[FromQuery] - Gets values from the query string.

[FromRoute] - Gets values from route data.

[FromForm] - Gets values from posted form fields.

[FromBody] - Gets values from the request body.

[FromHeader] - Gets values from HTTP headers.

[FromServices]

Code First Approach

Add-Migration

Database-Update

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

DataFirst Approach

1)Scaffold-DBContext "Name=ConnectionStrings:DefaultConnection" Microsoft.EntityFrameworkCore.SqlServer

or

Scaffold-DBContext "Name=ConnectionStrings:DefaultConnection" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -ContextDir Data

Scaffold-DBContext "Name=ConnectionStrings:DefaultConnection" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -ContextDir Data -Force

Partial class

Entity Framework (EF) requires each entity that you add or track to have a primary key.

The key helps EF know whether the object is new or existing in the database.

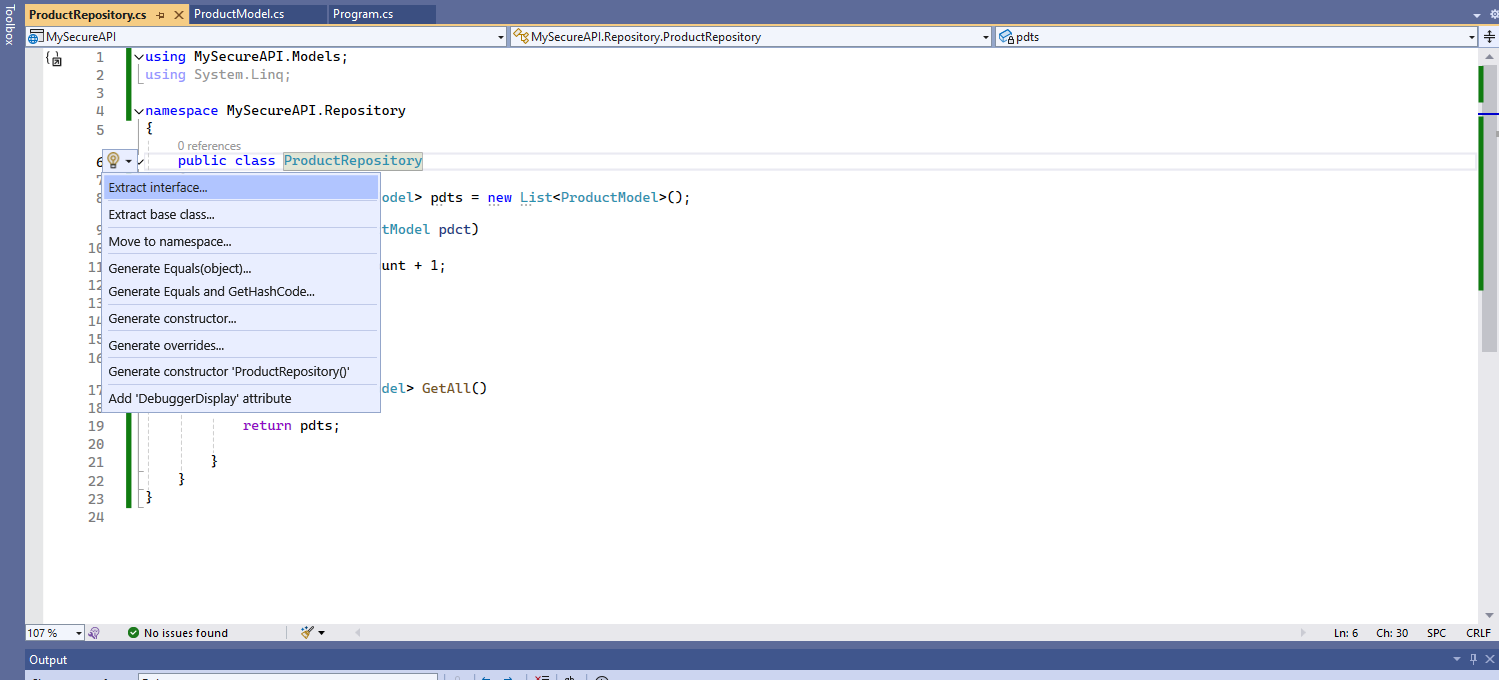
Keyless entities cannot be added or updated, only queried.

If you’re calling .Add() or .AddAsync() on it, you must have a key.

A screenshot of a computer

AI-generated content may be incorrect.

