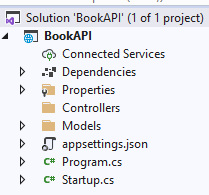
* Open Visual Studio 2019
* Click to File> New > Project from the Menu  
  Select the ASP.NET Core Web API template and click Next.
* Name the project BookApi and Create a new ASP.NET Core Web Application dialog, Select the API template and click Create.



**Add NuGet packages**  
• **Tools menu**, select **NuGet Package Manager** > **Manage NuGet Packages for Solution**.

• **Microsoft.EntityFrameworkCore.InMemory**

This class is created by deriving from **Microsoft.EntityFrameworkCore.DbContext** class.

using Microsoft.EntityFrameworkCore;

namespace BookApi.Models

{

public class BooksContext : DbContext

{

public BooksContext(DbContextOptions<BooksContext> options)

: base(options)

{

}

public DbSet<Book> Books { get; set; }

}

}

public class Book

{

public long Id { get; set; }

public string Name { get; set; }

}

Register the book api database context to Startup.cs file with the following code:

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<BooksContext>(opt => opt.UseInMemoryDatabase("Books"));

services.AddControllers();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "BookAPI", Version = "v1" });

});

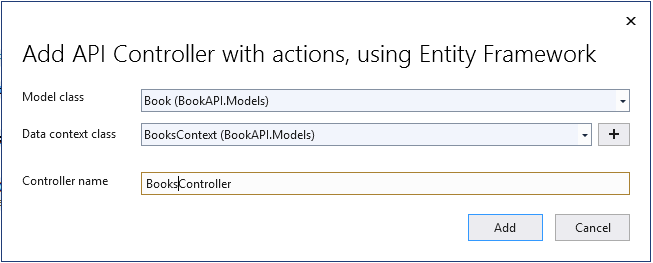
}

**Scaffold a controller**

Right-click the Controllers folder and select API **Controller with actions, using Entity Framework**, and then select Add.

In the **Add API Controller with actions, using Entity Framework** dialog:

Select **Book** in the Model class.  
Select **BooksContext** in the Data context class



using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using BookAPI.Models;

namespace BookAPI.Controllers

{

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

[ApiController]

public class BooksController : ControllerBase

{

private readonly BooksContext \_context;

public BooksController(BooksContext context)

{

\_context = context;

}

// GET: api/Books

[HttpGet]

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

{

return await \_context.Books.ToListAsync();

}

// GET: api/Books/5

[HttpGet("{id}")]

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

{

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

if (book == null)

{

return NotFound();

}

return book;

}

// PUT: api/Books/5

[HttpPut("{id}")]

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

{

if (id != book.Id)

{

return BadRequest();

}

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

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!BookExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Books

[HttpPost]

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

{

\_context.Books.Add(book);

await \_context.SaveChangesAsync();

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

}

// DELETE: api/Books/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBook(long id)

{

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

if (book == null)

{

return NotFound();

}

\_context.Books.Remove(book);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool BookExists(long id)

{

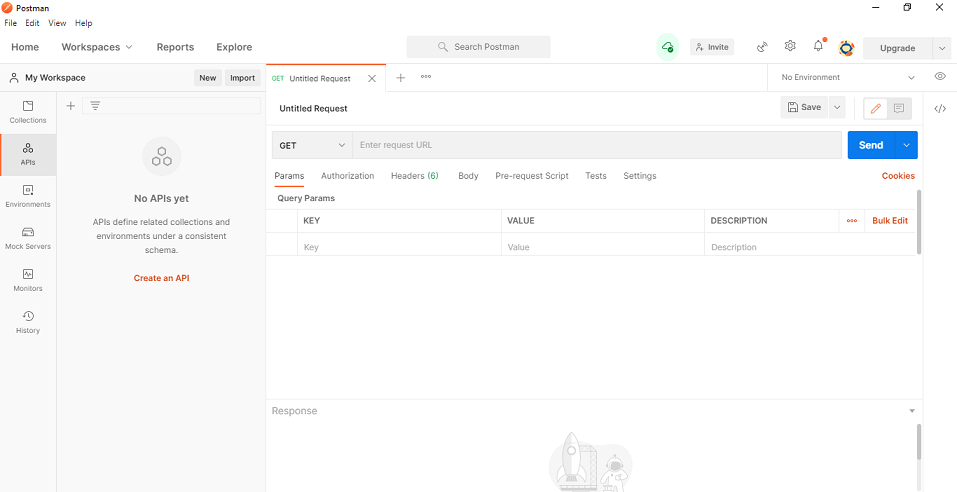
return \_context.Books.Any(e => e.Id == id);

}

}

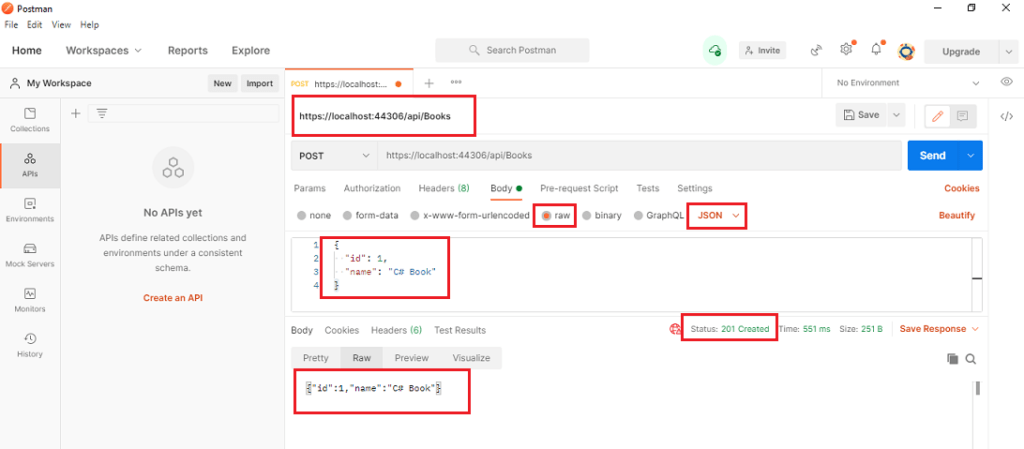
}

Open Postman and It will look like something similar to the following screenshot.



**Testing POST Web API with Postman**

Choose the HTTP verb as **POST** and set the **URL** then set the Content-Type as **application/json**. To do this click on the Header tab and provide the key value as shown in the below image



**Testing GET Web API with Postman**

Choose the HTTP verb as **GET** and set the **URL**  and click on send button as shown in the below image.

