C# Fundamentals

10. **Building a Mini Microservice with ASP.NET CoreObjective:Requirements:**
    - **Create a small RESTful API that manages a resource (e.g., Products, Orders, or Books) using ASP.NET Core.**
    - **Project Setup:**
        - **Set up a new ASP.NET Core Web API project.**
        - **Configure routing and controllers.**
    - **Dependency Injection:**
        - **Implement a service layer and register services using ASP.NET Core's dependency injection.**
    - **Data Access:**
        - **Use Entity Framework Core with an in-memory database (or SQLite) to perform CRUD operations.**
    - **Asynchronous Operations:**
        - **Use async/await in your controller actions to handle database operations.**
    - **Error Handling and Logging:**
        - **Implement middleware or filters for global exception handling.**
        - **Integrate basic logging to record actions and errors.**
    - **Testing and Documentation:**
        - **Write unit tests for your controllers and services.**
        - **Document the API endpoints (using tools like Swagger).**
    - **Advanced Considerations (Optional):**
        - **Incorporate design patterns such as Repository and Unit of Work.**
        - **Implement custom middleware for request/response logging or authentication.**

Program code:

Book.cs

```
//          Represents the data
namespace BookStoreApi.Models
{
   public class Book
   {
      public int Id { get; set; } // Primary key
      public string Title { get; set; }
      public string Author { get; set; }
      public decimal Price { get; set; }
   }
}
```

AppDbContext.cs

```
//          Connects to the database
using Microsoft.EntityFrameworkCore;
using BookStoreApi.Models; // includes Book class.
namespace BookStoreApi.Data
{
   public class AppDbContext : DbContext  // inherits from DbContext
   {
      // DbContextOptions<AppDbContext> - provides DB configuration like in-memory or SQL
      // It receives an options object (with connection info and config) and passes it to the base
DbContext class.
      public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

      // DbSet<Book> means EF Core should create and manage a table for Book entities. Books is the
name of the table.
      // Set<Book>() returns the internal set EF uses for queries and changes.
      // When someone accesses the Books property, return the result of Set<Book>()
      public DbSet<Book> Books => Set<Book>();
```

```
    }
}
```

IBookRepository.cs

```
// Defines methods for CRUD
using BookStoreApi.Models;
namespace BookStoreApi.Repositories
{
    public interface IBookRepository
    {
        // Task<> indicates this method is asynchronous
        // IEnumerable<Book> means it's a collection of Book objects.
        Task<IEnumerable<Book>> GetAllAsync();

        //  `?` means the return value can be `null` (nullable)
        // Returns a single book by its `id`, or `null` if not found.
        Task<Book?> GetByIdAsync(int id);

        Task<Book> AddAsync(Book book);

        // Returns the updated book, or `null` if not found.
        Task<Book?> UpdateAsync(int id, Book updatedBook);

        Task<bool> DeleteAsync(int id);
    }
}
```

BookRepository.cs

```
// Implements methods for CRud
```

```
using Microsoft.EntityFrameworkCore;
using BookStoreApi.Data;
using BookStoreApi.Models;

namespace BookStoreApi.Repositories
{
    public class BookRepository : IBookRepository
    {
        // `readonly` means it can only be set once in the constructor.
        private readonly AppDbContext _context;

        public BookRepository(AppDbContext context)
        {
            _context = context;
        }

        // ToListAsync() fetches the data as a list.
        public async Task<IEnumerable<Book>> GetAllAsync() =>
            await _context.Books.ToListAsync();

        // FindAsync looks it up using the primary key
        public async Task<Book?> GetByIdAsync(int id) =>
            await _context.Books.FindAsync(id);

        // SaveChangesAsync() writes it to the database.
        // Add() stages the book for insertion.
        public async Task<Book> AddAsync(Book book)
        {
            _context.Books.Add(book);
            await _context.SaveChangesAsync();
```

```
        return book;
    }

    public async Task<Book?> UpdateAsync(int id, Book updatedBook)
    {
        var existing = await _context.Books.FindAsync(id);
        if (existing == null) return null;

        existing.Title = updatedBook.Title;
        existing.Author = updatedBook.Author;
        existing.Price = updatedBook.Price;

        await _context.SaveChangesAsync();
        return existing;
    }

    public async Task<bool> DeleteAsync(int id)
    {
        var book = await _context.Books.FindAsync(id);
        if (book == null) return false;

        _context.Books.Remove(book);
        await _context.SaveChangesAsync();
        return true;
    }
  }
}
```

BooksController.cs

```
//          Exposes HTTP endpoints
```

```
using Microsoft.AspNetCore.Mvc;

using BookStoreApi.Models;

using BookStoreApi.Repositories;


namespace BookStoreApi.Controllers

{

    [ApiController]

    // [Route("api/[controller]")] means this controller handles requests like:GET /api/books, POST
/api/books, etc.

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

    public class BooksController : ControllerBase

    {

        private readonly IBookRepository _repo;


        // dependency injection - ASP.NET Core will automatically provide an instance of
`IBookRepository` when it creates the controller.

        public BooksController(IBookRepository repo)

        {

            _repo = repo;

        }


        // ActionResult<T> - returning data and HTTP responses and acts as a return type for controller
actions.

        // Calls the repository to get all books.

        [HttpGet]

        public async Task<ActionResult<IEnumerable<Book>>> GetAll() =>

            Ok(await _repo.GetAllAsync());


        // If found, returns 200 OK. If not, returns 404 Not Found.

        // GET /api/books/{id}

        [HttpGet("{id}")]

        public async Task<ActionResult<Book>> GetById(int id)
```

```
{
    var book = await _repo.GetByIdAsync(id);
    if (book == null) return NotFound();
    return Ok(book);
}

// Returns 201 Created with the location of the new book using CreatedAtAction().
// POST /api/books
[HttpPost]
public async Task<ActionResult<Book>> Create(Book book)
{
    var created = await _repo.AddAsync(book);
    return CreatedAtAction(nameof(GetById), new { id = created.Id }, created);
}

// If not found, 404 Not Found. If successful, 200 OK with the updated book.
// PUT /api/books/{id}
[HttpPut("{id}")]
public async Task<ActionResult<Book>> Update(int id, Book book)
{
    var updated = await _repo.UpdateAsync(id, book);
    if (updated == null) return NotFound();
    return Ok(updated);
}

// If not found, 404 Not Found. If deleted, 204 No Content (success but no body).
// DELETE /api/books/{id}
[HttpDelete("{id}")]
public async Task<IActionResult> Delete(int id)
{
    var deleted = await _repo.DeleteAsync(id);
```

```
            if (!deleted) return NotFound();
            return NoContent();
        }
    }
}
```

// Swagger is used bcoz it shows tehse as a UI where you can test each route interactively

Program.cs

// Sets up and starts the web app

```
using BookStoreApi.Data;
using BookStoreApi.Repositories;
using Microsoft.EntityFrameworkCore;
```

// This sets up the web app builder, which is used to configure: Services (like database, controllers), Middleware (like Swagger or routing)

```
var builder = WebApplication.CreateBuilder(args);
```

// Add services

// registering the AppDbContext and telling it to use an in-memory database named BooksDb. hence data will not persist after the app is closed

```
builder.Services.AddDbContext<AppDbContext>(opt => opt.UseInMemoryDatabase("BooksDb"));
```

// AddScoped means a new instance will be created for each HTTP request

// Whenever someone asks for IBookRepository, give them an instance of BookRepository

```
builder.Services.AddScoped<IBookRepository, BookRepository>();
```

