**Step 1: Setup the Database**

1. Open a terminal and connect to SQL Server using sqlcmd:

sqlcmd -U sa -P examlyMssql@123

1. Create the database:

CREATE DATABASE appdb;

GO

USE appdb;

GO

1. Create the Books table:

CREATE TABLE Books (

Id INT PRIMARY KEY,

Title VARCHAR(100) NOT NULL,

Author VARCHAR(100) NOT NULL,

Price DECIMAL(10,2) NOT NULL,

Quantity INT NOT NULL

);

GO

1. Insert sample data:

INSERT INTO Books (Id, Title, Author, Price, Quantity) VALUES

(1, 'C# Fundamentals', 'John Doe', 499.99, 10),

(2, 'ASP.NET Core Guide', 'Jane Smith', 599.50, 5),

(3, 'EF Core in Action', 'Mark Brown', 699.00, 8);

GO

**Step 2: Setup the .NET Project**

1. Create a new Web API project:

dotnet new webapi -n BookStoreApp

cd BookStoreApp

1. Install EF Core tools (if not already):

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0.6

dotnet dotnet-ef

**Step 3: Generate Models using Database-First Approach**

Run the **scaffold command** to generate the Book model and appdbContext:

dotnet dotnet-ef dbcontext scaffold "User ID=sa;password=examlyMssql@123;Server=localhost;Database=appdb;Trusted\_Connection=False;Persist Security Info=False;Encrypt=False" Microsoft.EntityFrameworkCore.SqlServer -o Models

This will generate:

* Book.cs → model class
* appdbContext.cs → EF Core DbContext class

**Step 4: Verify the Generated Model**

Models/Book.cs should look like:

public partial class Book

{

public int Id { get; set; }

public string Title { get; set; } = null!;

public string Author { get; set; } = null!;

public decimal Price { get; set; }

public int Quantity { get; set; }

}

Models/appdbContext.cs should look like:

public partial class appdbContext : DbContext

{

public appdbContext() { }

public appdbContext(DbContextOptions<appdbContext> options) : base(options) { }

public virtual DbSet<Book> Books { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("User ID=sa;password=examlyMssql@123;Server=localhost;Database=appdb;Trusted\_Connection=False;Persist Security Info=False;Encrypt=False");

}

}

}

**Step 5: Create a Controller**

Add Controllers/BooksController.cs:

using Microsoft.AspNetCore.Mvc;

using BookStoreApp.Models;

using Microsoft.EntityFrameworkCore;

namespace BookStoreApp.Controllers

{

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

[ApiController]

public class BooksController : ControllerBase

{

private readonly appdbContext \_context;

public BooksController(appdbContext context)

{

\_context = context;

}

// GET: api/Books

[HttpGet]

public async Task<ActionResult<IEnumerable<Book>>> GetBooks()

{

return await \_context.Books.ToListAsync();

}

// GET: api/Books/1

[HttpGet("{id}")]

public async Task<ActionResult<Book>> GetBook(int id)

{

var book = await \_context.Books.FindAsync(id);

if (book == null)

return NotFound();

return book;

}

// POST: api/Books

[HttpPost]

public async Task<ActionResult<Book>> PostBook(Book book)

{

\_context.Books.Add(book);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetBook), new { id = book.Id }, book);

}

// PUT: api/Books/1

[HttpPut("{id}")]

public async Task<IActionResult> PutBook(int id, Book book)

{

if (id != book.Id)

return BadRequest();

\_context.Entry(book).State = EntityState.Modified;

await \_context.SaveChangesAsync();

return NoContent();

}

// DELETE: api/Books/1

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBook(int id)

{

var book = await \_context.Books.FindAsync(id);

if (book == null)

return NotFound();

\_context.Books.Remove(book);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

**Step 6: Run the Project**

1. Restore packages:

dotnet restore

1. Build the project:

dotnet build

1. Run the project:

dotnet run

By default, it will run on https://localhost:5001 or http://localhost:5000.  
Use Postman or Swagger to test API endpoints.