**Step 1: Create Database and Table**

1. Open terminal and connect to SQL Server:

sqlcmd -U sa -P examlyMssql@123

1. Create the database:

CREATE DATABASE appdb;

GO

USE appdb;

GO

1. Create **Library** table:

CREATE TABLE Library (

LibraryId INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Address VARCHAR(200) NOT NULL,

City VARCHAR(50) NOT NULL,

State VARCHAR(50) NOT NULL,

ZipCode VARCHAR(20) NOT NULL,

PhoneNumber VARCHAR(20) NOT NULL

);

GO

1. Insert sample data manually:

INSERT INTO Library (LibraryId, Name, Address, City, State, ZipCode, PhoneNumber) VALUES

(1, 'Central Library', '123 Main St', 'New York', 'NY', '10001', '123-456-7890'),

(2, 'Westside Library', '456 Elm St', 'Los Angeles', 'CA', '90001', '987-654-3210');

GO

**Step 2: Setup .NET Project**

1. Create a new Web API project:

dotnet new webapi -n LibraryApp

cd LibraryApp

1. Install EF Core tools:

dotnet new tool-manifest

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

dotnet dotnet-ef

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

Run the **scaffold command**:

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:

* Library.cs → model class
* appdbContext.cs → DbContext class

**Step 4: Verify Model Class**

Models/Library.cs should look like:

public partial class Library

{

public int LibraryId { get; set; }

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

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

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

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

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

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

}

**Step 5: Verify DbContext**

Models/appdbContext.cs should look like:

public partial class appdbContext : DbContext

{

public appdbContext() { }

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

public virtual DbSet<Library> Libraries { 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 6: Create Controller**

Controllers/LibraryController.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using LibraryApp.Models;

namespace LibraryApp.Controllers

{

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

[ApiController]

public class LibraryController : ControllerBase

{

private readonly appdbContext \_context;

public LibraryController(appdbContext context)

{

\_context = context;

}

// GET: api/Library

[HttpGet]

public async Task<ActionResult<IEnumerable<Library>>> GetLibraries()

{

return await \_context.Libraries.ToListAsync();

}

// GET: api/Library/1

[HttpGet("{id}")]

public async Task<ActionResult<Library>> GetLibrary(int id)

{

var library = await \_context.Libraries.FindAsync(id);

if (library == null) return NotFound();

return library;

}

// POST: api/Library

[HttpPost]

public async Task<ActionResult<Library>> PostLibrary(Library library)

{

\_context.Libraries.Add(library);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetLibrary), new { id = library.LibraryId }, library);

}

// PUT: api/Library/1

[HttpPut("{id}")]

public async Task<IActionResult> PutLibrary(int id, Library library)

{

if (id != library.LibraryId) return BadRequest();

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

await \_context.SaveChangesAsync();

return NoContent();

}

// DELETE: api/Library/1

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteLibrary(int id)

{

var library = await \_context.Libraries.FindAsync(id);

if (library == null) return NotFound();

\_context.Libraries.Remove(library);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

**Step 7: Run the Project**

1. Restore packages:

dotnet restore

1. Build the project:

dotnet build

1. Run the project:

dotnet run

* Default ports: https://localhost:5001 or http://localhost:5000
* Test API endpoints via Swagger (/swagger) or Postman.

**Step 8: Test API Endpoints**

| **HTTP Method** | **Endpoint** | **Action** |
| --- | --- | --- |
| GET | /api/Library | Get all libraries |
| GET | /api/Library/1 | Get library by Id |
| POST | /api/Library | Add new library |
| PUT | /api/Library/1 | Update library |
| DELETE | /api/Library/1 | Delete library |