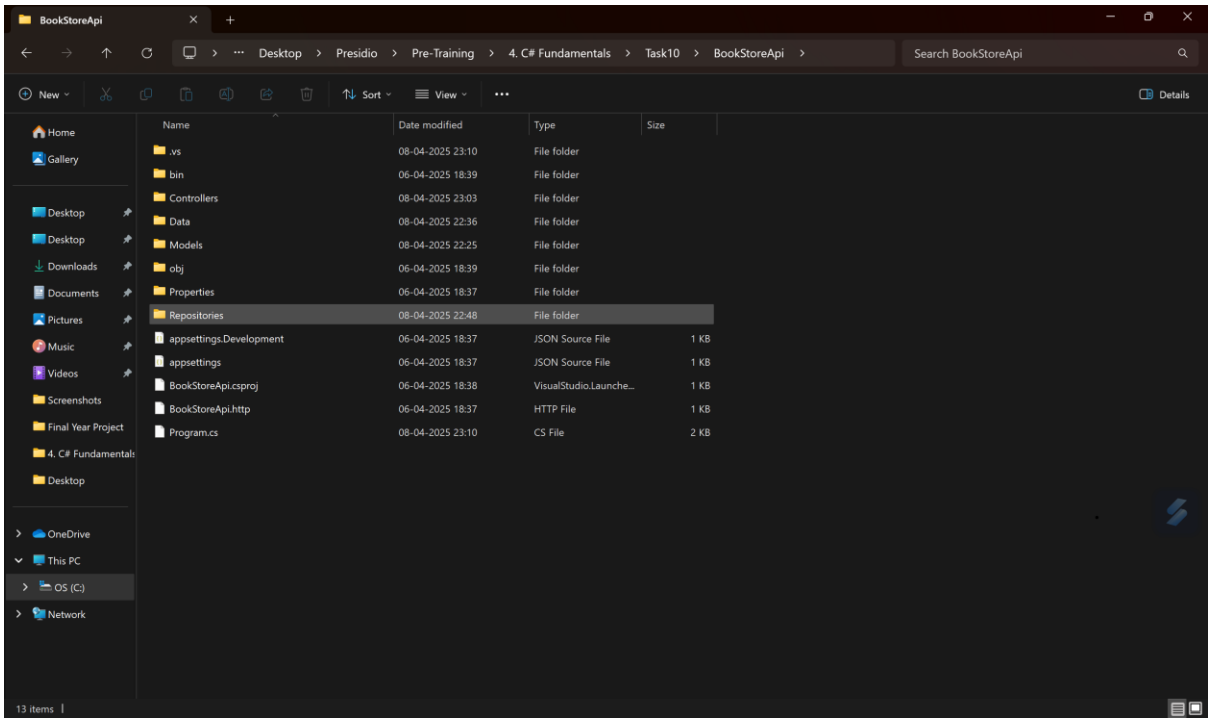
// adds support for API BooksController

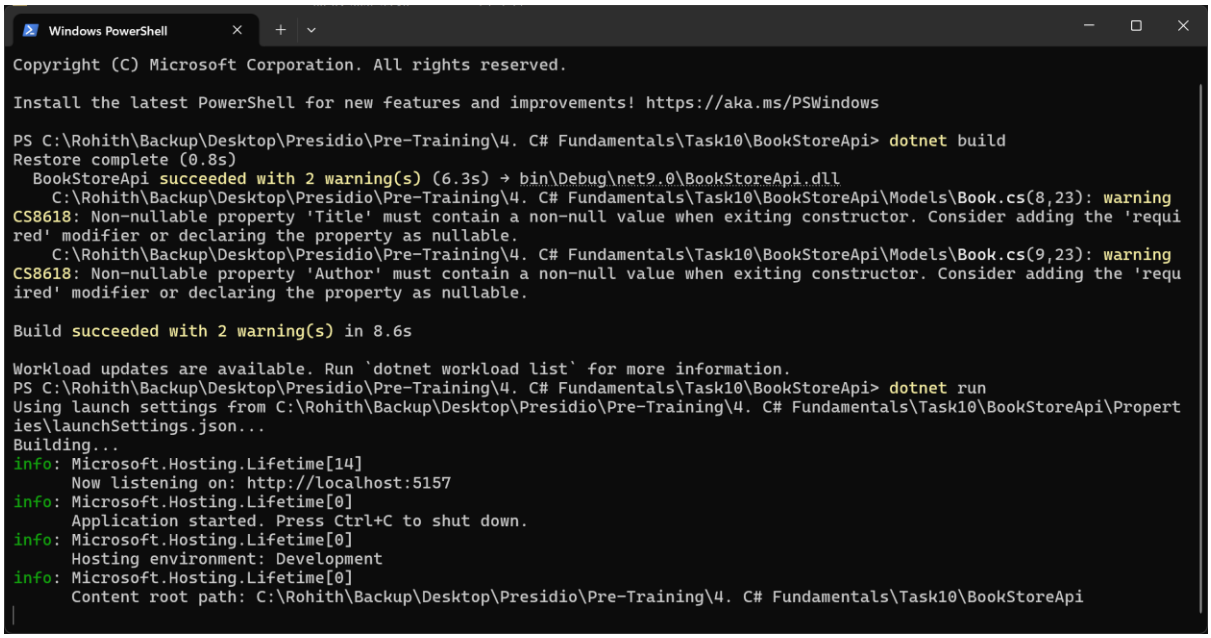
```
builder.Services.AddControllers();
```

```
// AddEndpointsApiExplorer() helps Swagger discover endpoints.
builder.Services.AddEndpointsApiExplorer();

// AddSwaggerGen() generates Swagger UI to test APIs interactively.
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Middleware
if (app.Environment.IsDevelopment())
{
    // Enables Swagger middleware
    app.UseSwagger();

    // Adds a nice web-based Swagger UI at /swagger
    app.UseSwaggerUI();
}

// Adds middleware to handle authorization.
app.UseAuthorization();

// Use all controllers with routing like api/books, api/books/{id} etc
app.MapControllers();

// Starts the web application — begins listening for HTTP requests.
app.Run();
```

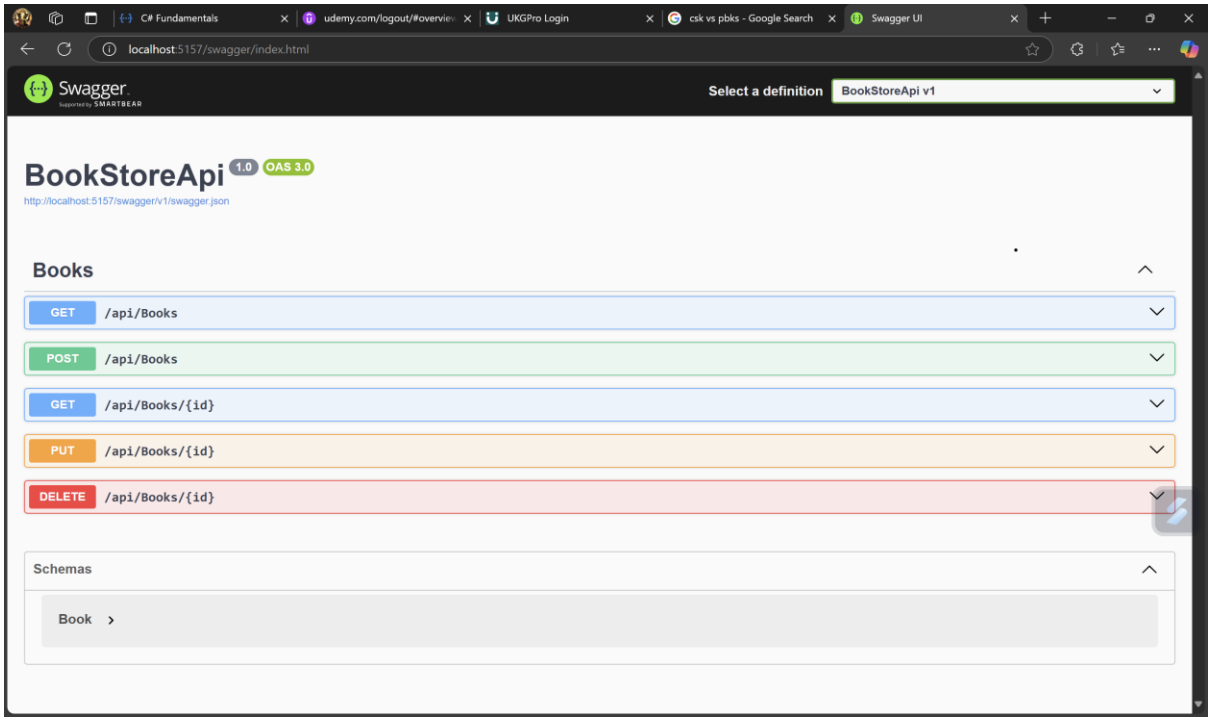
Output:

