successful did it fail based on different scenario we have different status code

Example of status code: 201 created,

201 means it created resource which I asked for and I’m going to return some data along with that

In content type: we say that data that I’m sending back is of type text and then in actual content lets say the server is sending back magical api

Imp thing: server itself is stateless : stateless means server doesn’t remember any things

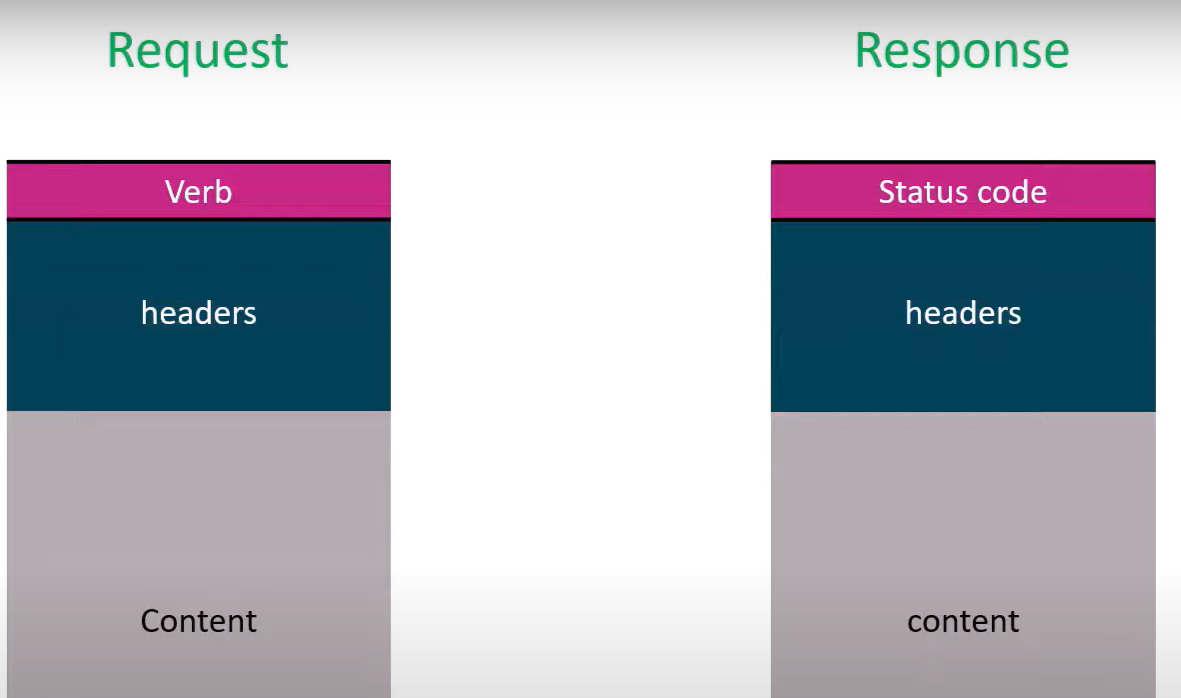
Server gets many requests

Once server process those requests and sends a response back it forgets about that request

**Remember:** if server remember **all requests just imagine how massive memory consumption will be we may need very large amounts of data storage facilities**

If we store all requests: **disk will be overloaded** with the **data of all the requests** that it received

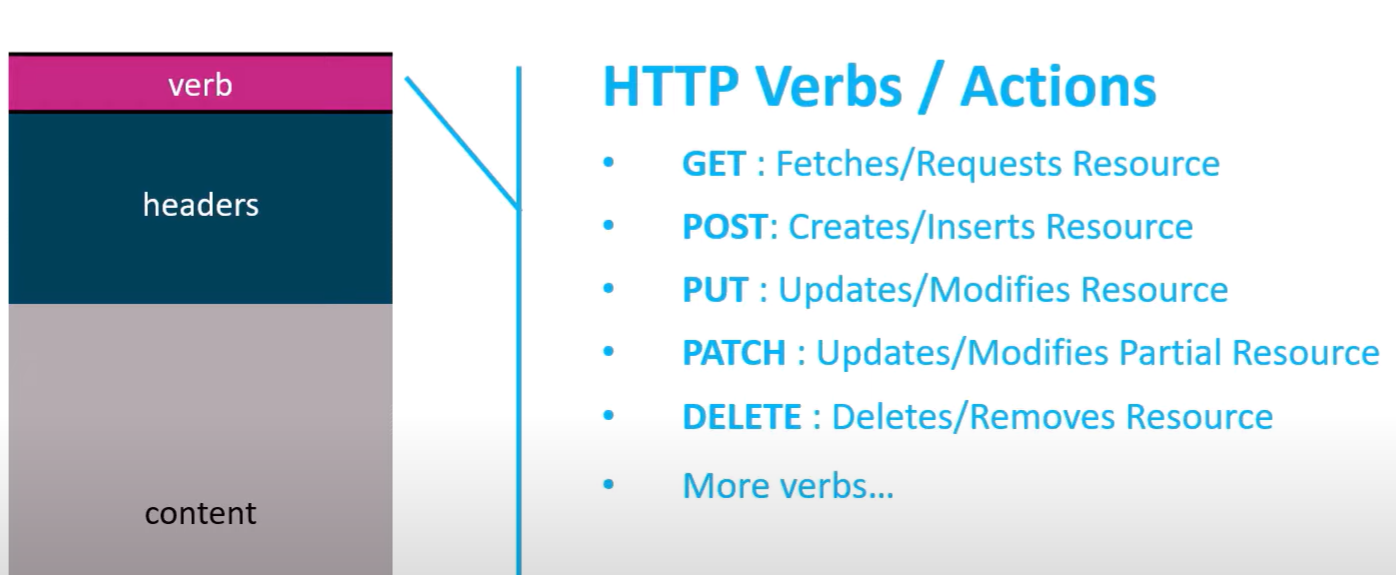
Th**at’s why server will be stateles**s



See above: we’ve request originated from the client, request will have verb header and content

For this request we’ll have response and response will have status code headers and content

#. Understanding request object



Request has different verb: like above get …

Header:



