<https://docs.microsoft.com/en-us/learn/modules/build-web-api-minimal-database/?wt.mc_id=academic-0000-chnoring>

EF Core is a lightweight, extensible, open source, and cross-platform data access technology for .NET applications.

EF Core can serve as an object-relational mapper, which:

* Enables .NET developers to work with a database by using .NET objects.
* Eliminates the need for most of the data-access code that typically needs to be written.

EF Core supports a large number of popular databases, including SQLite, MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.

## Perform CRUD operations with EF Core

The context class is responsible for querying and saving data to your entity classes, and for creating and managing the database connection.

|  |  |
| --- | --- |
| Query data | var pizzas = await db.Pizzas.ToListAsync(); |
| Insert | await db.pizzas.AddAsync(  new Pizza { ID = 1, Name = "Pepperoni",  Description = "The classic pepperoni pizza" }); |
| Delete | var pizza = await db.pizzas.FindAsync(id);  if (pizza is null)  {  //Handle error  }  db.pizzas.Remove(pizza); |
| Update | int id = 1;  var updatepizza = new Pizza { Name = "Pineapple", Description = "Ummmm?" })  var pizza = await db.pizzas.FindAsync(id);  if (pizza is null)  {  //Handle error  }  pizza.Item = updatepizza.Item;  pizza.IsComplete = updatepizza.IsComplete;  await db.SaveChangesAsync(); |
|  |  |

## Use the EF Core in-memory database

EF Core includes an in-memory database provider that can be used to test your application.

This module uses the .NET 6.0 SDK. Ensure that you have .NET 6.0 installed by running the following command in your preferred terminal:

New terminal

Create a web API by running dotnet new:

cd..

cd temp

dotnet new web -o PizzaStore -f net6.0

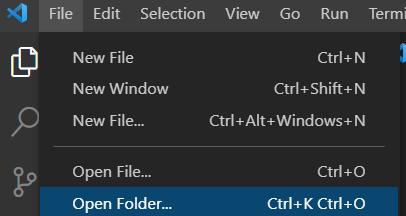
cd PizzaStore

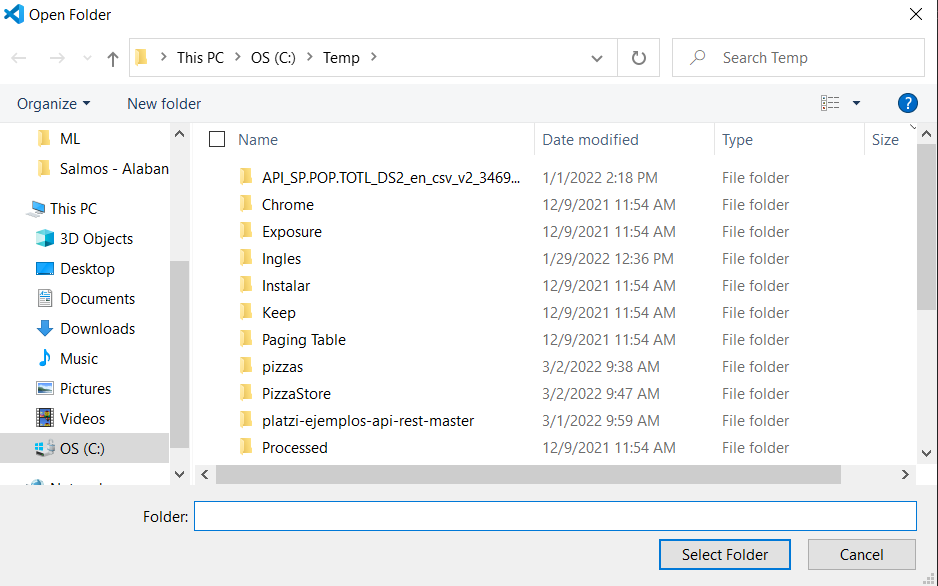
|  |  |
| --- | --- |
| Install the Swashbuckle package: | dotnet add package Swashbuckle.AspNetCore --version 6.2.3 |
| Install the EntityFrameworkCore.InMemory package: | dotnet add package Microsoft.EntityFrameworkCore.InMemory --version 6.0 |

Create a *Pizza.cs* file and give it the following content:

(create folder models if you want)

Open Folder





**The model**

using Microsoft.EntityFrameworkCore;

namespace PizzaStore.Models

{

    public class Pizza

    {

      public int Id { get; set; }

      public string? Name { get; set; }

      public string? Description { get; set; }

  }

}

The preceding Pizza class is a simple object that represents a pizza. This code is your data model.

