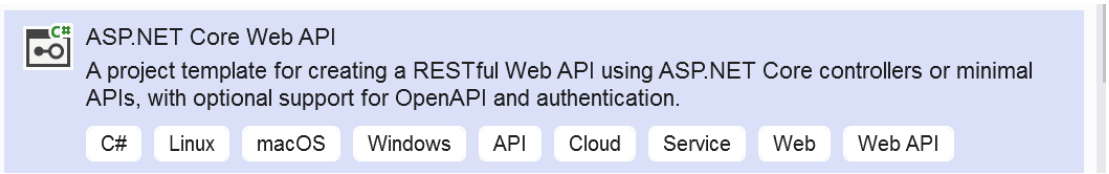
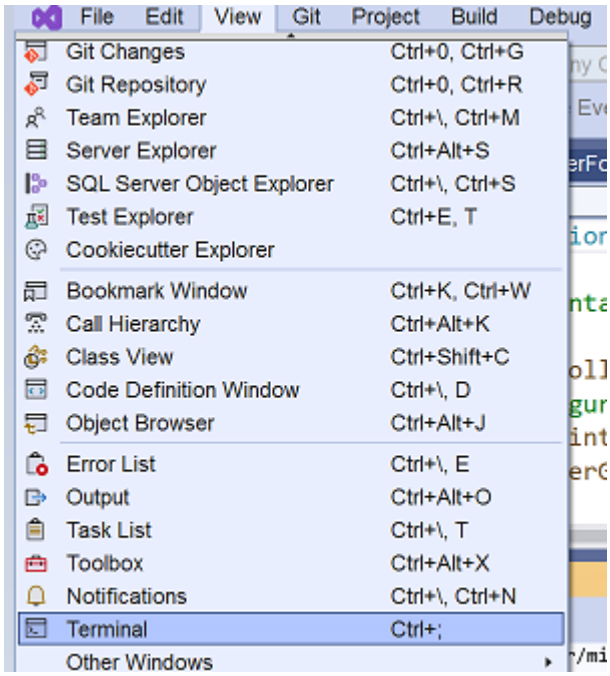
Step 1:



Step 2:



https://medium.com/@ravipatel.it/a-beginners-guide-to-entity-framework-core-ef-core-5cde48fc7f7a

Step 3:

From terminal (CLI):

dotnet new console -n EFCoreInterview //create a new project

cd EFCoreInterview

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

Now:

create a C# class to represent the data

For this example, we'll create a **Student** class:

Student.cs

```
namespace EFCoreExample
{
   public class Student
   {
      public int Id { get; set; } ;//you need ID anyway
      public string Name { get; set; }
      public int Age { get; set; }
```

```
    }
}
```

Create a DbContext (auto – you don't need create DB in SQLServer it's created alone)

AppDbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace EFCoreExample
{
    public class AppDbContext : DbContext
    {
        public DbSet<Student> Students { get; set; }//this is name of the table

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Server=
(local);Database=EFCoreExampleDB;Trusted_Connection=True; TrustServerCertificate=True
");
        }
    }
}
```

Note also can: Server= (localdb)\mssqllocaldb or example's connectionString connectionString="metadata=res://*/CME.csdl|res://*/CME.ssdl|res://*/CME.msl;provider=System.Data.SqlClient;provider connection string="data source=nameOfServer;initial catalog=CME.Form_Dev;persist security info=True;user id=sa;password=abcd;MultipleActiveResultSets=True;App=EntityFramework""

Program.cs

```
using Microsoft.EntityFrameworkCore;
using WebApplication1.Models;

var builder = WebApplication.CreateBuilder(args);
//DI for DbContext
builder.Services.AddDbContext<WeatherForecastDBContext>(options =>
options.UseSqlServer(@"Server=
(local);Database=EFCoreExampleDB;Trusted_Connection=True; TrustServerCertificate=True
"));
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
```

```
builder.Services.AddOpenApi();



var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.MapOpenApi();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Next step:
dotnet add package Microsoft.EntityFrameworkCore.Tools
dotnet ef migrations add InitialCreate
dotnet ef database update