Has above things

Headers: are set of name: value pair that are meta data about the request

Content type:questions what is content type of the request

Is it binary data json file or xml or is it a plain file

Content length: defines size of the content

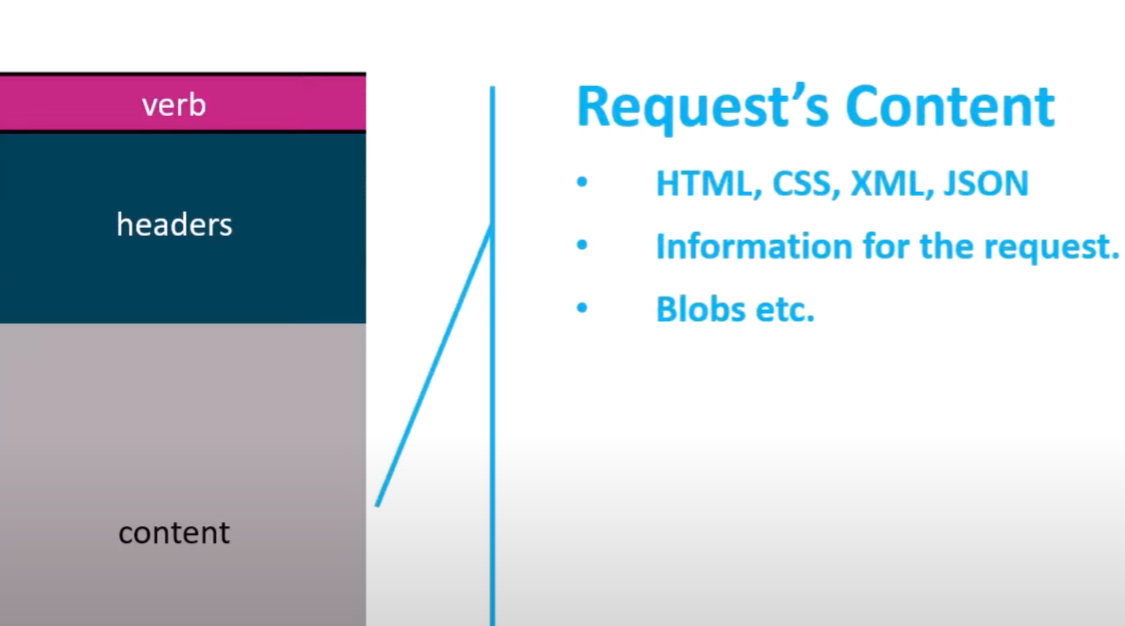
Authorization: some times request needs to be authenticated in that case authorization header will be populated with better token

Accept type: means what type of request is acceptable json xml so on

Headers: hundres of headers we can define our own header

Headers have metadata of content and request

Content:



Content is optional field here we can pass the content that the server will require to complete the request

It can be a json object which can be deserialized by the server to process the request

It can blobs that have to be created or updated

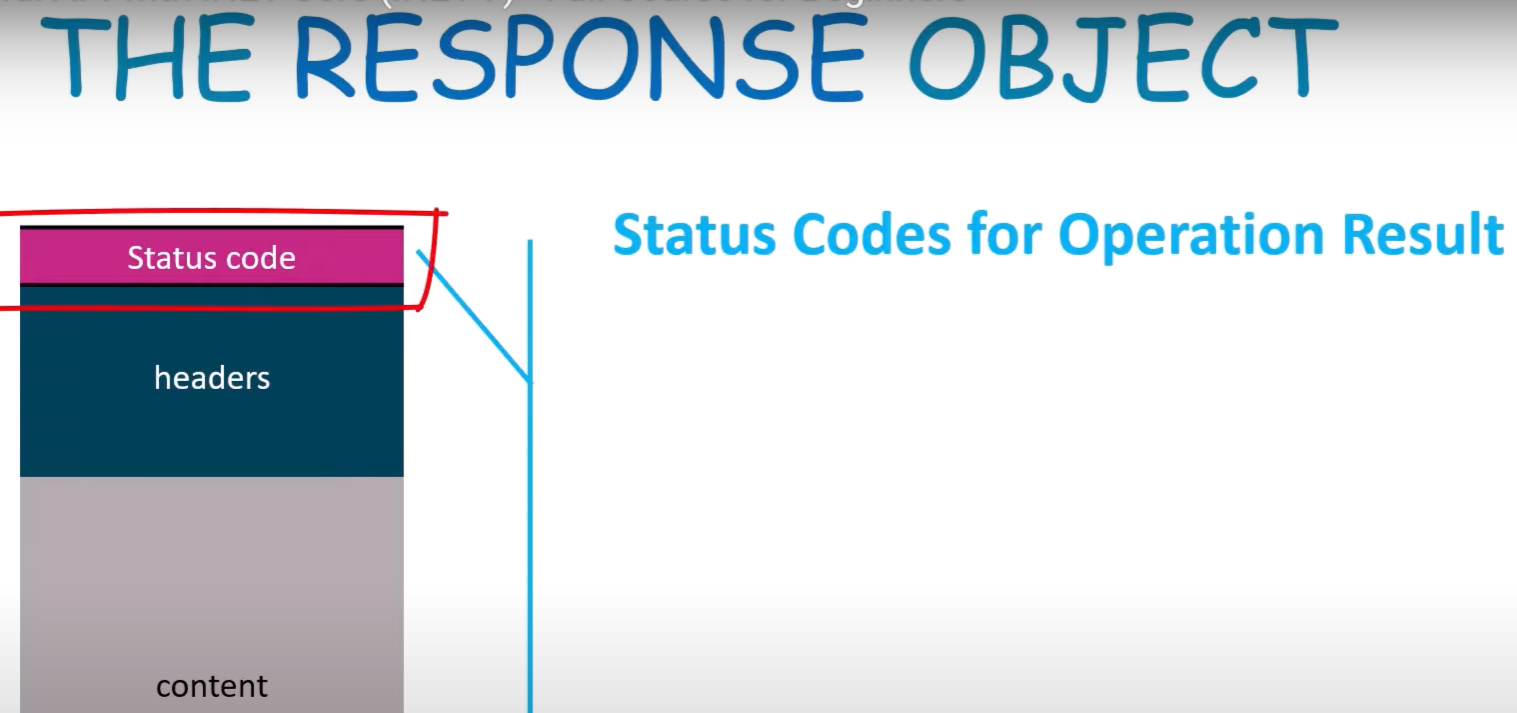
If you’re trying to retrieve a resource then you’ll be using http get verb get will never have the body we’re requesting the api to return some thing and the api does not expect any data in the request

If we post or update then we might have to pass the object that needs to be updated or created that the way api can extract that object from the content and process the request

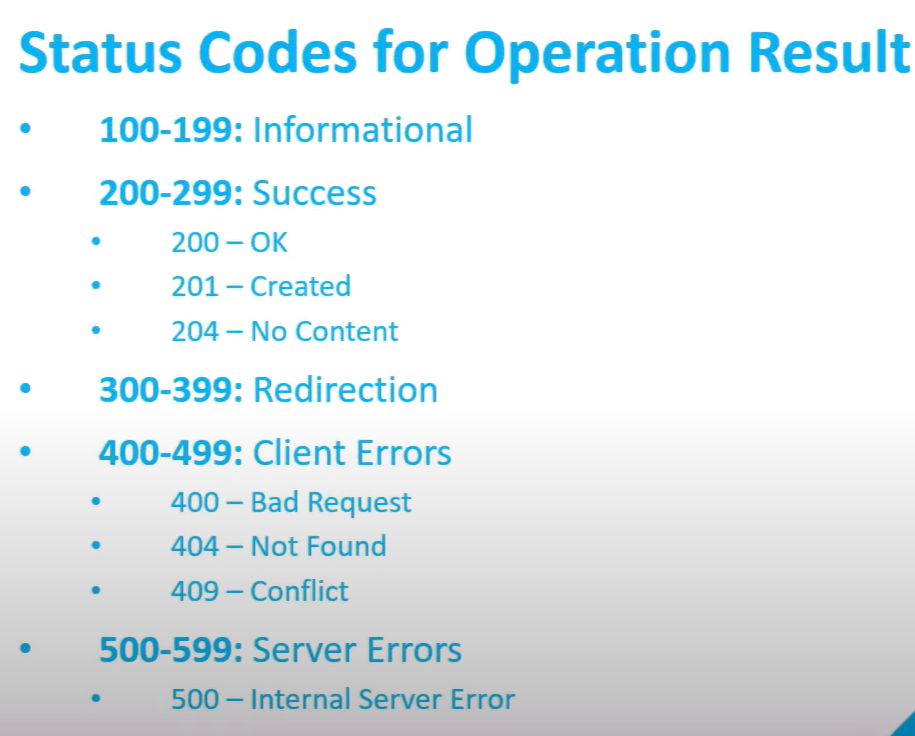
#. Examine response object

Time: 15:35

Once request is received by the server or api it’ll process that it might be accepted or be rejected fail or success whatever it’s it returns back a response