Open *Program.cs* and add the highlighted code:

using Microsoft.OpenApi.Models;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo {

Title = "PizzaStore API", Description = "Making the Pizzas you love", Version = "v1"

});

}

);

var app = builder.Build();

app.UseSwagger(); app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "PizzaStore API V1");

}

);

app.MapGet("/", () => "Hello World!");

app.Run();

You might receive a prompt from Visual Studio Code to add assets to debug the project. Click Yes in the dialog.

Add using Microsoft.EntityFrameworkCore; to the top of your

* *Program.cs*
* and *Pizza.cs* files.

In Models folder, create a PizzaDb class. (optional name PizzaContext.cs) The PizzaDb class will do the following tasks:

* Expose your Pizzas property from your list of Pizza in the database.
* Use UseInMemoryDatabase to wire the in-memory database storage.

using Microsoft.EntityFrameworkCore;

namespace PizzaStore.Models

{

   class PizzaDb : DbContext

   {

        public PizzaDb(DbContextOptions options) : base(options) { }

        public DbSet<Pizza> Pizzas { get; set; }

    }

}

DbContext represents a connection or session that's used to query and save instances of entities in a database.

Your program.cs must have

Add using PizzaStore.Models; to the top of your Program.cs file.

In *Program.cs*, before the call to AddSwaggerGen, add the following code:

builder.Services.AddDbContext<PizzaDb>(options => options.UseInMemoryDatabase("items"));

To read from a list of items in the pizza list, add the following code above the call to app.Run(); to add a "/pizzas" route:

app.MapGet("/pizzas", async (PizzaDb db) => await db.Pizzas.ToListAsync());

using Microsoft.OpenApi.Models;

using Microsoft.EntityFrameworkCore;

using PizzaStore.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddDbContext<PizzaDb>(options => options.UseInMemoryDatabase("items"));

builder.Services.AddSwaggerGen(c =>

    {

          c.SwaggerDoc("v1", new OpenApiInfo {

                Title = "PizzaStore API",

Description = "Making the Pizzas you love",

Version = "v1"

          });

     }

);

var app = builder.Build();

app.UseSwagger(); app.UseSwaggerUI(c =>

    {

          c.SwaggerEndpoint("/swagger/v1/swagger.json", "PizzaStore API V1");

     }

);

app.MapGet("/", () => "Hello World!");

app.MapGet("/pizzas", async (PizzaDb db) => await db.Pizzas.ToListAsync());

app.Run();

**Run the application**

* dotnet restore (guarde antes todo)
* dotnet build
* dotnet run

This action will build the app and host it on a port from 5000-5300.

HTTPS will have a port selected for it in the range 7000-7300.

**Note**

If you want to override the random port selection behavior, you can set the ports to use in *launchSettings.json*.

**dotnet run**

Si da error

error CS1022: Type or namespace definition, or end-of-file expected [C:\Temp\PizzaStore\PizzaStore.csproj]

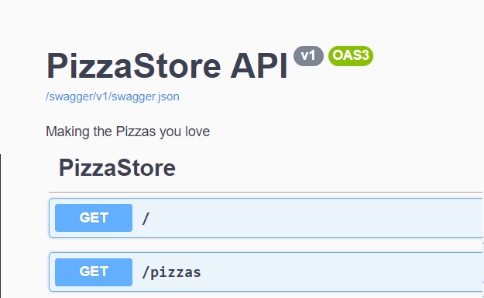
* Guarde todo
* dotnet restore
* dotnet build
* dotnet run

Use segundo link porque primero es https y el segundo http

<http://localhost:5071/>

Hello World!