Repository



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Interface

A screenshot of a computer

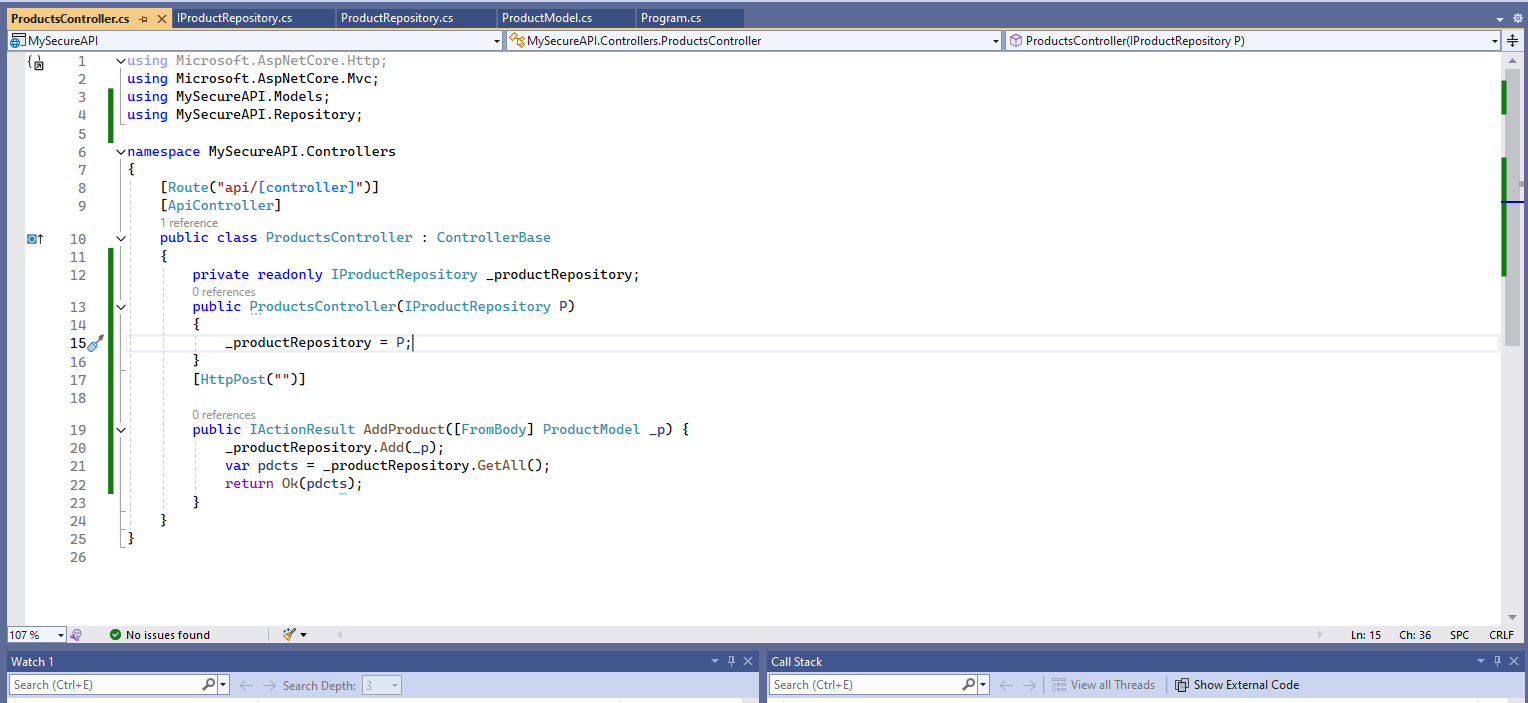
AI-generated content may be incorrect.

Product Model

A computer screen with text

AI-generated content may be incorrect.

ProductsController



In Startup.cs (In new version its in program.cs) -DI-Singleton ---

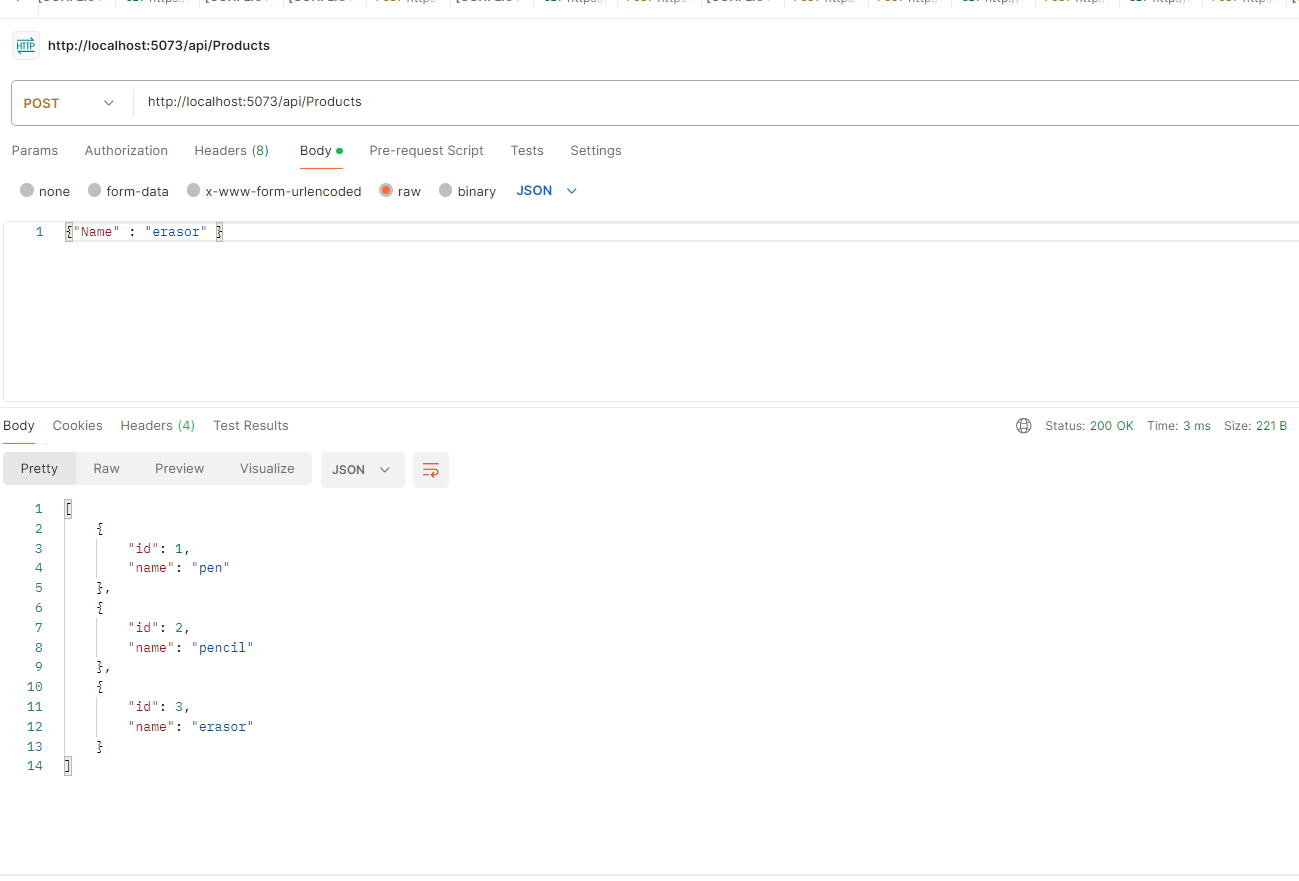
A computer screen with text

AI-generated content may be incorrect.