\*. **Status code**:



First thing in response is status code

Represents simply a number which represents what was done on the server and its final result

There are some ranges of nums which represents some properties

100-199 informational

200-299 successful: request was completed successfully

201-When we create or update we want to know the user that create or update was successful

204: if we don’t want to pass the updated record in that case 204

300-399 redirection

400-499 client errors

Means: error in the request that was sent: we may not have included a query string or some part of uri does not exist

Famous code we see frequently: 404 not found

If we access an endpoint which does not exist , if we want to retrieve some data based on id and that id doesn’t exist in that case api may return 404

400 status code stands for bad request: means some thing in request not as expected

500-599 server error

Means errors or exception encountered on server while processing the request

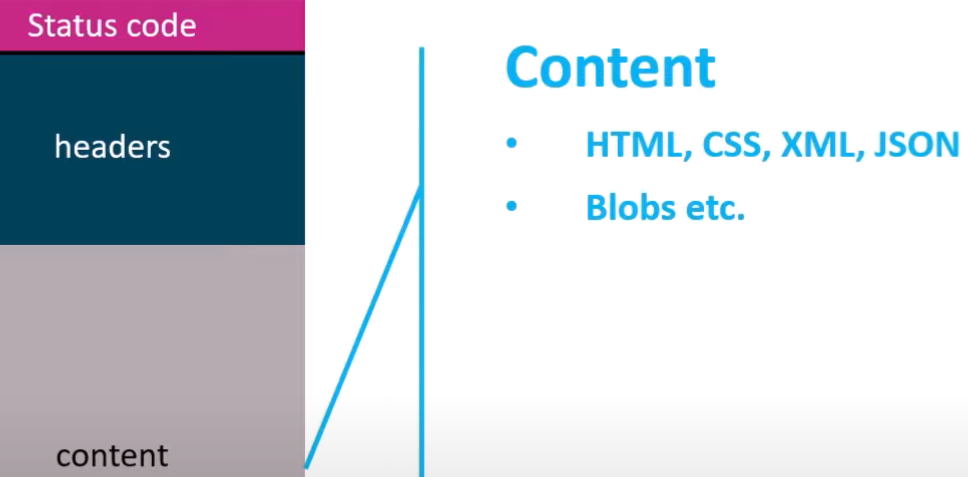
Based on status code

\*. **Headers from response** object:



headers contains metadata of response object

\*. Content in response object



Similar to content in request object we’ve content in response object

That might contain json result

#. Willapi controller inherit controlbase

Controllerbase class contains common methods for returning all the data and users related to the controller’s internal application

Use **ControllerBase** when:

You are building a RESTful API only (no views).

You don’t need Razor Views or HTML rendering.

Your controller actions return JSON, XML, or plain data (via return Ok(), return NotFound(), etc.)

Use **Controller** when:

You are building a web application (MVC) with Razor Views.

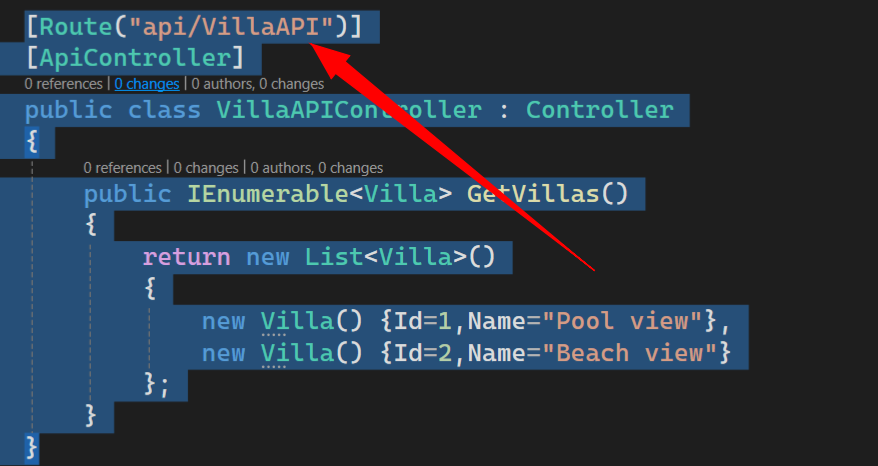
You need to return HTML views using View() method.

You’re using both API and UI logic in the same app (hybrid).

\*. Apicontroller attribute:

Notifies the application that this will be an api controller

When we have apicontroller we must define route in controller or else we get error



VillaAPI which is controller name is route of this controller

Now if we run app still we get error in browser

**Failed to load API definition.**

**Errors**

Hide

**Fetch error**

response status is 500 https://localhost:7185/swagger/v1/swagger.json

Above is error

At action method level put some http verb not at controller level

Code

[Route("api/VillaAPI")]

[ApiController]

public class VillaAPIController : Controller

{

[HttpGet]

public IEnumerable<Villa> GetVillas()

{

return new List<Villa>()

{

new Villa() {Id=1,Name="Pool view"},

new Villa() {Id=2,Name="Beach view"}

};

}

}

above

route(‘api/[controller]’)

if we do like above it automatically takes controller name that is VillaAPI name it takes

goto browser type this

[localhost:7185/api/VillaAPI](https://localhost:7185/api/VillaAPI)

above we see data

A screenshot of a computer

AI-generated content may be incorrect.

Returns data like above

#. Using dtos instead of using data object that is model

List<dataObject>

38:20

Here we use dto instead of model class which we created

He created VillaDTO class in that put id name and datecreated this object used in controller

public class VillaDTO

{

public int Id { get; set; }

public string Name { get; set; }

public DateTime CreatedDate { get; set; }

}

Above

Instead of database we create villa datastore:VillaStore

This is separate class similar to like database: here we create new list VillaDTO and for this we add data there itself inside villaStore

40:53

#. Add another api endpoint passing id to api but got error how to solve what to change

[HttpGet]

public VillaDTO GetVilla(int id)

{

return VillaStore.villaList.FirstOrDefault(x => x.Id == id);

}

Above code

Now we get error because api gets confused to call either getVillas method or getVilla

<https://localhost:7185/api/VillaAPI>

go through above url you get error

**An unhandled exception occurred while processing the request.**

AmbiguousMatchException: The request matched multiple endpoints. Matches:  
MagicVillaApi.Controllers.VillaAPIController.GetVilla (MagicVillaApi)  
MagicVillaApi.Controllers.VillaAPIController.GetVillas (MagicVillaApi)

Error says GetVilla and GetVillas used conflicts

Api gets confused whether it has to invoke GetVillas or GetVilla method

Why because same HttpGet for 2mthods

In order to tell GetVilla expects id parameter then for [HttpGet] we’ve write id so=>

[HttpGet(“id”)]

[HttpGet]

public IEnumerable<VillaDTO> GetVillas()

{

return VillaStore.villaList;

}

[HttpGet("id")]

public VillaDTO GetVilla(int id)

{

return VillaStore.villaList.FirstOrDefault(x => x.Id == id);