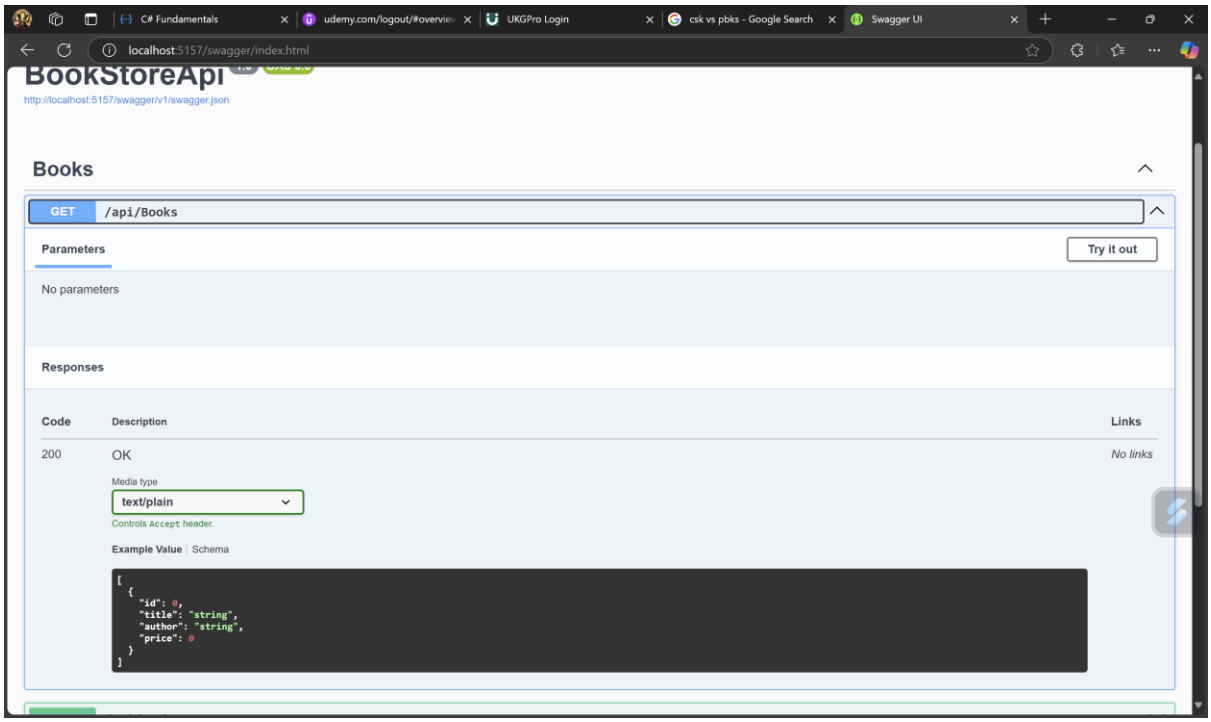
Go into directory where BookStoreApi.csproj is present





GO to http://localhost:5157/swagger/





Click Try it out and then execute

C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

GET /api/Books

Parameters

Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5157/api/Books' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5157/api/Books
```

Server response

Code	Details
200	Response body

```
[]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:03:44 GMT
server: Kestrel
transfer-encoding: chunked
```

C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

```
  'http://localhost:5157/api/Books' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5157/api/Books
```

Server response

Code	Details
200	Response body

```
[]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:03:44 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

Media type

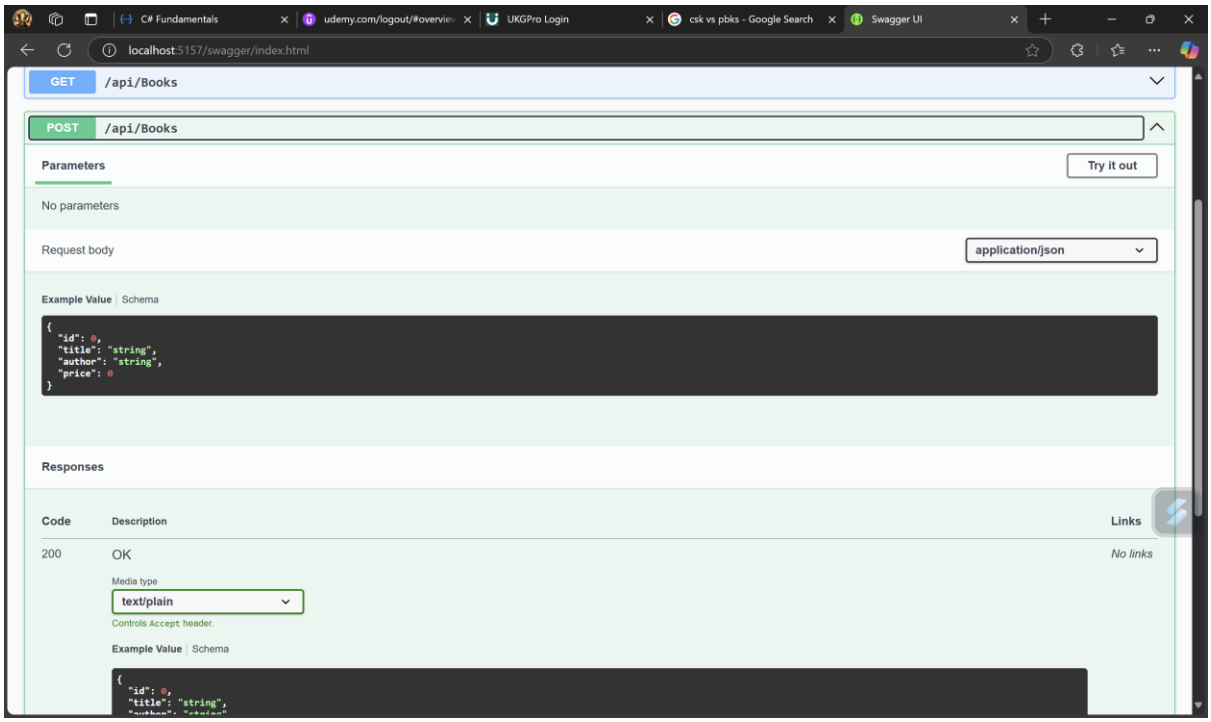
text/plain

Controls Accept header.

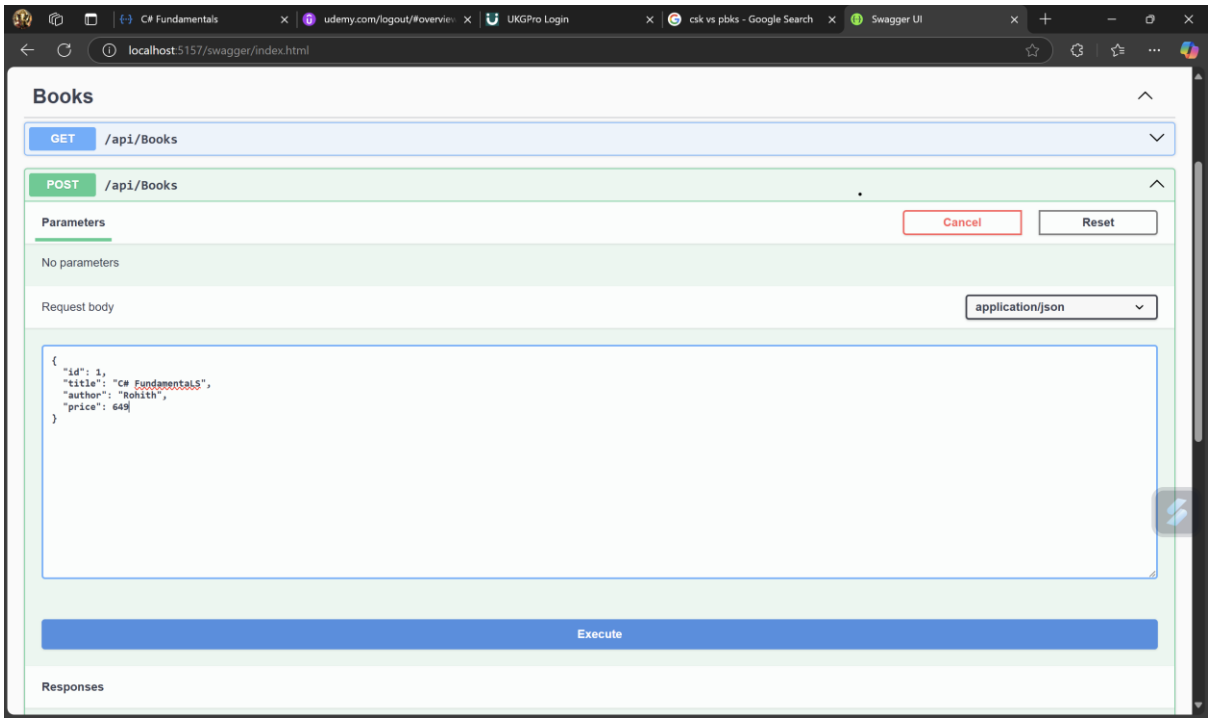
Example Value | Schema

```
[
  {
    "id": 0,
    "title": "string",
    "author": "string",
    "price": 0
  }
]
```

Now lets post some data



Click on try it out and execute



C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5157/api/Books' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 1,
  "title": "C# FundamentaLS",
  "author": "Rohith",
  "price": 649
}'
```

Request URL

```
http://localhost:5157/api/Books
```

Server response

Code	Details
201 *Undocumented*	**Response body**

```
{
  "id": 1,
  "title": "C# FundamentaLS",
  "author": "Rohith",
  "price": 649
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:06:40 GMT
location: http://localhost:5157/api/Books/1
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