So in controller we are not adding each time **new** Repository, we are just assigning Interface.

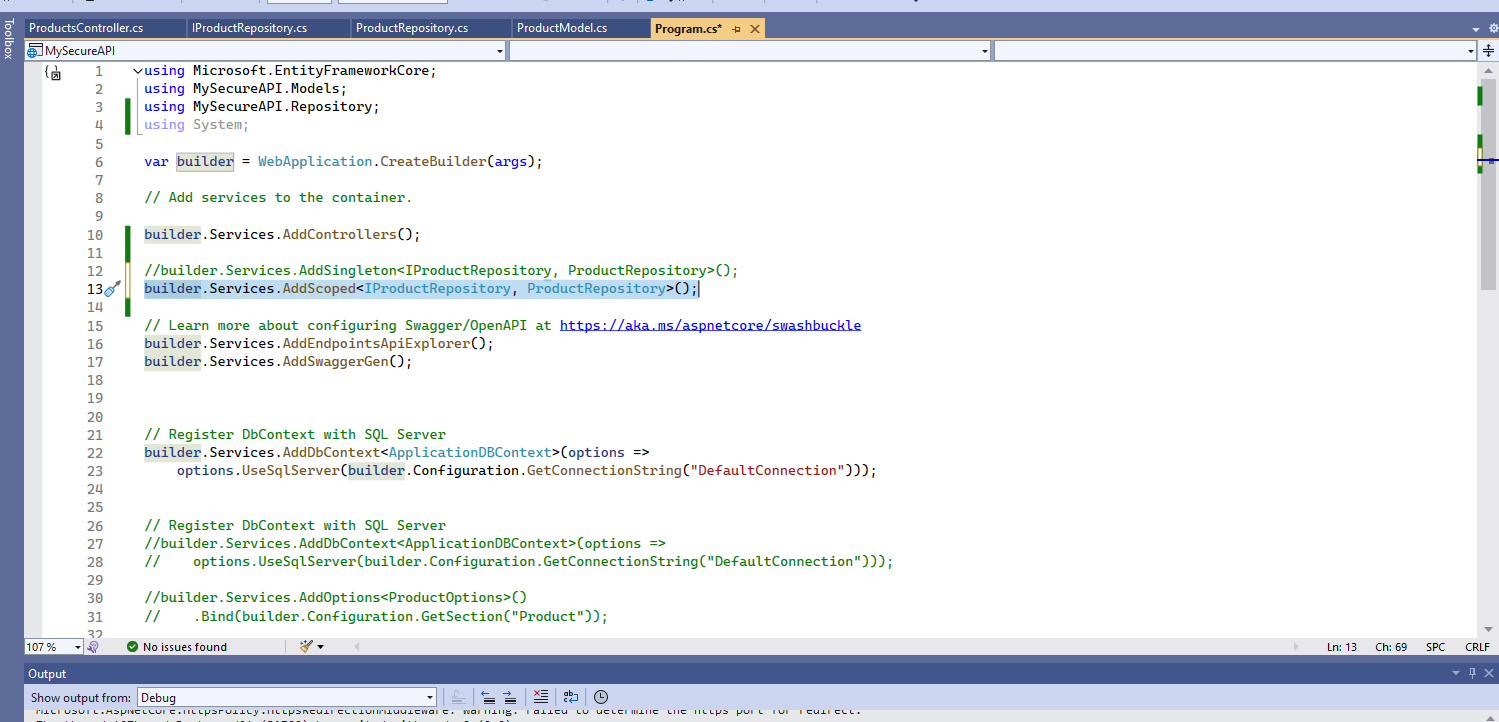
A screenshot of a computer

AI-generated content may be incorrect.



Each time we call this url, each product get added – its appending to existing product list. Singleton- 1 instance through out.