}

Above is corrected GetVilla it’s httpget has id with it

Now all works fine

**https://localhost:7185/api/VillaAPI/id?id=1**

Then url will be above

[HttpGet("{id:int}")]

public VillaDTO GetVilla(int id)

{

return VillaStore.villaList.FirstOrDefault(x => x.Id == id);

}

Above says only int can be used

44:00

#. Return end point with some status code

When we have different status code that we need to define with all of the end points

One way is using: **ActionResult**

Actionresult is implementation of IActionresult with that we can use any return type that we want

If all are good in GetVillas we want to return

Return Ok returns empty 200 status code

[HttpGet]

public ActionResult<IEnumerable<VillaDTO>> GetVillas()

{

return Ok(VillaStore.villaList);

}

[HttpGet("{id:int}")]

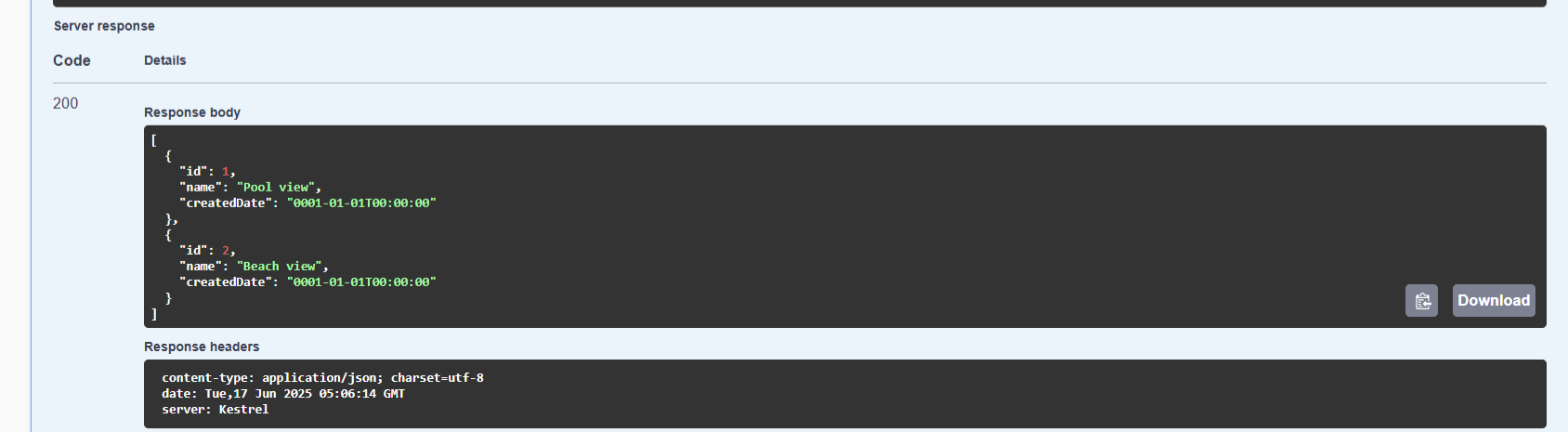
public ActionResult<VillaDTO> GetVilla(int id)

{

return Ok(VillaStore.villaList.FirstOrDefault(x => x.Id == id));

}

Above code we used actionresult and Ok which returns 200 status code



Swagger screen shot

400 bad request if id==0 we return bad request

404 not found

[HttpGet("{id:int}")]

public ActionResult<VillaDTO> GetVilla(int id)

{

if (id == 0)

{

return BadRequest();

}

var villa = VillaStore.villaList.FirstOrDefault(x => x.Id == id);

if (villa == null)

{

return NotFound();

}

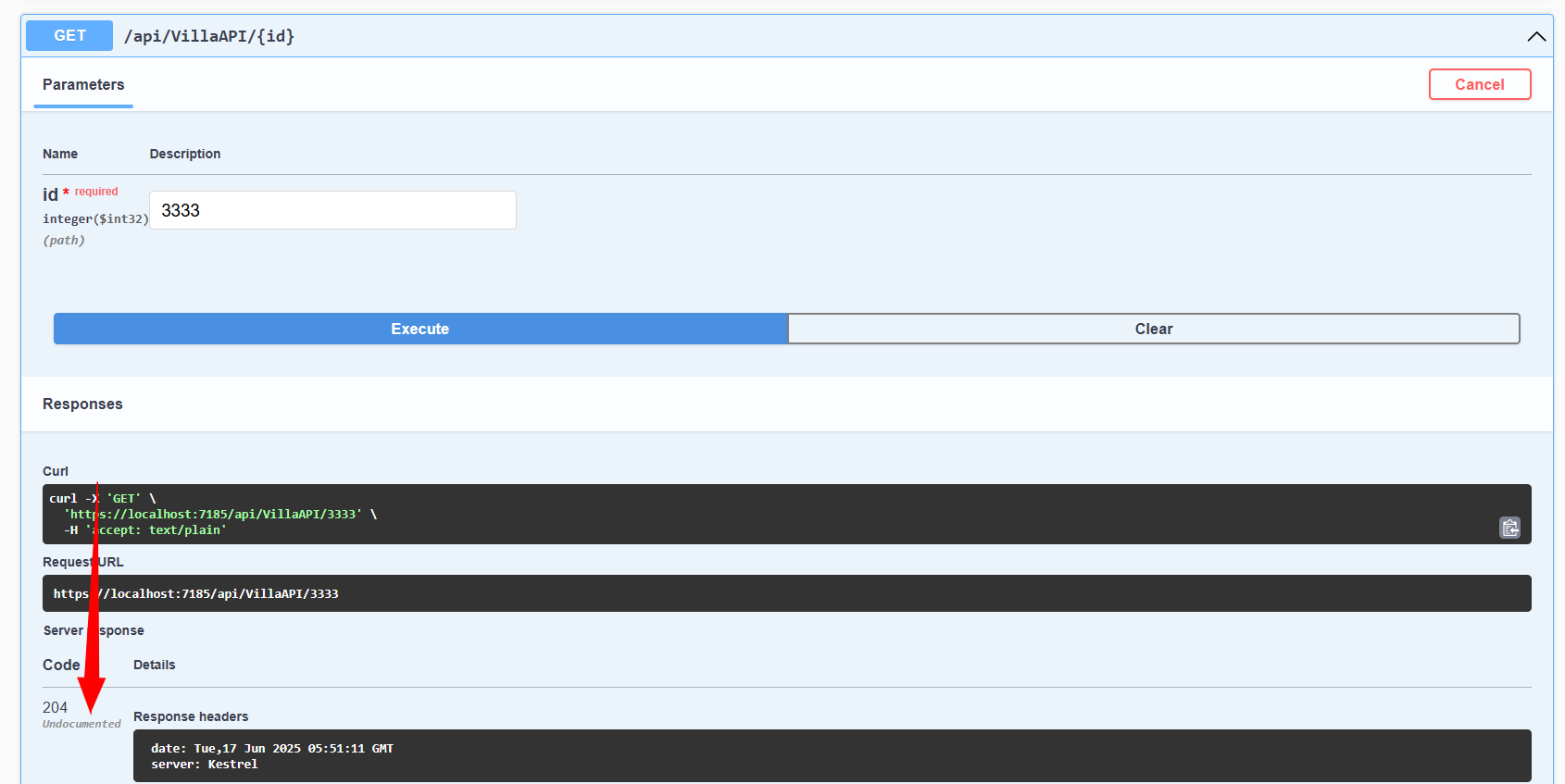
return Ok(VillaStore.villaList.FirstOrDefault(x => x.Id == id));