Request URL

```
http://localhost:5157/api/Books
```

Server response

Code	Details
201 *Undocumented*	**Response body**

```
{
  "id": 1,
  "title": "C# FundamentaLS",
  "author": "Rohith",
  "price": 649
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:06:40 GMT
location: http://localhost:5157/api/Books/1
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

text/plain

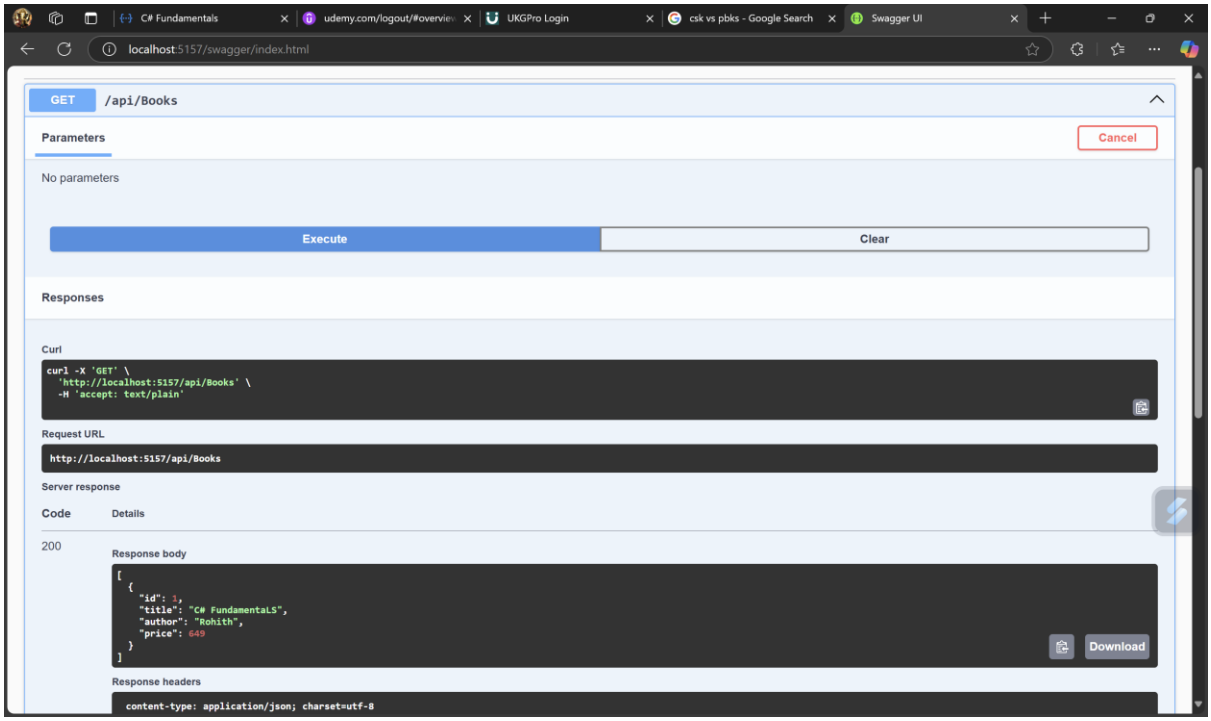
Controls Accept header.

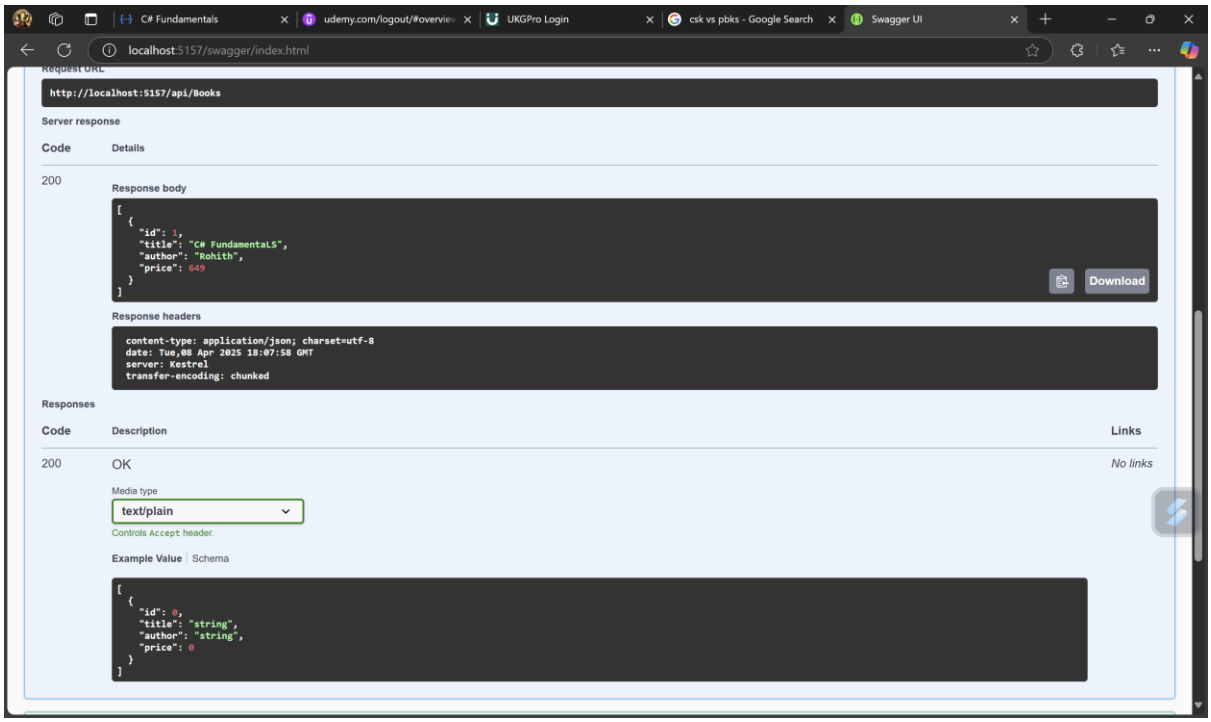
Example Value | Schema

