Parts of a .NET Core Web API Controller that retrieves information from a ResourceBook table using Entity Framework Core

**ChatGPT said:**

To create a .NET Core Web API controller that retrieves data from a ResourceBook table using **Entity Framework Core**, follow these steps:

**✅ 1. Create a ResourceBook Entity**

public class ResourceBook

{

public int Id { get; set; }

public string Make { get; set; }

public string Model { get; set; }

public int Year { get; set; }

}

**✅ 2. Create the DbContext**

using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<ResourceBook> ResourceBook{ get; set; }

}

**✅ 3. Add DbContext to Startup.cs or Program.cs**

If you're using .NET 6 or later (minimal hosting model):

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

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

*Make sure you have a connection string in appsettings.json:*

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=StudyResourceDb;Trusted\_Connection=True;"

}

**✅ 4. Create the ResourceBooks**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Threading.Tasks;

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

[ApiController]

public class ResourceBooks : ControllerBase

{

private readonly ApplicationDbContext \_context;

public ResourceBooks(ApplicationDbContext context)

{

\_context = context;

}

// GET: api/ResourceBooks

[HttpGet]

public async Task<ActionResult<IEnumerable<ResourceBook>>> GetResourceBooks()

{

return await \_context.ResourceBooks.ToListAsync();

}

// GET: api/ResourceBooks/5

[HttpGet("{id}")]

public async Task<ActionResult<ResourceBook>> GetResourceBook(int id)

{

var ResourceBook = await \_context.ResourceBooks.FindAsync(id);

if (ResourceBook == null)

{

return NotFound();

}

return ResourceBook;

}

}

**✅ 5. Run Migrations & Update DB (optional)**

dotnet ef migrations add InitialCreate

dotnet ef database update

**🧪 Test the API**

Once the app is running, you can test:

* GET /api/ResourceBook→ Get all ResourceBooks
* GET /api/ResourceBooks/{id} → Get a single ResourceBook by ID

Use **Postman**, **curl**, or Swagger UI.

Let me know if you'd like to include POST, PUT, or DELETE endpoints too.

**You said:**

what code is needed for a .NET Core Web API Controller to retrieve information from a ResourceBook table using Entity Framework Core. Please provide all code including supporting code.

**ChatGPT said:**

Here's a complete example of how to create a **.NET Core Web API** project that retrieves information from a ResourceBook table using **Entity Framework Core**.

**✅ Tech Stack**

* .NET 6+ (can be adapted to 3.1/5.0)
* Entity Framework Core
* SQL Server (can be swapped with another provider)

**📁 Project Structure**

WebApplication1/

│

├── Controllers/

│ └── ResourceBooks.cs

│

├── Models/

│ └── ResourceBook.cs

│

├── Data/

│ └── ApplicationDbContext.cs

│

├── Program.cs

├── appsettings.json

**1. ✅ Model: ResourceBook.cs**

📄 Models/ResourceBook.cs

namespace WebApplication1.Models

{

public class ResourceBook

{

public int Id { get; set; }

public string Make { get; set; } = string.Empty;

public string Model { get; set; } = string.Empty;

public int Year { get; set; }

}

}

**2. ✅ DbContext: ApplicationDbContext.cs**

📄 Data/ApplicationDbContext.cs

using Microsoft.EntityFrameworkCore;

using WebApplication1.Models;

namespace WebApplication1.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options) { }

public DbSet<Car> ResourceBook { get; set; }

}

}

**3. ✅ Controller: ResourceBooks.cs**

📄 Controllers/ResourceBooks.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using WebApplication1.Data;

using WebApplication1.Models;

namespace WebApplication1.Controllers

{

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

[ApiController]

public class ResourceBooks : ControllerBase

{

private readonly ApplicationDbContext \_context;

public ResourceBooks(ApplicationDbContext context)

{

\_context = context;

}

// GET: api/ResourceBooks

[HttpGet]

public async Task<ActionResult<IEnumerable<ResourceBook>>> GetResourceBooks()

{

return await \_context.ResourceBooks.ToListAsync();

}

// GET: api/ResourceBooks/5

[HttpGet("{id}")]

public async Task<ActionResult<ResourceBook>> GetResourceBook(int id)

{

var ResourceBook = await \_context.ResourceBooks.FindAsync(id);

if (ResourceBook == null)

{

return NotFound();

}

return ResourceBook;

}

}

}

**4. ✅ Configure the App: Program.cs**

📄 Program.cs (for .NET 6+)

using WebApplication1.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

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

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5. ✅ Configuration: appsettings.json**

📄 appsettings.json

{

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=StudyResourceDb;Trusted\_Connection=True;TrustServerCertificate=True"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**6. ✅ Initialize the Database (Migration)**

Open terminal in your project root and run:

dotnet ef migrations add InitialCreate

dotnet ef database update

Make sure you have installed EF Core tools:

dotnet tool install --global dotnet-ef

**7. 🧪 Test API Endpoints**

Once running, test the endpoints:

* GET /api/ResourceBook → Returns all ResourceBooks
* GET /api/ResourceBooks/1 → Returns ResourceBook with ID 1

Swagger UI: navigate to https://localhost:{port}/swagger

**🧱 Optional: Add Sample Seed Data?**

Let me know if you want to include seed data or add full CRUD (create, update, delete) endpoints too.

Top of Form

Bottom of Form

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

**We use cookies**

Some cookies are essential for this site to function and cannot be turned off. Others are set to help us understand how our service performs and is used, and to support our marketing efforts. Learn more in our [Cookie Policy](https://openai.com/policies/cookie-policy/). For more info or to customize your preferences, tap ‘Manage Cookies’.