}

Above code 400, 404 and 200 included

#. To document api we have decorator [producesresponsetype]

In swagger type 0 or id as 1000 you see status code 400 with undocumented



So above it returned status code but it’s undocumented

Means it’s not documented in api

ProducesResponseType in this we can define multiple response type that we can produce

[HttpGet("{id:int}")]

[ProducesResponseType(200)]

[ProducesResponseType(400)]

[ProducesResponseType(404)]

public ActionResult<VillaDTO> GetVilla(int id)

{

if (id == 0)

{

return BadRequest();

}

Code above after add produces.. with status code now run the app you don’t see

Undocumented in swagger below status code

Also we see 2 responses in swager one is server response which is proper response

And other is responses in that we see 200 400 and 404

In our controller api method: we have return type VillaDTO so in swagger also

**{**

**"id": 0,**

**"name": "string",**

**"createdDate": "2025-06-17T06:02:24.705Z"**

**}**

It tells 200 response will be in above format above format coming because of return type is VillaDTO

We can also pass type to producesresponsetype

[ProducesResponseType(200,Type=typeof(VillaDTO))]

[ProducesResponseType(400)]

[ProducesResponseType(404)]

public ActionResult<VillaDTO> GetVilla(int id)

{

if (id == 0)

{

return BadRequest();

}

Like above

We can also pass status code to producesresponsetype

[HttpGet("{id:int}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<VillaDTO> GetVilla(int id)

{

if (id == 0)

{

return BadRequest();

}

Like above

#. Adding new data to villa store

We use httppost verb for insert new data

When we use httppost object that we receive here is from httpbody we add that attribute as parameter

54:44

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<VillaDTO> CreateVilla([FromBody]VillaDTO villaDTO)

{

if(villaDTO == null)

{

return BadRequest(villaDTO);

}

if(villaDTO.Id > 0) {

return StatusCode(StatusCodes.Status500InternalServerError);

}

villaDTO.Id = VillaStore.villaList.OrderByDescending(u => u.Id).FirstOrDefault().Id + 1;

VillaStore.villaList.Add(villaDTO);

return Ok(villaDTO);

}

Above code post request pass

{

"id": 0,

"name": "test"

}

Param like this in swagger

You get 200 ok request

55:00 after this not typing

1:03:18

Api input validataion VillaDTO for name [required] [maxlength]

Now in post request if you don’t pass name it gives error says name field required

This is done by [apicontroller] decorator

Video 1:9:00 he explained difference b/w using actionresult and iactionresult

In actionresult we’ll pass return type but in iactionresult we don’t pass any return type so it’s no content return type

Video 1:12:00 explained about put request

Video 1:17:00 httppatch request explained

For jsonpatch check it in jsonpatch.com

<https://jsonpatch.com/> from this website

{ "op": "replace", "path": "/biscuits/0/name", "value": "Chocolate Digestive" }

above is format of jsonpatch we’ve to follow

in swagger check patch tab

[

{

"operationType": 0,

"path": "string",

"op": "string",

"from": "string",

"value": "string"

}

]

Request body body will be like above

1:22:58

For patch request for id=2

[

{

"path": "/Name",

"op": "replace",

"value": "test4"

}

]

Request body I passed above

Why put replace? Check in jsonpatch.com

Op means operation what do you want to do: we want to replace old value of id 2 with new value

Path means: which field you want to replace

Time: 1:30:13 using postman here

#. Postman accept type etc

Explained headers, accept type and all

A screenshot of a computer program

AI-generated content may be incorrect.

Even though accept type I put application/xml it’s showing in json format

If we keep ReturnHttpNotAcceptable=true then in swagger we can’t see data it gives 406 error

So only for testing purpose below use it **later comment above code**

builder.Services.AddControllers(options =>

{

options.ReturnHttpNotAcceptable = true;

}).AddNewtonsoftJson();

Inside program.cs file we do changes we add above

#. Now to support xml formatting we again do changes

builder.Services.AddControllers(options =>

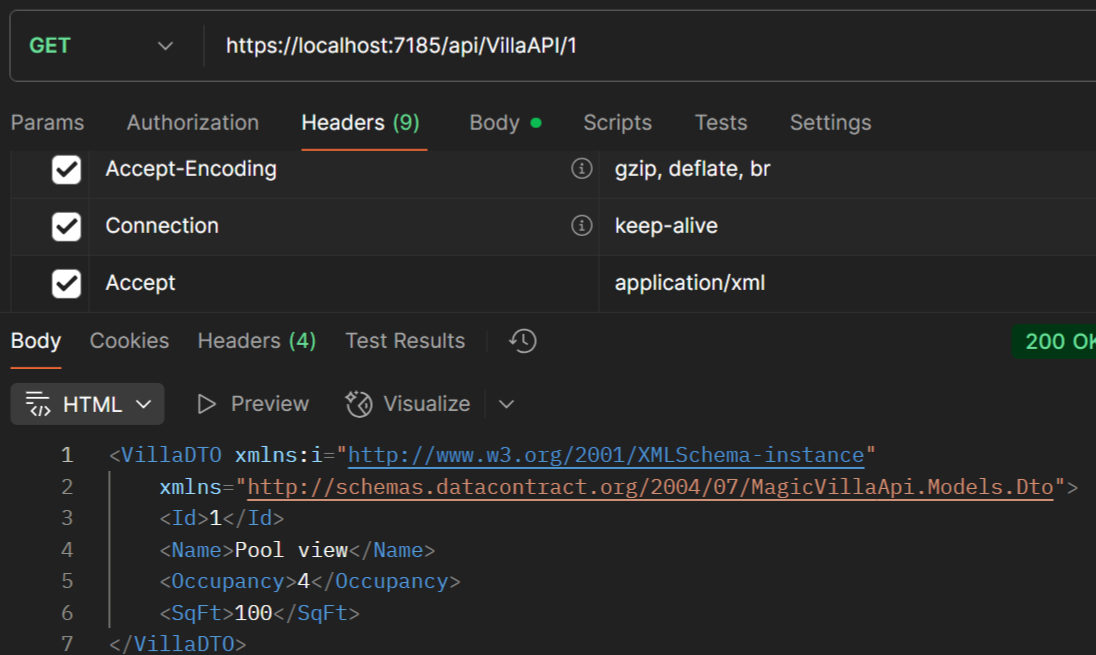
{

options.ReturnHttpNotAcceptable = true;

}).AddNewtonsoftJson().AddXmlDataContractSerializerFormatters();

above

after adding for xml now we get data in xml format



#. Logging in .net core api

By default is built in console window

For including logging we need to inject via dependency injection

To check or see logging is built in go to solution exploerer -> appsettings.json we have some section for logging where default log level is information also we have warning

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

Above

So where will our logs will drop:

When you run your app: you see console window in that we see logs

A screenshot of a computer

AI-generated content may be incorrect.