```
{
  "id": 0,
  "title": "string",
  "author": "string",
  "price": 0
}
```

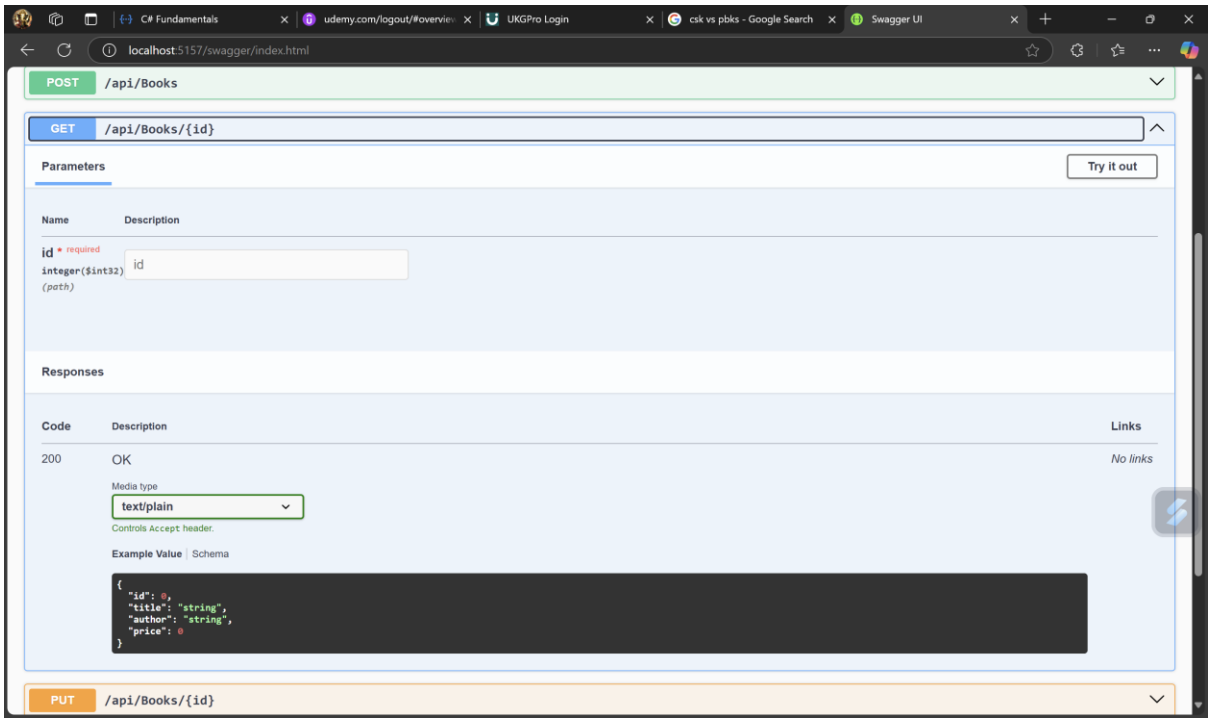
Now lets check the GET /api/Books

Click on Execute

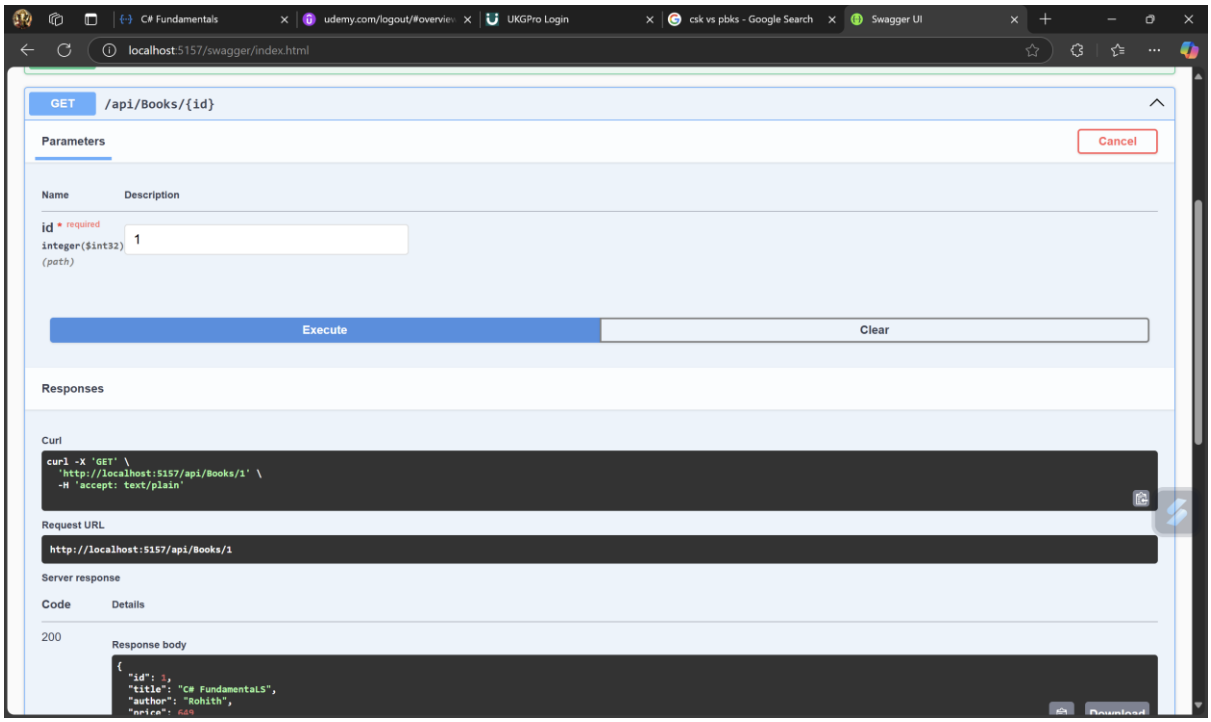


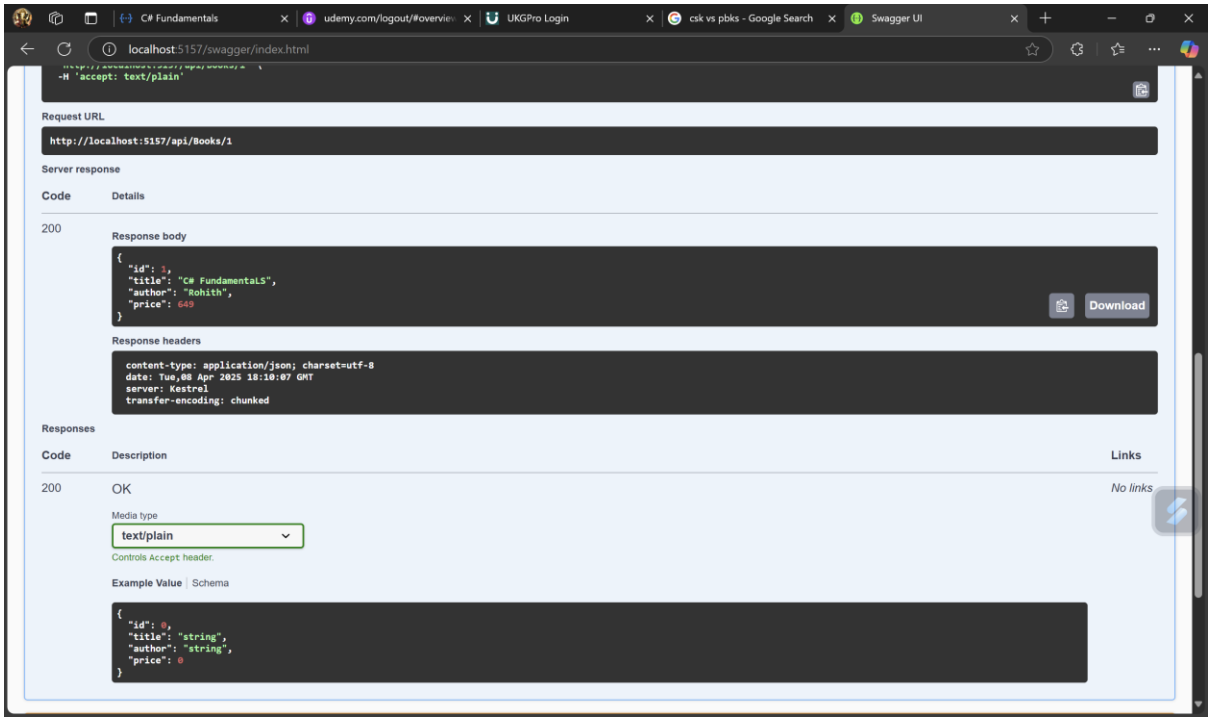


Now lets search for specific id: GET /api/Books/{id}

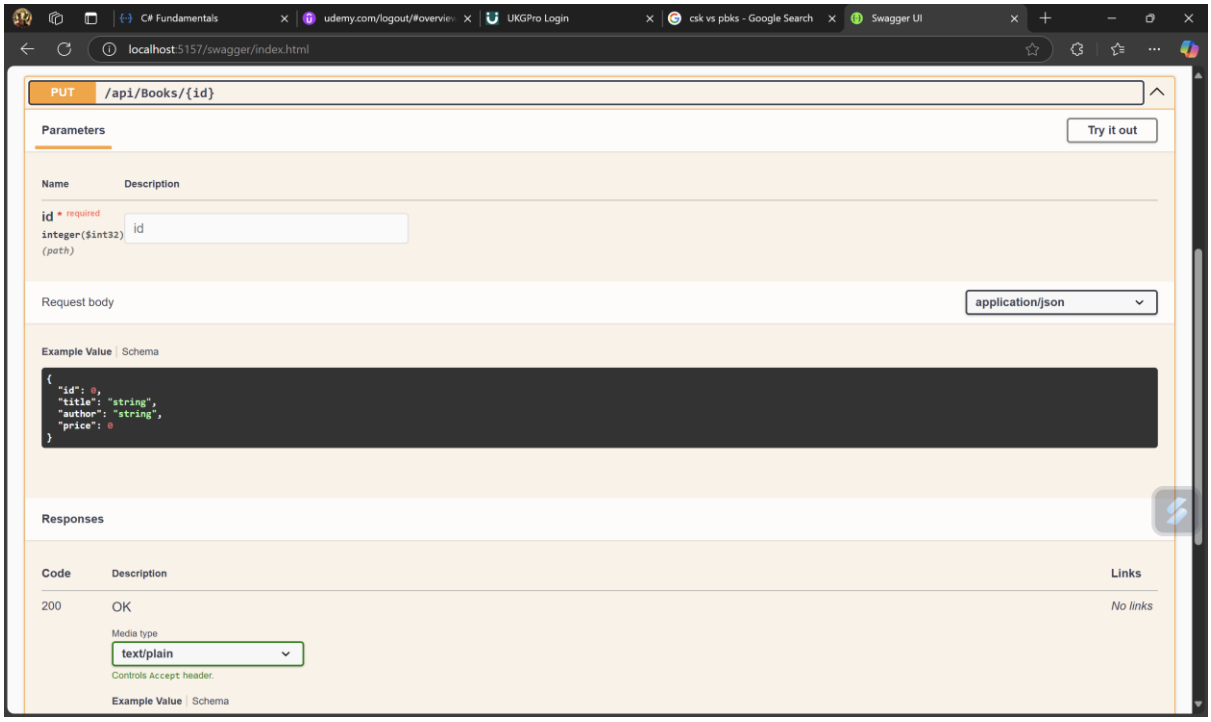


Click on Try it out, Enter id and Execute



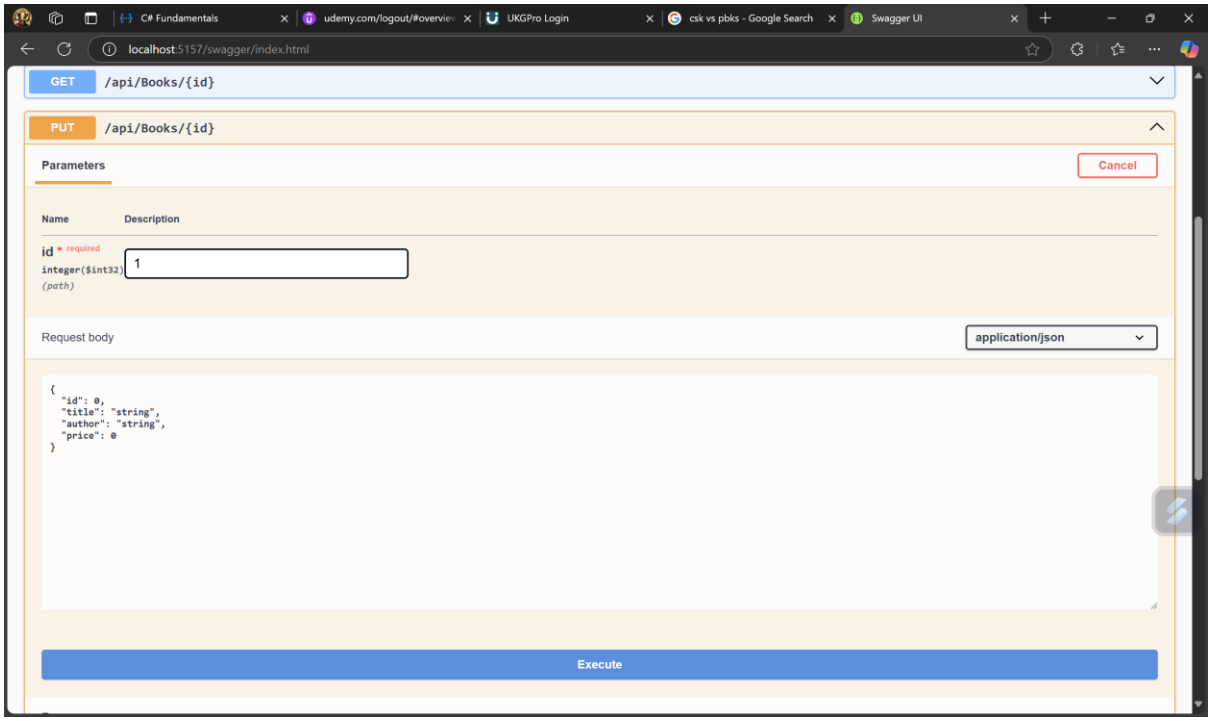