Scoped Service

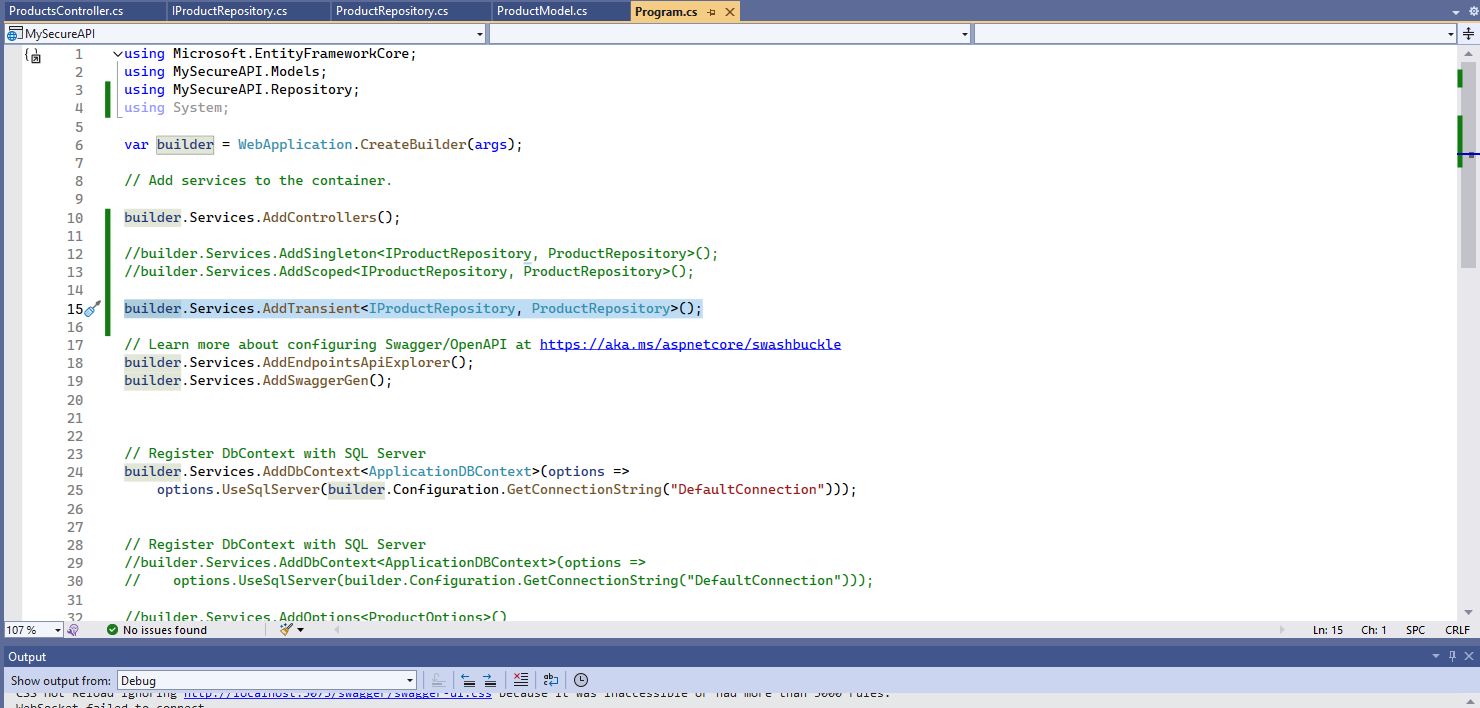


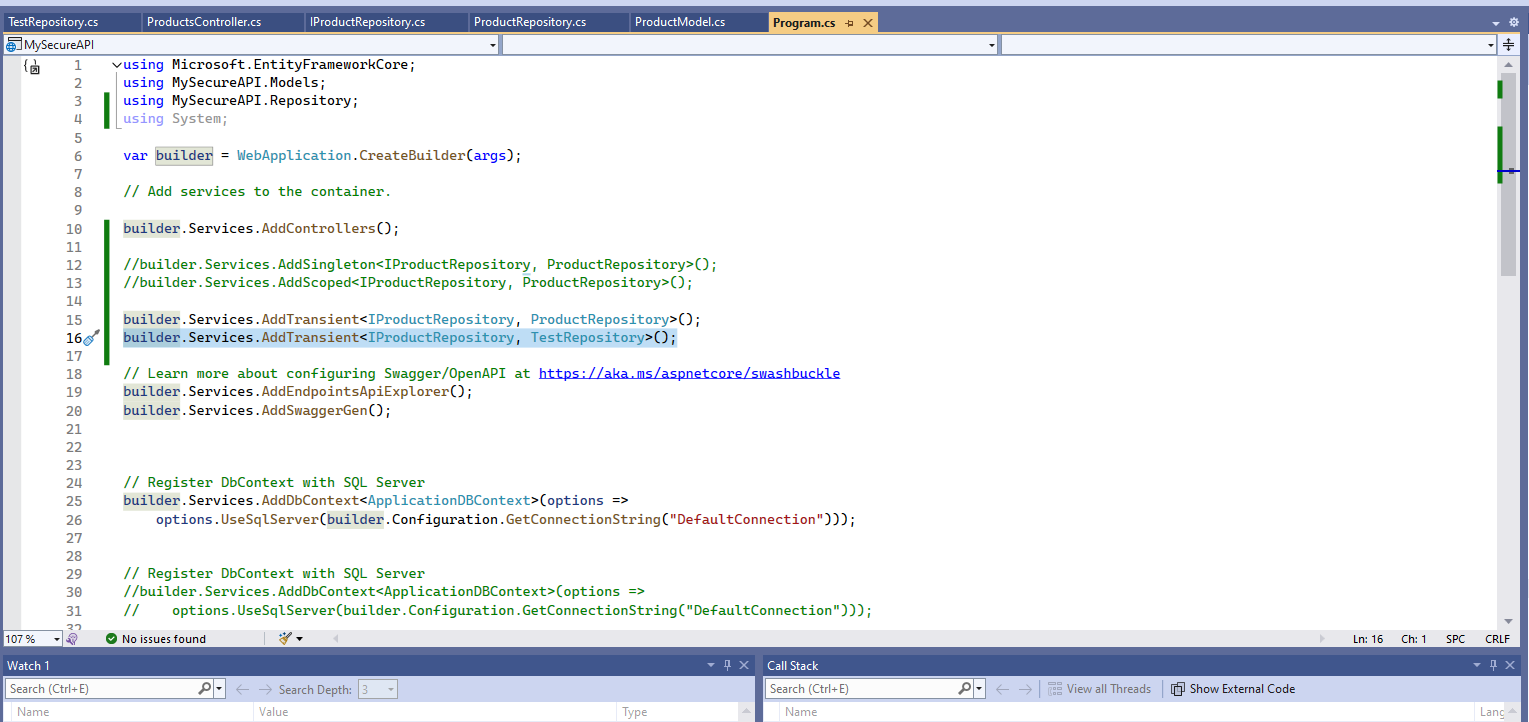
A screenshot of a computer

AI-generated content may be incorrect.

Each time, url refresh, add item its id becoming 1 because fresh product list.

Add Transient method





A screenshot of a computer

AI-generated content may be incorrect.