We can add more logs to see in console window

Like above in console window we see logs

How to use or put logs in console window from our controller using constructor dependency injection in controller VillaAPIController

private readonly ILogger<VillaAPIController> logger;

public VillaAPIController(ILogger<VillaAPIController> \_logger)

{

logger = \_logger;

}

Like above we add in DI

With DI we directly get implementation

After defining above we use logger in every http method

Below code we put log for getting all villas:

public ActionResult<IEnumerable<VillaDTO>> GetVillas()

{

\_logger.LogInformation("Getting all villas");

return Ok(VillaStore.villaList);

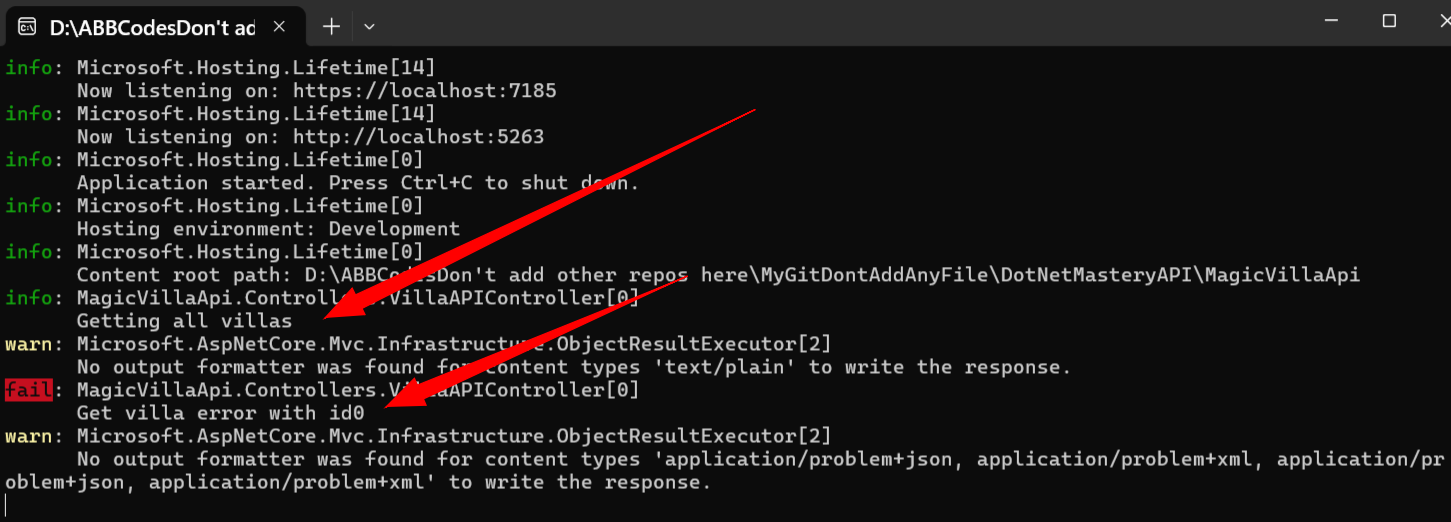
}

Above

#. For errors we put log like this

\_logger.LogError("Get villa error with id" + id);

Above



As we can see now 2 logs are printed in console

In production he told we use **seriallogger** for .net core we use **serialog.netcore**

In logging we can also set: minimum log level also inside program.cs file

1:35:00 or moe

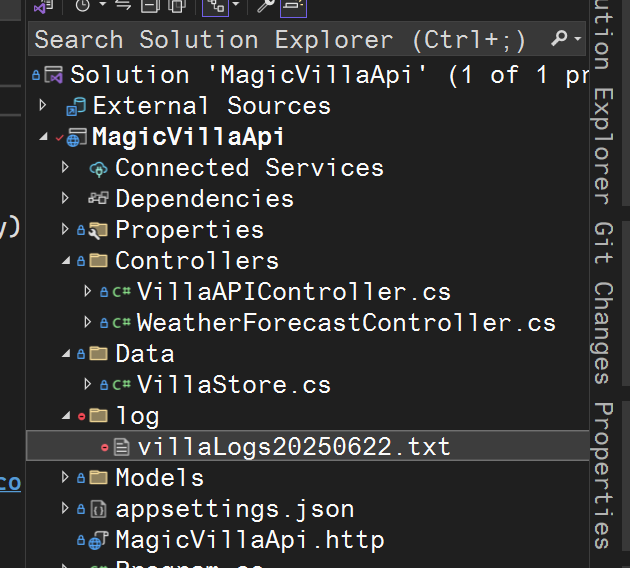
#. Using serilog .net core 1:37:00

Serilog we use it for custom implementation of logging

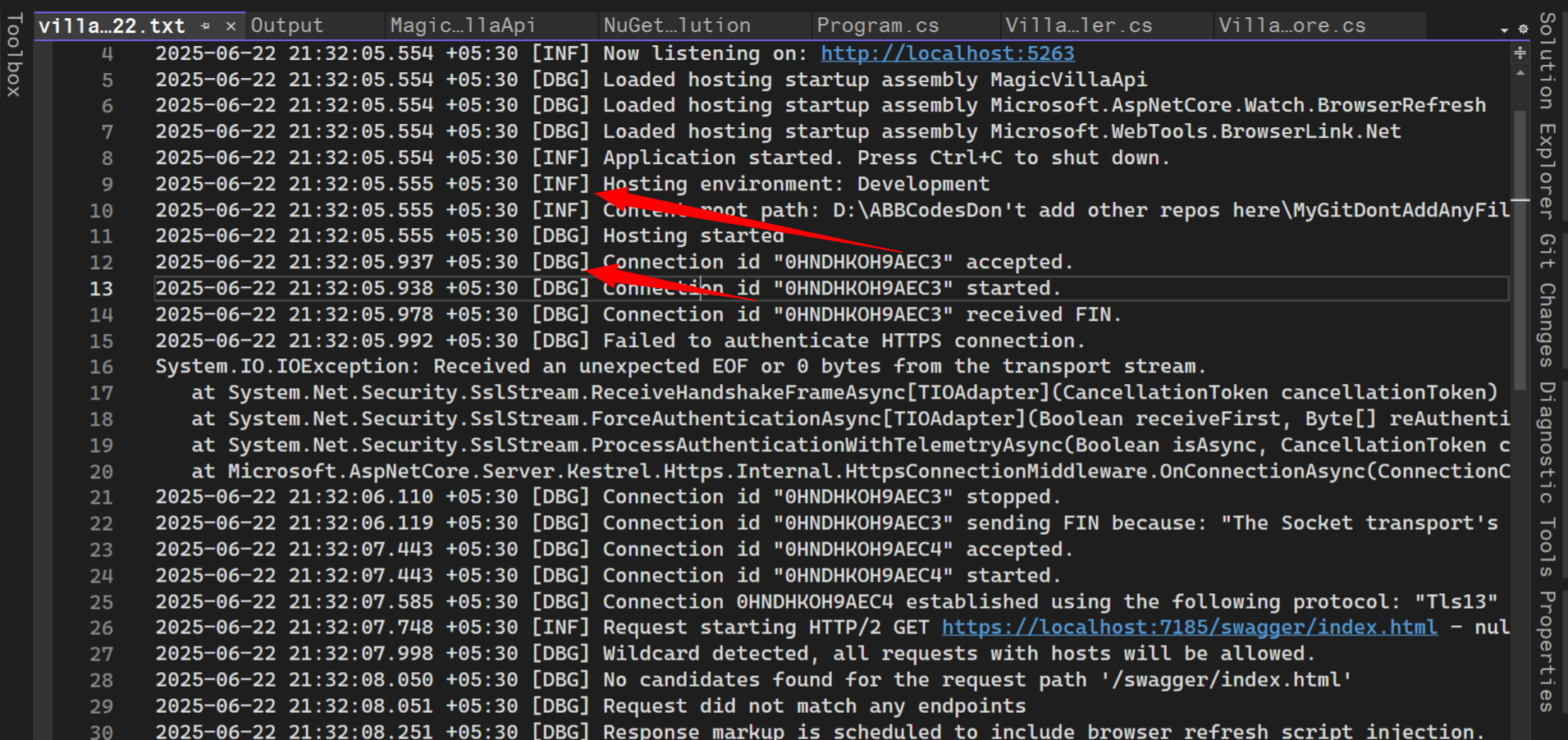
Log.Logger = new LoggerConfiguration().MinimumLevel.Debug()