Next step:
dotnet run
open site :
https://localhost:44302/weatherforecast

Very important!
Migrations allow you to update your database schema as **your model changes**.
We save all history of migrations so we write something like that:
because we add new field in student model that his name Timestamp
after that we write in CLI
dotnet ef migrations add AddStudentCreatedTimestamp
dotnet ef database update

Queries:

```
// Adding a new student
var student = new Student { Name = "John Doe", Age = 20 };
context.Students.Add(student);
context.SaveChanges();

// Querying the student (get from DB)
var query = context.Students.Where(s => s.Name == "John Doe");

foreach (var stud in query)
{
    Console.WriteLine($"Student: {stud.Name}, Age: {stud.Age}");
}
```

Updating Data

```
var student = context.Students.First(s => s.Name == "Alice");
student.Age = 23;
context.SaveChanges();
```

Deleting Data

```
var student = context.Students.First(s => s.Name == "Alice");
context.Students.Remove(student);
context.SaveChanges();
```

Get all students:

```
var students = context.Students.ToList();
```

Filter students by age:

```
var students = context.Students.Where(s => s.Age > 20).ToList();
```

Get a single student by ID:

```
var student = context.Students.Find(1);
```

controller:

```
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
```

```
namespace WebApplication1.Controllers
{


    [ApiController]
    [Route("[controller]")] //or if you want [Route("api/GetWeatherForecast")]
    public class WeatherForecastController : Controller
    {
        private static readonly string[] Summaries = new[]
        {
            "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering",
"Scorching"
        };

         private readonly WeatherForecastDBContext _context;
        public WeatherForecastController(WeatherForecastDBContext  context)
        {
            _context = context;
        }


         [HttpGet("/bb/{id:int}")]  // GET /bb/2
         public IEnumerable<WeatherForecast> Get2(int id)
        {
            Console.WriteLine("test{0}", id);

            _context.wf.Add(new WeatherForecast
            {
                Date = DateOnly.FromDateTime(DateTime.Now.AddDays(5)),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            });
            _context.SaveChanges();
            Console.WriteLine("test2");
            return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            })
            .ToArray();
        }
```

```
    [HttpGet(Name = "GetWeatherForecast")]
    public IEnumerable<WeatherForecast> Get()
    {

        _context.wf.Add(new WeatherForecast {
            Date = DateOnly.FromDateTime(DateTime.Now.AddDays(5)),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        });
        _context.SaveChanges();
        Console.WriteLine("test");
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        })
        .ToArray();

    }


    // DELETE: api/DCandidate/5
    [HttpDelete("{id}")]
    public async Task<ActionResult<WeatherForecast>> DeleteDCandidate(int id)
    {
        var dCandidate = await _context.wf.FindAsync(id);
        if (dCandidate == null)
        {
            return NotFound();
        }

        _context.wf.Remove(dCandidate);
        await _context.SaveChangesAsync();

        return dCandidate;

    }


    // POST: api/DCandidate
    // To protect from overposting attacks, please enable the specific properties you want to
bind to, for
    // more details see https://aka.ms/RazorPagesCRUD.
```

```
        [HttpPost]
        public async Task<ActionResult<WeatherForecast>> PostDCandidate(WeatherForecast
dCandidate)
        {
            _context.wf.Add(dCandidate);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetDCandidate", new { id = dCandidate.Id }, dCandidate);
        }



    }
}
```

Controllers

SalesCustomersController.cs

```
[Route("api/cities")]
[ApiController]
        public class SalesCustomersController : Controller
        …
          // GET: SalesCustomers/AddOrEdit
        [HttpGet("{id}")]
          public IActionResult AddOrEdit(int id = 0)
        …
              // POST: SalesCustomers/AddOrEdit
      [HttpPost]
      public async Task<IActionResult>

…
    // PUT: api/DCandidate/5
      [HttpPut("{id}")]
      public async Task<IActionResult> PutDCandidate(int id, DCandidate dCandidate)


      // DELETE: api/DCandidate/5
      [HttpDelete("{id}")]
      public async Task<ActionResult<DCandidate>> DeleteDCandidate(int id)

      // GET: api/DCandidat/Search
      [HttpGet("Search/{name}")]
```

```
public async Task<ActionResult<DCandidate>> SearchCity(string name)
```