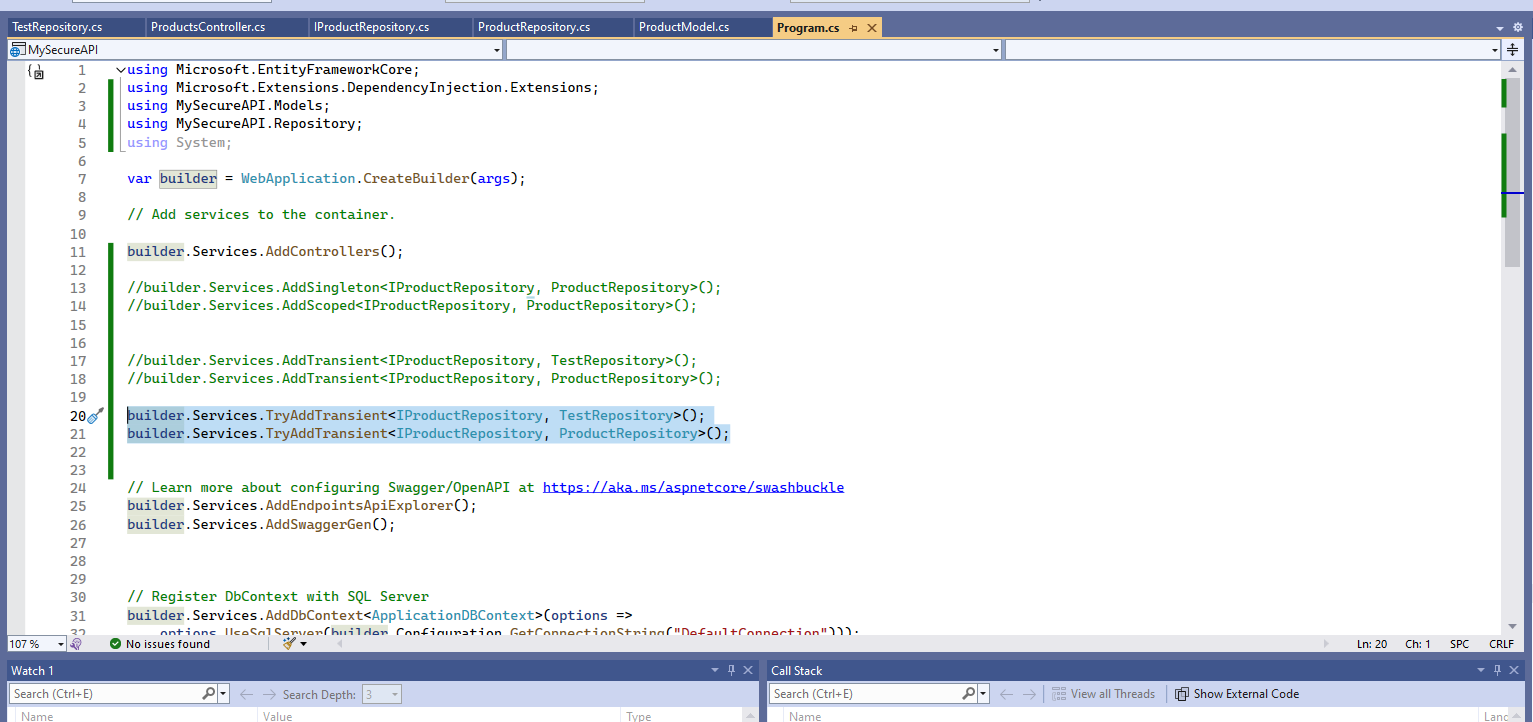
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