.WriteTo.File("log/villaLogs.txt",rollingInterval:RollingInterval.Day).CreateLogger();

Above code when we write inside program.cs file now when we run app we see **log** folder created and **inside that we see one logfile villalogTimestamp.txt**



**When your** app is opened you can’t open log file you have to close app then only above log file you can access



**We can** see different log levels here dbg debug and inf infor

We comment that code and use default implementation

1:43:00

#. Custom logging with DI depend inject.

In controller implement custom logger

He created new folder Loggining inside that created interface ILogging

After that added new class inside logging folder class name: Logging

Inside Ilogging one method log parameters of log method log message and type: like error info

If we want to implement custom logging like above we have to register that inside program.cs file using builder.services

You have read in .net core multiple lifetime: singeton, transient, scoped now we register using scoeped lifetime

Maximum or longest lifetime is singeton

It’s created when app starts that object used every time application requests an implementation

Scoped: is basically for every request it’ll create a new object and provide that where it’s requested

For logging we use singleton as we need one logger that’ll be used throughout the app

builder.Services.AddSingleton<ILogging, Logging>();

Like above we use

Now run the app: as we used logging in

Now what is that use of DI with program.cs: first we created Logging class inside Logging folder with we implemented Log method

Now we do some changes we create new LoggingV2.cs class file with new changes in that class

Now to apply this changes to entire app

builder.Services.AddSingleton<ILogging, LoggingV2>();

we need to put LoggingV2 inside program.cs file that’s it new changes in log applied to entire app

#.real world scenario add database to our application 1:52:00

We use entityframework core which orm in this app

We have villa model so we need villa table

When we create table there must be primary key

Inside model class for variable decorate [key] it becomes primary key

#. Follow same step connect to db:

Because currently I connect to db in office laptop when you do in personal laptop computer number changes

With ef core whatever variable in villa class for that column will be created in db

Primary key:[Key]

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

above code will manage id for us automatically we don’t have to handle it

when we work with ef core we don’t work with db directly everything will be done by code itself

this is called: code first approach

with efcore: it has dbcontext it manages all db entities in our app

when we do like below

public class ApplicationDbContext : DbContext

{

public DbSet<Villa> Villas { get; set; }

}

In sqlserver table created with name Villas

When querying dbset we use linq statements those linq statements automatically translated to sql queries by ef core

**#. Link** application dbcontext to use the connection string and register in program.cs

**Link** application dbcontext to use the connection string and register in program.cs **and w**e have to register this to dependency injection

To add things in container we add in program.cs file

builder.Services.AddDbContext<ApplicationDbContext>();

above ApplicationDbContext is class which we created inside Data folder

next we provide options with the coneection string

next using helper method we get connection string and pass it to usesqlserver

below

builder.Services.AddDbContext<ApplicationDbContext>(option =>

{

option.UseSqlServer(builder.Configuration.GetConnectionString("DefaultSQLConnection"));

});

is code for register connection string

DefaultSQLConnection present in appsettings.json file

builder.Configuration.GetConnectionString("DefaultSQLConnection") will fetch connection string and pass it to usesqlserver

some basic steps needed to configure ef core:

\*\* now go back to ApplicationDbContext we have that dbcontext but we’ve to pass that connection string to dbcontext as well because we use basic features of dbcontext

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options) { }

public DbSet<Villa> Villas { get; set; }

}

Above constructor very imp for ‘add-migration’ script to work or else it gives error

Next in ApplicationDbContext we add constructor where we expect db context options we pass that options to base class which is dbcontext these are basic steps need to configure ef core in any dotnet app

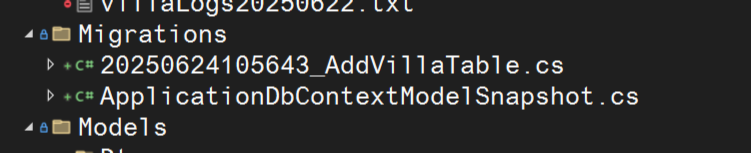
Above steps is same in all .net core web app to configure entity framework core let it be web api, mvc, blazor

#. In code first approach we have to do migration so it entityframework changes reflects in db

We create set of script that tells db that you need to create this table

To create script we have command

Add-migration AddVillaTable



We now see this

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Villas",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Name = table.Column<string>(type: "nvarchar(max)", nullable: false),

Details = table.Column<string>(type: "nvarchar(max)", nullable: false),

Rate = table.Column<double>(type: "float", nullable: false),

Sqft = table.Column<int>(typ

Migration file looks like this

Above image see it has createtable command it’s not sql query statement it’s linq statement

Based on it ef core auto creates table and all the query

It has 2 methods up and down

protected override void Up(MigrationBuilder migrationBuilder)

protected override void Down(MigrationBuilder migrationBuilder)

above in migration

up means what it basically needs to do

if up fails for some reason it rollbacks the changes

next how to apply the migration?

To apply migration we go to package manager console run command: update-database

I got error :

<https://stackoverflow.com/questions/76394279/scaffold-dbcontext-culturenotfoundexception-only-the-invariant-culture-is-sup>

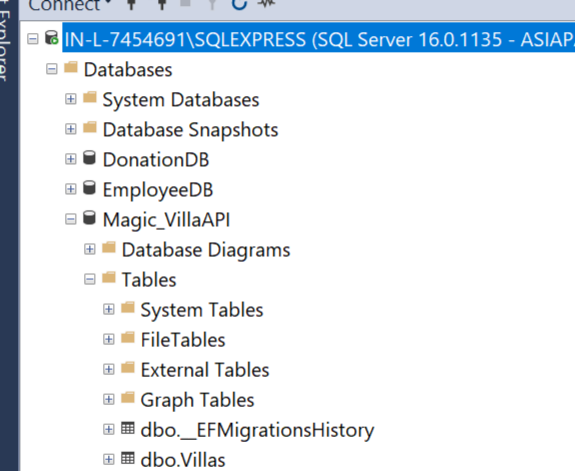
solved using above link

<InvariantGlobalization>false</InvariantGlobalization>

Set

But in video he got error: ssl provider error

update-database command: first check if database is present if it’s not present it creates that



Like above tables created

What is this \_\_EFMigrationsHistory thable: it tracks also synchronize migrations we have in our code vs db

It tells which migration applied

We don’t have to touch db to make any db changes

Time: 2:11:43

Data folder villastore delete it

#. How to see some record in db table with migration

We do in applicationdbcontext where we have dbset we have one method that we can override our migration