Now lets modify the id 1 using PUT /api/Books/{id}

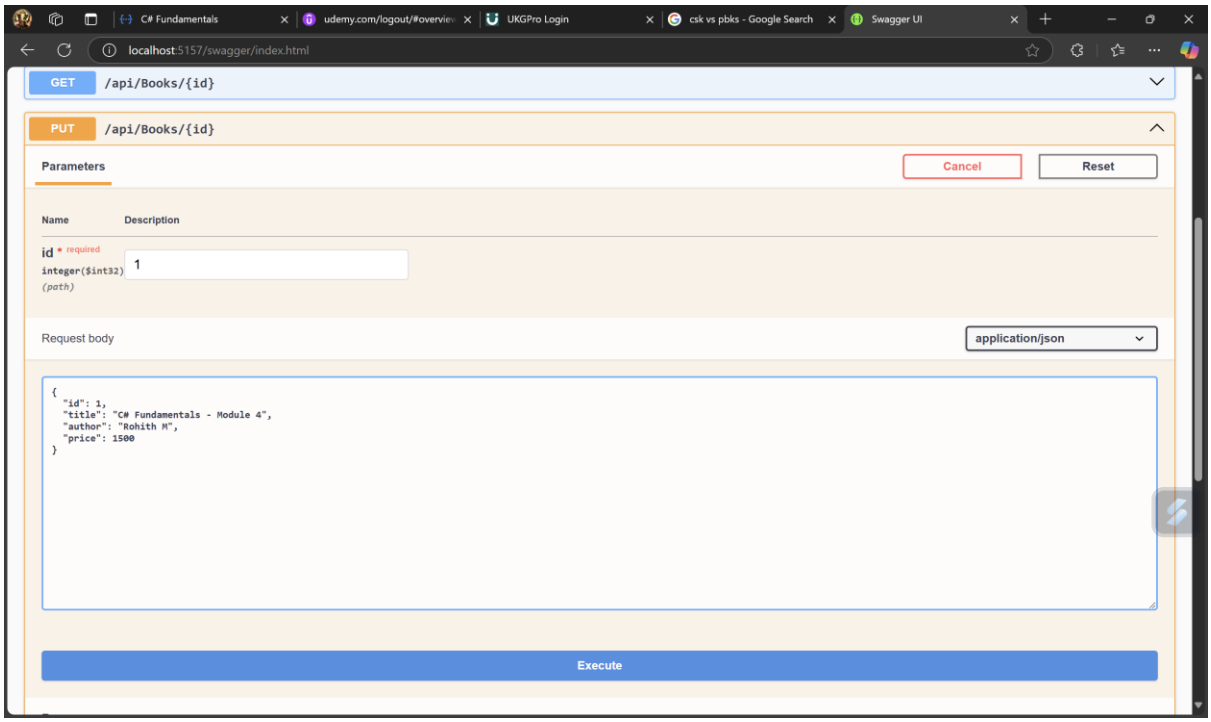


Click on Try it out

Enter id and update the details of Request Body



Now click on Execute



C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

Execute | Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:5157/api/Books/1' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 1,
  "title": "C# Fundamentals - Module 4",
  "author": "Rohith M",
  "price": 1500
}'
```