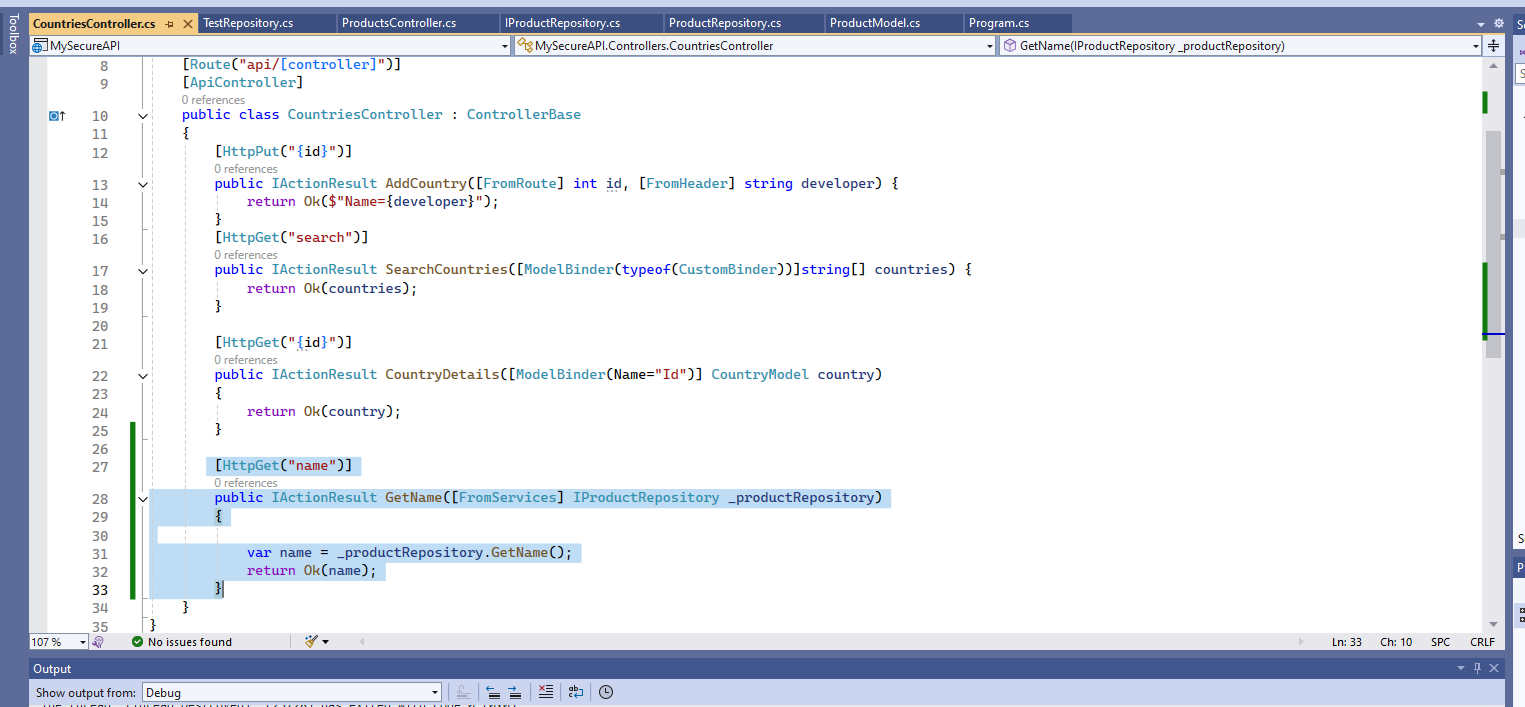
TryAddTransient



A screenshot of a computer

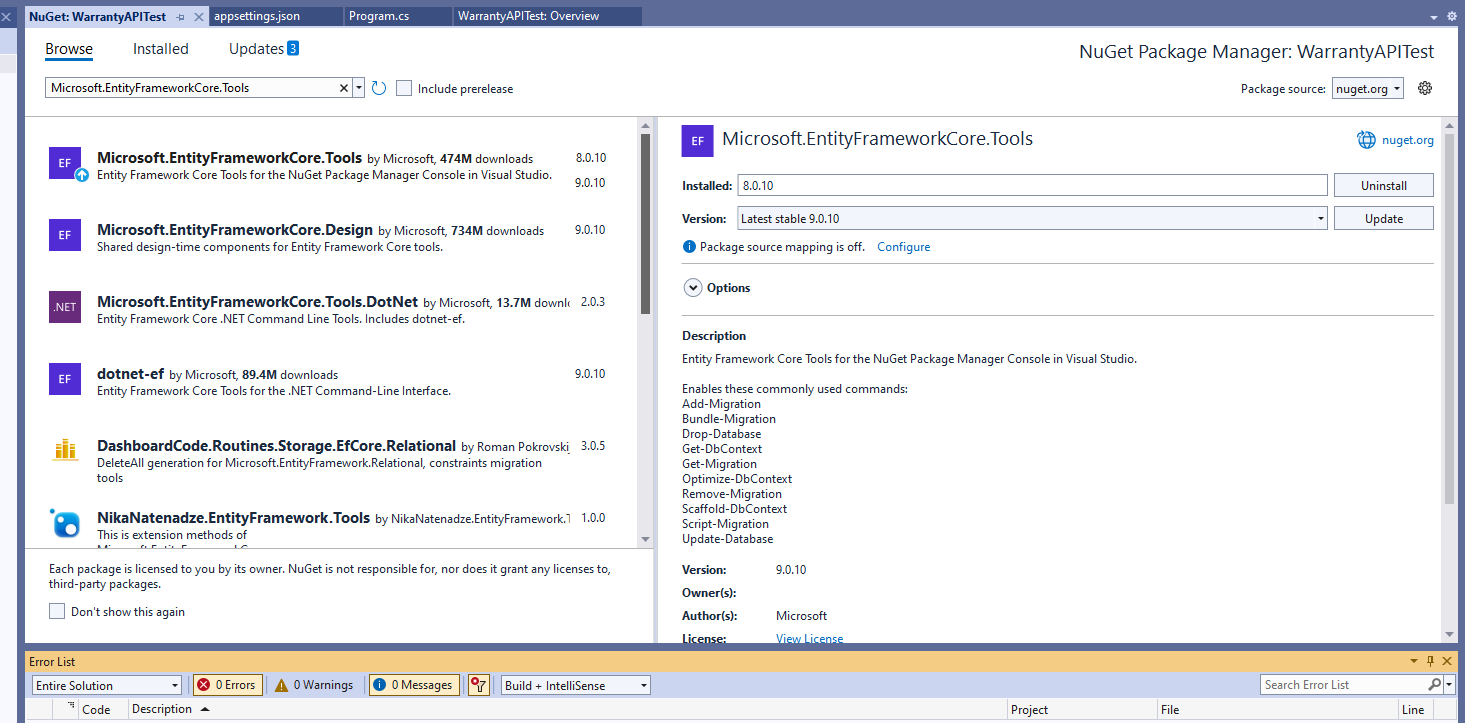
AI-generated content may be incorrect.

From Services Attribute



A screenshot of a computer

AI-generated content may be incorrect.



A screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

GET

A screenshot of a computer

AI-generated content may be incorrect.

GET BY ID

A white screen with black text

AI-generated content may be incorrect.

POST

A screenshot of a computer

AI-generated content may be incorrect.

PUT

A white background with text

AI-generated content may be incorrect.

PATCH (replace,remove)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

-A screenshot of a computer

AI-generated content may be incorrect.