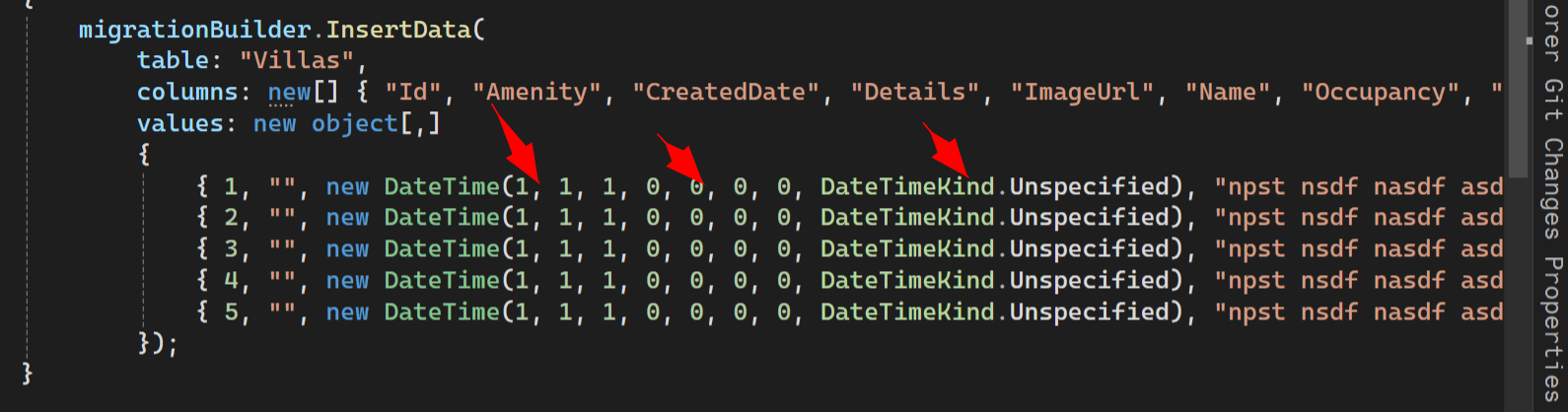
Inside ApplicationDbContext file 5 new Villa(){} data I added

Next again go to package manager console

* add-migration SeedVillaTable

A screenshot of a computer

AI-generated content may be incorrect.



In this above seedVilla migration datetime is not valid

Once above seevillatable migration added we again did change that is added CreatedDate column in ApplicationDbContext

CreatedDate = DateTime.Now

We already added SeedVillaTable migration but in previous migration DateTime was not valid

So now in applicationdbcontext we added createddate so we updated it

**Now should I delete old migration and add new migration**

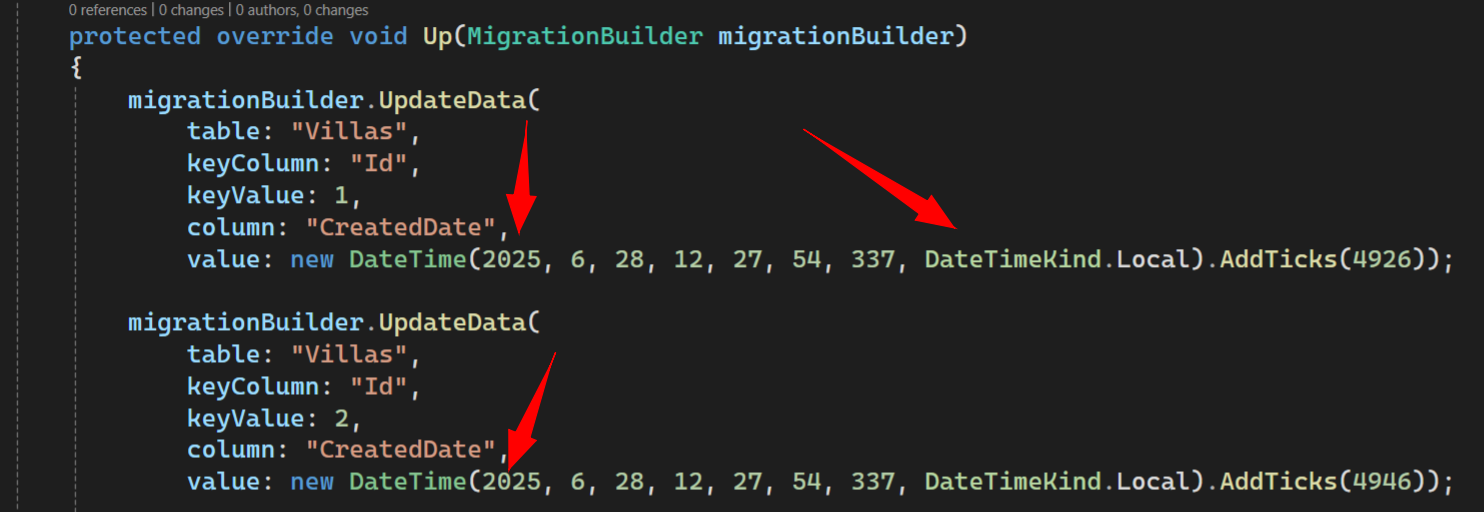
You’ll never delete any migration unless you’re confident on what you do because that will mess up every thing with EF CORE

After updating appdbcontext by adding new column createddate: we add new migration

add-migration SeedVillaTableWithCreatedDate

above command created new migration

it does update date for all the row



Updated datetime for all the row it only updated datetime because of migration

update-database

Next run command above

#. Using applicationdbcontext in our villaapicontroller earlier we use villastore for saving data

Earlier in villaapicontroller we used villastore for saving data

Now we need to use dbcontext

Delete villastore we don’t use it

We use applicationdbcontext we added that in our container in program.cs file because we have registered our service in our container

We can extract that using dependency injection

Create readonly applicationdbcontext inside villaapicontroller

private readonly ApplicationDbContext \_db;

public VillaAPIController(ApplicationDbContext db)

{

\_db = db;

}

Above code DI

NEXT GetVillas method

public ActionResult<IEnumerable<VillaDTO>> GetVillas()

{

return Ok(VillaStore.villaList);

}

Instead of above code we use below code with \_db

When you type \_db it shows all db sets

return Ok(\_db.Villas.ToList());

Like above sql query is not needed

Get villa with particular id

var villa = \_db.Villas.FirstOrDefault(x => x.Id == id);

above

in createvilla

now

public ActionResult<VillaDTO> CreateVilla([FromBody]VillaDTO villaDTO)

{

if (\_db.Villas.FirstOrDefault(u=>u.Name.ToLower() == villaDTO.Name.ToLower()) != null)

{

ModelState.AddModelError("", "Villa already Exists!");

return BadRequest(ModelState);

}

if(villaDTO == null)

{

return BadRequest(villaDTO);

}

if(villaDTO.Id > 0) {

return StatusCode(StatusCodes.Status500InternalServerError);

}

Villa model = new()

{

Amenity = villaDTO.Amenity,

Details = villaDTO.Details,

Id = villaDTO.Id,

ImageUrl = villaDTO.ImageUrl,

Name = villaDTO.Name,

Occupancy = villaDTO.Occupancy,

Rate = villaDTO.Rate,

Sqft = villaDTO.Sqft

};

\_db.Villas.Add(model);

\_db.SaveChanges();

return CreatedAtRoute("GetVilla", new {id=villaDTO.Id}, villaDTO);

}

here

#. Next testing

While testing he faced issues with patch request and also he solved it