<http://localhost:5071/>swagger



Select the GET /pizzas button, and you'll see that the list is empty under Response body.

## Create new items

Let's POST new tasks to the pizzas list under app.MapGet that you created earlier, in program.cs

app.MapPost("/pizza", async (PizzaDb db, Pizza pizza) =>

{

    await db.Pizzas.AddAsync(pizza);

    await db.SaveChangesAsync();

    return Results.Created($"/pizza/{pizza.Id}", pizza);

});

app.Run();

* Guarde todo
* dotnet restore
* dotnet build
* dotnet run
* <http://localhost:5071/swagger>
* Select **POST / pizza**
* Select **Try it out**

|  |  |
| --- | --- |
| Replace the request body with the following:  {  "name": "Pepperoni",  "description": "A classic pepperoni pizza"  }  Select **Execute**. |  |
| {  "Id": 2,  "name": "Vegetariana",  "description": "Solo vegetales"  } |  |
| To read the items in the list:   1. Select **GET /pizzas**. 2. Select **Try it out**. 3. Select **Execute**. | [  {  "id": 1,  "name": "Pepperoni",  "description": "A pepperoni pizza"  }  ] |
| http://localhost:5071/pizzas | [{"id":1,"name":"Pepperoni","description":"A classic pepperoni pizza"},  {"id":2,"name":"Pepperoni2","description":"A cl2assic pepperoni pizza"}] |

To update / delete in program.cs

app.MapGet("/", () => "Hello World!");

// LIST

app.MapGet("/pizzas", async (PizzaDb db) => await db.Pizzas.ToListAsync());

// ONE ITEM

app.MapGet("/pizza/{id}", async (PizzaDb db, int id) => await db.Pizzas.FindAsync(id));

// INSERT

app.MapPost("/InsPizza", async (PizzaDb db, Pizza pizza) =>

{

    await db.Pizzas.AddAsync(pizza);

    await db.SaveChangesAsync();

    return Results.Created($"/pizza/{pizza.Id}", pizza);

});

// UPDATE

app.MapPut("/UpdPizza/{id}", async (PizzaDb db, Pizza updatepizza, int id) =>

{

    var pizza = await db.Pizzas.FindAsync(id);

    if (pizza is null) return Results.NotFound();

    pizza.Name = updatepizza.Name;

    pizza.Description = updatepizza.Description;

    await db.SaveChangesAsync();

    return Results.NoContent();

});

// DELETE

app.MapDelete("/DelPizza/{id}", async (PizzaDb db, int id) =>

{

  var pizza = await db.Pizzas.FindAsync(id);

  if (pizza is null)

  {

    return Results.NotFound();

  }

  db.Pizzas.Remove(pizza);

  await db.SaveChangesAsync();

  return Results.Ok();

});

app.Run();

|  |  |
| --- | --- |
| http://localhost:5071/pizza/2 | **{**  **"id": 2,**  **"name": "pizza1",**  **"description": "modalidad"**  **},** |
| http://localhost:5071/pizzas | **[**  **{**  **"id": 1,**  **"name": "string",**  **"description": "string"**  **},**    **{**  **"id": 3,**  **"name": "pizza1",**  **"description": "modalidad"**  **}**  **]** |

DELETE

<http://localhost:5071/DelPizza/1>

UPDATE

<http://localhost:5071/UpdPizza/2?Id=2&Name=Veremos&Description=ojala>

Carga de datos inicial

app.MapPost("/InicioPizza", async (PizzaDb db, Pizza pizza) =>

{

    Pizza pizza1 = new Pizza { Id = 1, Name = "pizza1", Description="another" };

    await db.Pizzas.AddAsync(pizza1);

    await db.SaveChangesAsync();

    Pizza pizza2 = new Pizza { Id = 2, Name = "pizza2", Description="another222" };

    await db.Pizzas.AddAsync(pizza2);

    await db.SaveChangesAsync();

    return Results.Created($"/pizza/{pizza.Id}", pizza);

});