Request URL

```
http://localhost:5157/api/Books/1
```

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "title": "C# Fundamentals - Module 4",
  "author": "Rohith M",
  "price": 1500
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:14:34 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links

C# Fundamentals | udemy.com/logout/#overvie | UKGPro Login | csk vs pbks - Google Search | Swagger UI

localhost:5157/swagger/index.html

http://localhost:5157/api/Books/1

Server response

Code	Details
200	Response body

{
  "id": 1,
  "title": "C# Fundamentals - Module 4",
  "author": "Rohith M",
  "price": 1500
}

Download

Response headers

content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:14:34 GMT
server: Kestrel
transfer-encoding: chunked

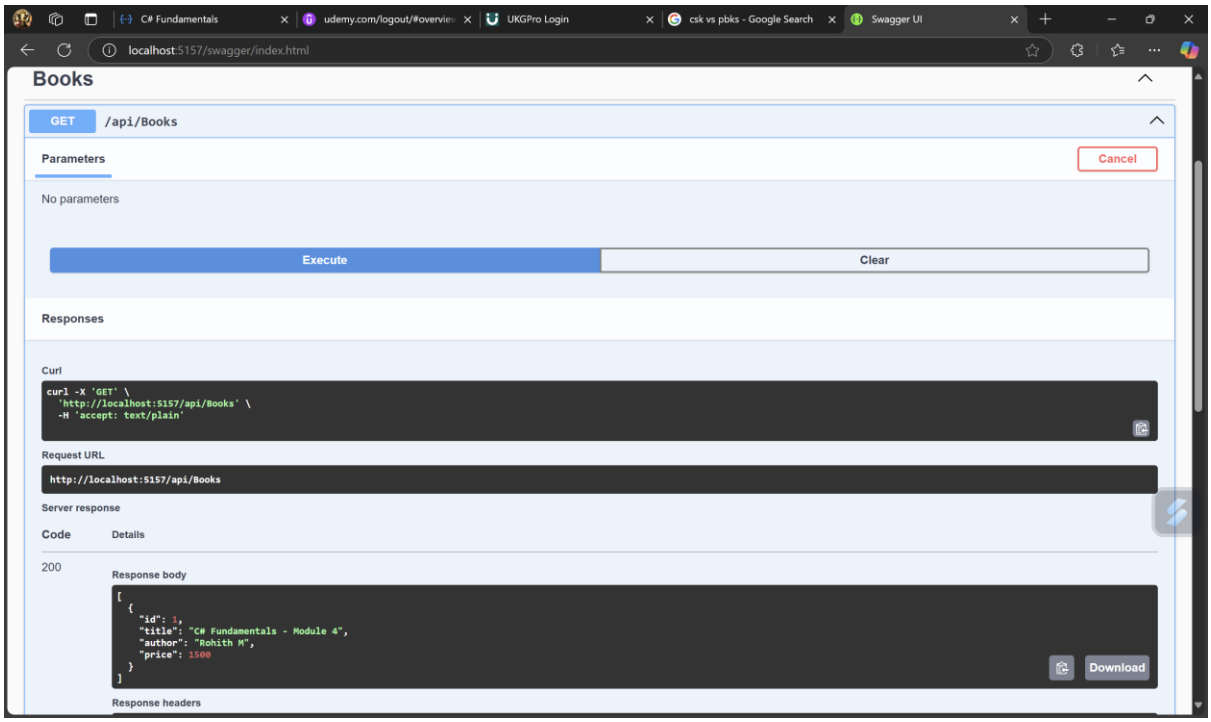
Responses

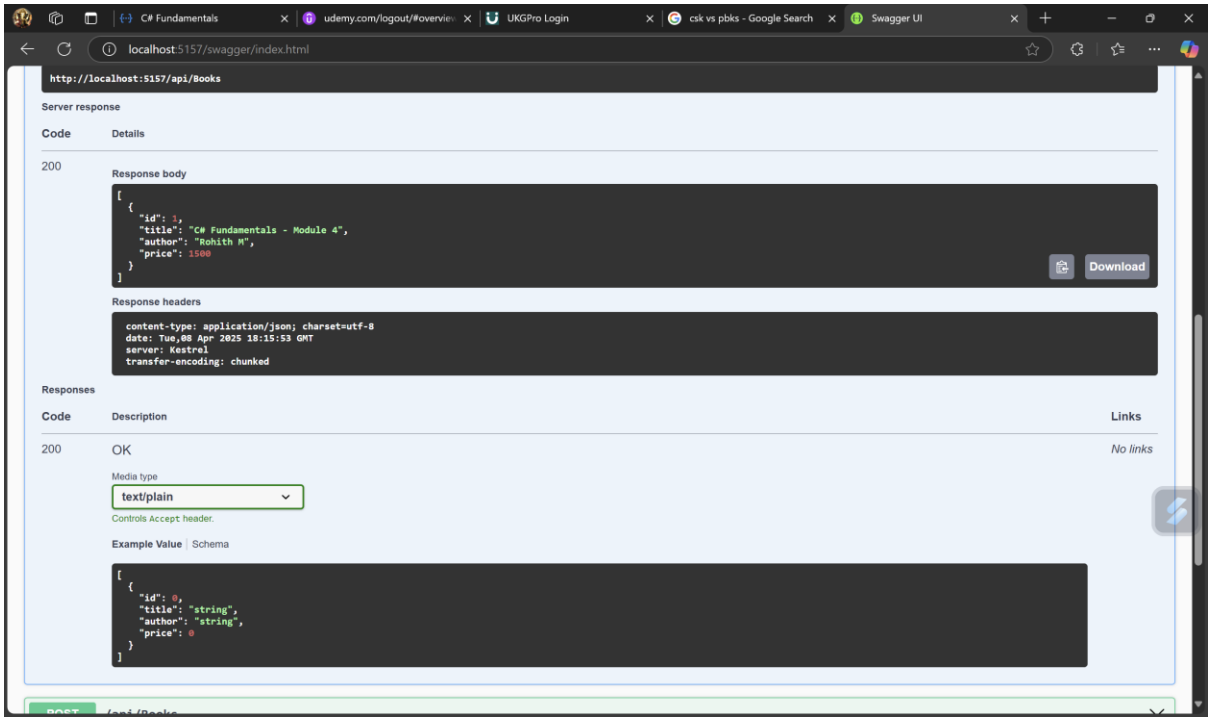
Code	Description	Links
200	OK Media type text/plain Controls Accept header. Example Value | Schema	*No links*

```
{
  "id": 0,
  "title": "string",
  "author": "string",
  "price": 0
}
```