DELETE

A screenshot of a computer

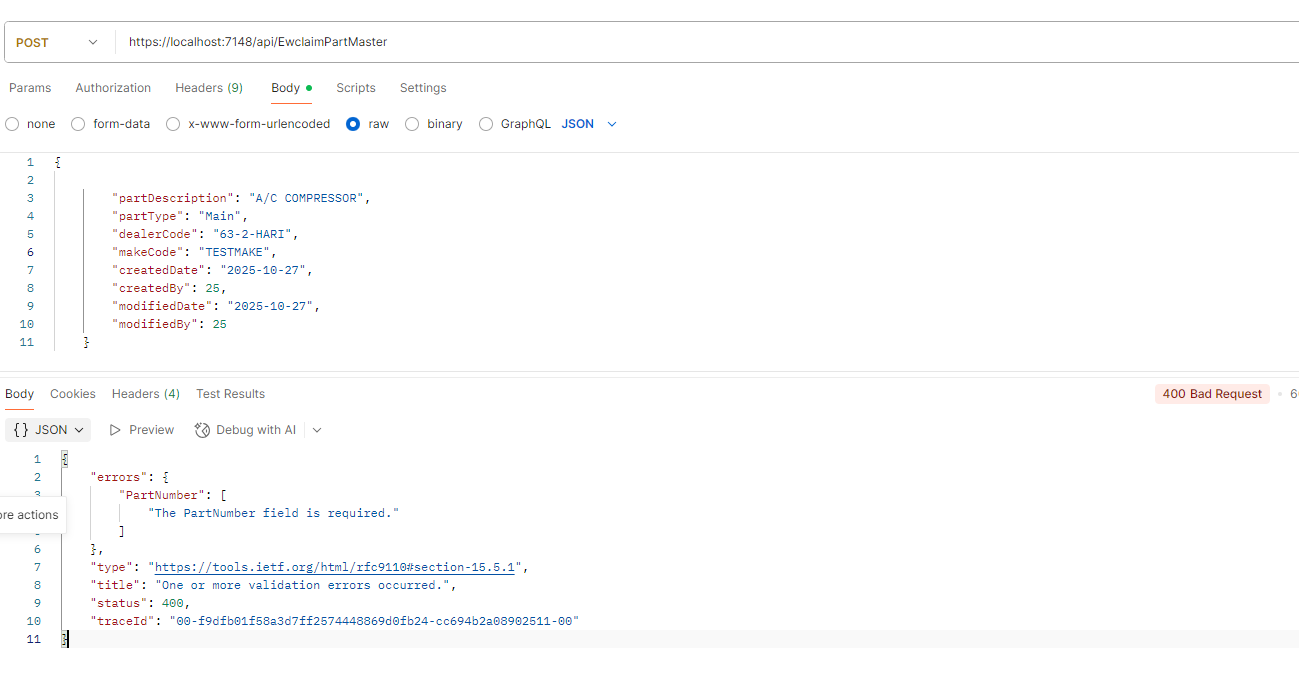
AI-generated content may be incorrect.

Validations

using System.ComponentModel.DataAnnotations;

[Required]

public string PartNumber { get; set; }



using Microsoft.AspNetCore.JsonPatch;

using WarrantyAPITest.Models;

namespace WarrantyAPITest.Interfaces

{

public interface IEwclaimPartMasterRepository

{

Task<List<EwclaimPartMaster>> GetAllAsync();

Task<EwclaimPartMaster?> GetByIdAsync(int \_partnumber);

Task<int> AddAsync(EwclaimPartMaster \_EwclaimPartMasterModel);

Task UpdateAsync(int PartID, EwclaimPartMaster \_part);

Task DeleteAsync(int Partid);

Task UpdatePatchAsync(int partid, JsonPatchDocument \_EwclaimPartMaster);

}

}

using Azure;

using Microsoft.AspNetCore.Http.HttpResults;

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.JsonPatch;

using System.Diagnostics.Eventing.Reader;

using WarrantyAPITest.Data;

using WarrantyAPITest.Interfaces;

using WarrantyAPITest.Models;

using JsonPatchDocument = Microsoft.AspNetCore.JsonPatch.JsonPatchDocument;

namespace WarrantyAPITest.Repository

{

public class EwclaimPartMasterRepository : IEwclaimPartMasterRepository

{

private readonly GaponlineDemo1Context \_context;

public EwclaimPartMasterRepository(GaponlineDemo1Context context)

{

\_context = context;

}

public async Task<List<EwclaimPartMaster>> GetAllAsync()

{

//throw new NotImplementedException();

var records = await \_context.EwclaimPartMasters.Select(x=> new EwclaimPartMaster()

{

PartId=x.PartId,

PartNumber =x.PartNumber,

PartDescription = x.PartDescription,

PartType = x.PartType,

DealerCode = x.DealerCode,

MakeCode = x.PartNumber,

CreatedDate = x.CreatedDate,

CreatedBy = x.CreatedBy,

ModifiedDate = x.ModifiedDate,

ModifiedBy = x.ModifiedBy

}

).ToListAsync();

return records;

}

//public async Task<IEnumerable<EwclaimPartMaster>> GetAllAsync()

//{

// //throw new NotImplementedException();

// var records= await \_context.EwclaimPartMasters.ToListAsync();

//}

public async Task<EwclaimPartMaster?> GetByIdAsync(int \_partnumber)

{

//throw new NotImplementedException();

var records = await \_context.EwclaimPartMasters.Where(x => x.PartId == \_partnumber).Select(x => new EwclaimPartMaster()

{

PartId = x.PartId,

PartNumber = x.PartNumber,

PartDescription = x.PartDescription,

PartType = x.PartType,

DealerCode = x.DealerCode,

MakeCode = x.PartNumber,

CreatedDate = x.CreatedDate,

CreatedBy = x.CreatedBy,

ModifiedDate = x.ModifiedDate,

ModifiedBy = x.ModifiedBy

}

).FirstOrDefaultAsync();

return records;

}

public async Task<int> AddAsync(EwclaimPartMaster \_EwclaimPartMasterModel)

{

//throw new NotImplementedException();

var \_EwclaimPartMaster = new EwclaimPartMaster()

{

PartNumber = \_EwclaimPartMasterModel.PartNumber,

PartDescription = \_EwclaimPartMasterModel.PartDescription,

PartType = \_EwclaimPartMasterModel.PartType,

DealerCode = \_EwclaimPartMasterModel.DealerCode,

MakeCode = \_EwclaimPartMasterModel.PartNumber,

CreatedDate = \_EwclaimPartMasterModel.CreatedDate,

CreatedBy = \_EwclaimPartMasterModel.CreatedBy,

ModifiedDate = \_EwclaimPartMasterModel.ModifiedDate,

ModifiedBy = \_EwclaimPartMasterModel.ModifiedBy

};

\_context.EwclaimPartMasters.Add(\_EwclaimPartMaster);

await \_context.SaveChangesAsync();

return \_EwclaimPartMaster.PartId;

}

//public Task UpdateAsync(int partID)

//{

// // throw new NotImplementedException();

// EwclaimPartMaster \_EwclaimPartMaster = GetByIdAsync(partID);

//}

public async Task UpdateAsync( int PartID, EwclaimPartMaster \_EwclaimPartMasterModel)

{

/\*

//Method 1 (2 db works : 1 - fetch ;2- update)

var existPartMaster= await \_context.EwclaimPartMasters.FindAsync(PartID);

if (existPartMaster != null) {

existPartMaster.PartId = \_EwclaimPartMasterModel.PartId;

existPartMaster.PartNumber = \_EwclaimPartMasterModel.PartNumber;

existPartMaster.PartDescription = \_EwclaimPartMasterModel.PartDescription;

existPartMaster.PartType = \_EwclaimPartMasterModel.PartType;

existPartMaster.DealerCode = \_EwclaimPartMasterModel.DealerCode;

existPartMaster.MakeCode = \_EwclaimPartMasterModel.MakeCode;

existPartMaster.CreatedDate = \_EwclaimPartMasterModel.CreatedDate;

existPartMaster.CreatedBy = \_EwclaimPartMasterModel.CreatedBy;

existPartMaster.ModifiedDate = \_EwclaimPartMasterModel.ModifiedDate;

existPartMaster.ModifiedBy = \_EwclaimPartMasterModel.ModifiedBy;

await \_context.SaveChangesAsync();

}

\*/

//Method 2 (1 db works : 1- update)

var \_EwclaimPartMaster = new EwclaimPartMaster()

{

PartId= PartID,

PartNumber = \_EwclaimPartMasterModel.PartNumber,

PartDescription = \_EwclaimPartMasterModel.PartDescription,

PartType = \_EwclaimPartMasterModel.PartType,

DealerCode = \_EwclaimPartMasterModel.DealerCode,

MakeCode = \_EwclaimPartMasterModel.MakeCode,

CreatedDate = \_EwclaimPartMasterModel.CreatedDate,

CreatedBy = \_EwclaimPartMasterModel.CreatedBy,

ModifiedDate = \_EwclaimPartMasterModel.ModifiedDate,

ModifiedBy = \_EwclaimPartMasterModel.ModifiedBy

};

\_context.EwclaimPartMasters.Update(\_EwclaimPartMaster);

await \_context.SaveChangesAsync();

}

public async Task UpdatePatchAsync(int partId, JsonPatchDocument \_EwclaimPartMaster) {

var existPartMaster = await \_context.EwclaimPartMasters.FindAsync(partId);

if (existPartMaster != null) {

\_EwclaimPartMaster.ApplyTo(existPartMaster);

await \_context.SaveChangesAsync();

}

}

public async Task DeleteAsync(int Partid) {

var existPartMaster = new EwclaimPartMaster() { PartId = Partid };

\_context.EwclaimPartMasters.Remove(existPartMaster);

await \_context.SaveChangesAsync();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace WarrantyAPITest.Models;

public partial class EwclaimPartMaster

{

public int PartId { get; set; }

[Required]

public string PartNumber { get; set; }

public string? PartDescription { get; set; }

public string? PartType { get; set; }

public string? DealerCode { get; set; }

public string? MakeCode { get; set; }

public DateTime? CreatedDate { get; set; }

public short? CreatedBy { get; set; }

public DateTime? ModifiedDate { get; set; }

public short? ModifiedBy { get; set; }

}

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.JsonPatch;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using WarrantyAPITest.Data;

using WarrantyAPITest.Interfaces;

using WarrantyAPITest.Models;

namespace WarrantyAPITest.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class EwclaimPartMasterController: ControllerBase

{

private readonly IEwclaimPartMasterRepository \_IEwclaimPartMasterRepository;

public EwclaimPartMasterController(IEwclaimPartMasterRepository objIEwclaimPartMasterRepository)

{

\_IEwclaimPartMasterRepository = objIEwclaimPartMasterRepository;

}

[HttpGet("")]

public async Task<ActionResult> GetAllParts() {

var parts=await \_IEwclaimPartMasterRepository.GetAllAsync();

return Ok(parts);

}

[HttpGet("{partnum}")]

public async Task<ActionResult> GetPartsByID([FromRoute] int partnum)

{

var ewclaimPartMasters = await \_IEwclaimPartMasterRepository.GetByIdAsync(partnum);

return Ok(ewclaimPartMasters);

}

//public async Task<ActionResult> GetPartsByID([FromRoute]string partnum)

//{

// var ewclaimPartMasters = await \_IEwclaimPartMasterRepository.GetByIdAsync(partnum);

// return Ok(ewclaimPartMasters);

//}

[HttpPost("")]

public async Task<ActionResult> AddNewPart([FromBody] EwclaimPartMaster \_part)

{

var partNo = await \_IEwclaimPartMasterRepository.AddAsync(\_part);

return CreatedAtAction(nameof(GetPartsByID),new { partnum = partNo ,Controller= "EwclaimPartMaster" },partNo );

// return CreatedAtAction(nameof(GetPartsByID), new { partnum = partNo, Controller = "EwclaimPartMaster" }, \_part);

}

[HttpPut("{partid}")]

public async Task<ActionResult> UpdateNewPart([FromRoute] int partid, [FromBody] EwclaimPartMaster \_part)

{

await \_IEwclaimPartMasterRepository.UpdateAsync(partid, \_part);

return Ok();

}

[HttpPatch("{partid}")]

public async Task<ActionResult> UpdateNewPartPatch([FromRoute] int partid, [FromBody] JsonPatchDocument \_part)

{

await \_IEwclaimPartMasterRepository.UpdatePatchAsync(partid, \_part);

return Ok();

}

[HttpDelete("{partid}")]

public async Task<ActionResult> DeletePart([FromRoute] int partid)

{

await \_IEwclaimPartMasterRepository.DeleteAsync(partid);

return Ok();

}

}

}