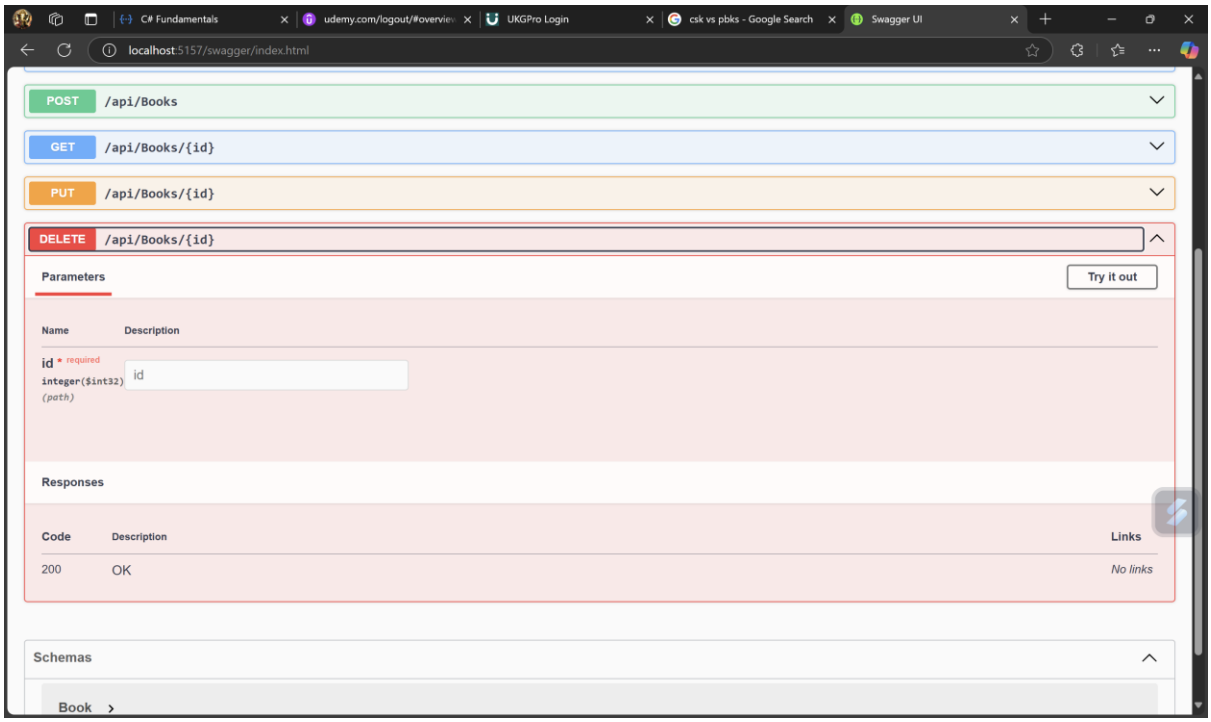
DELETE /api/Books/{id}

Now lets check the table using GET /api/Books

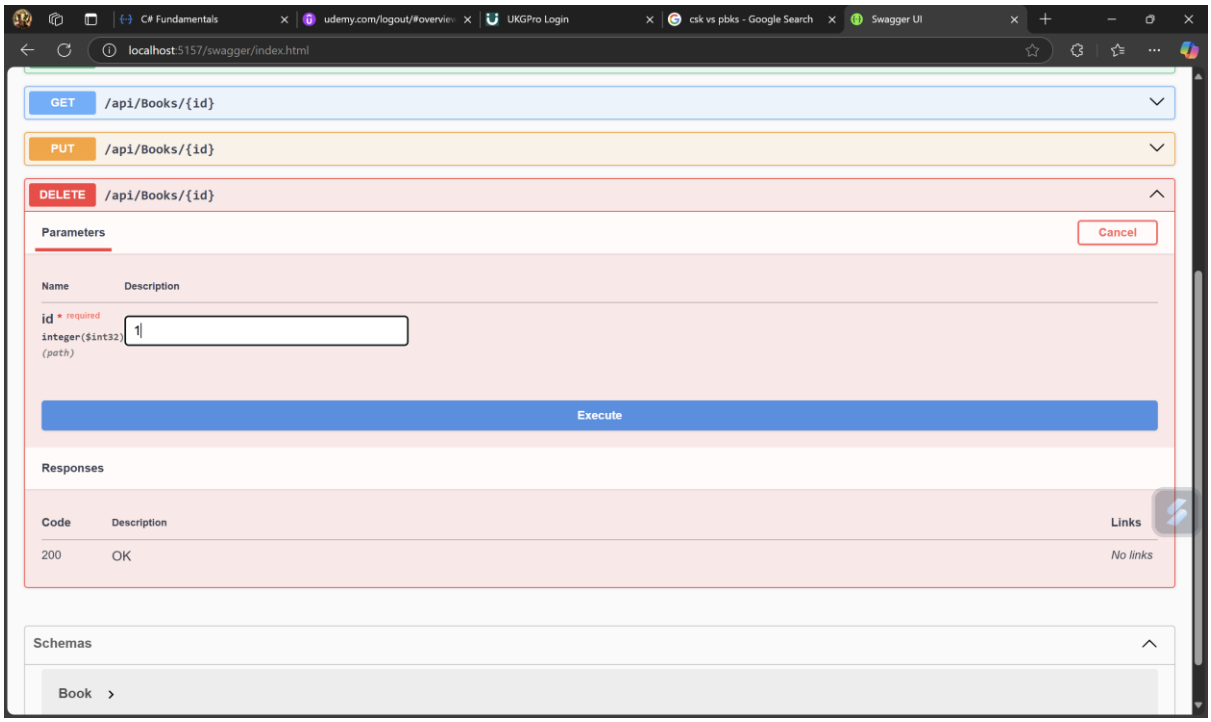


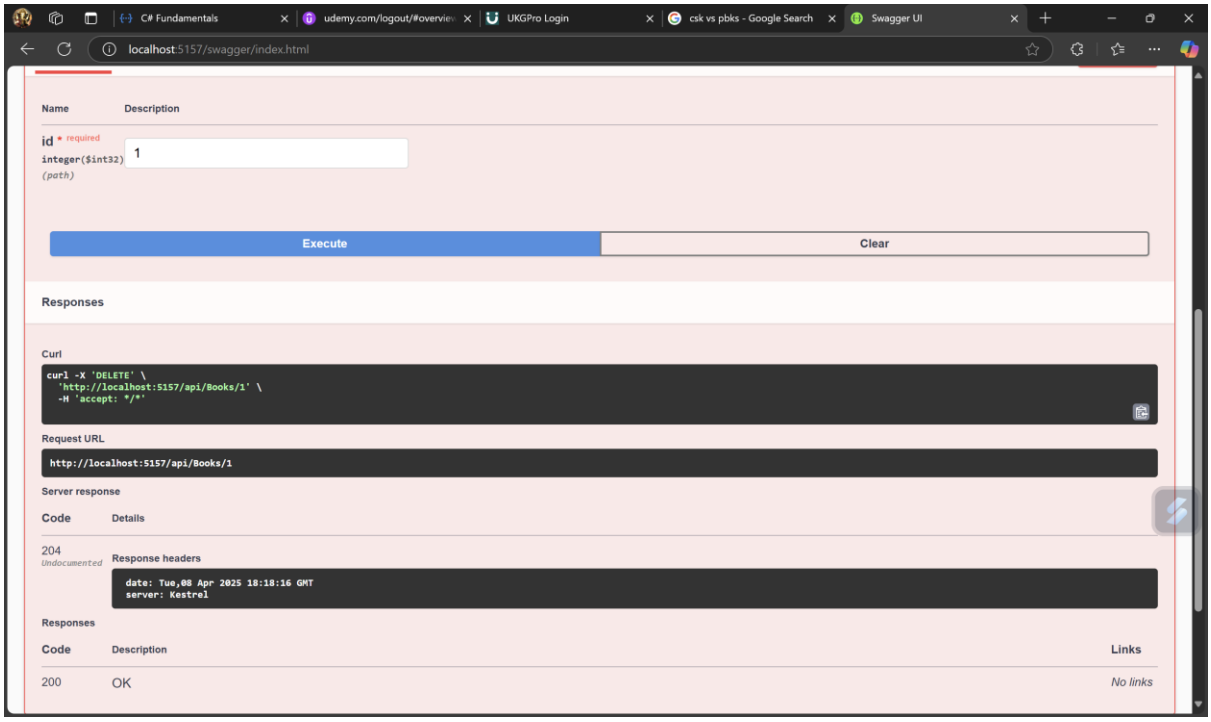


Now lets try DELETE /api/Books/{id}

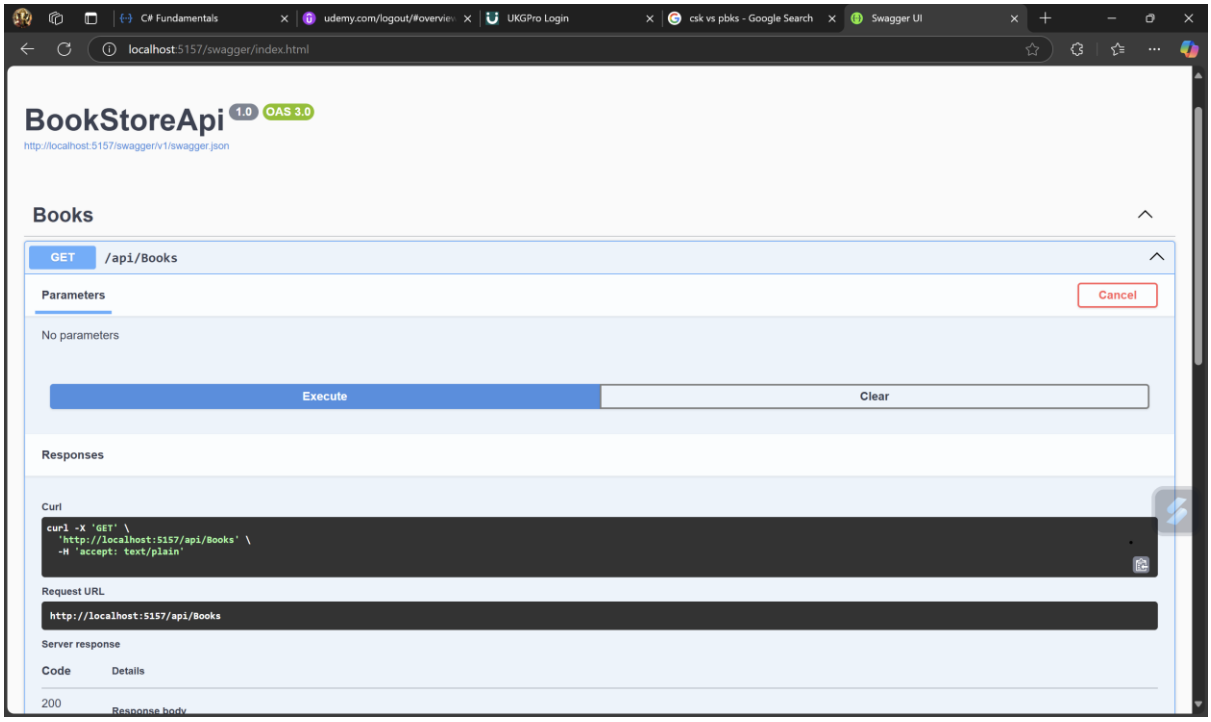


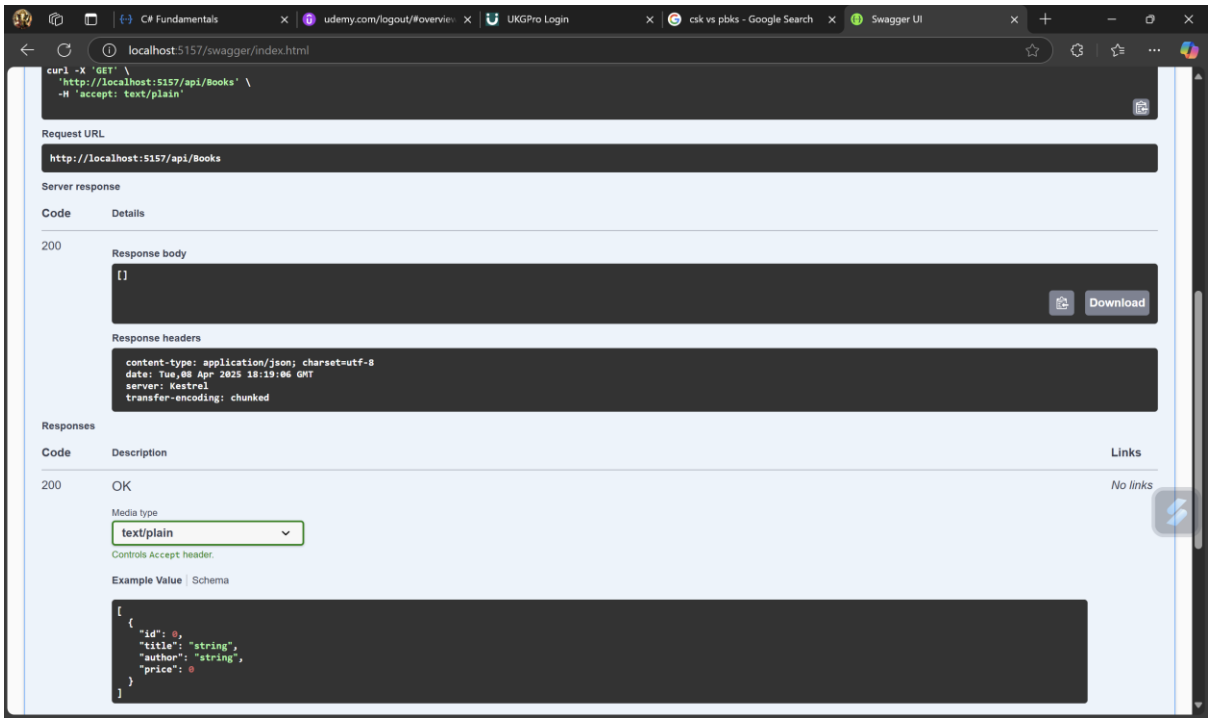
Now click on Try it out, Enter id and Execute

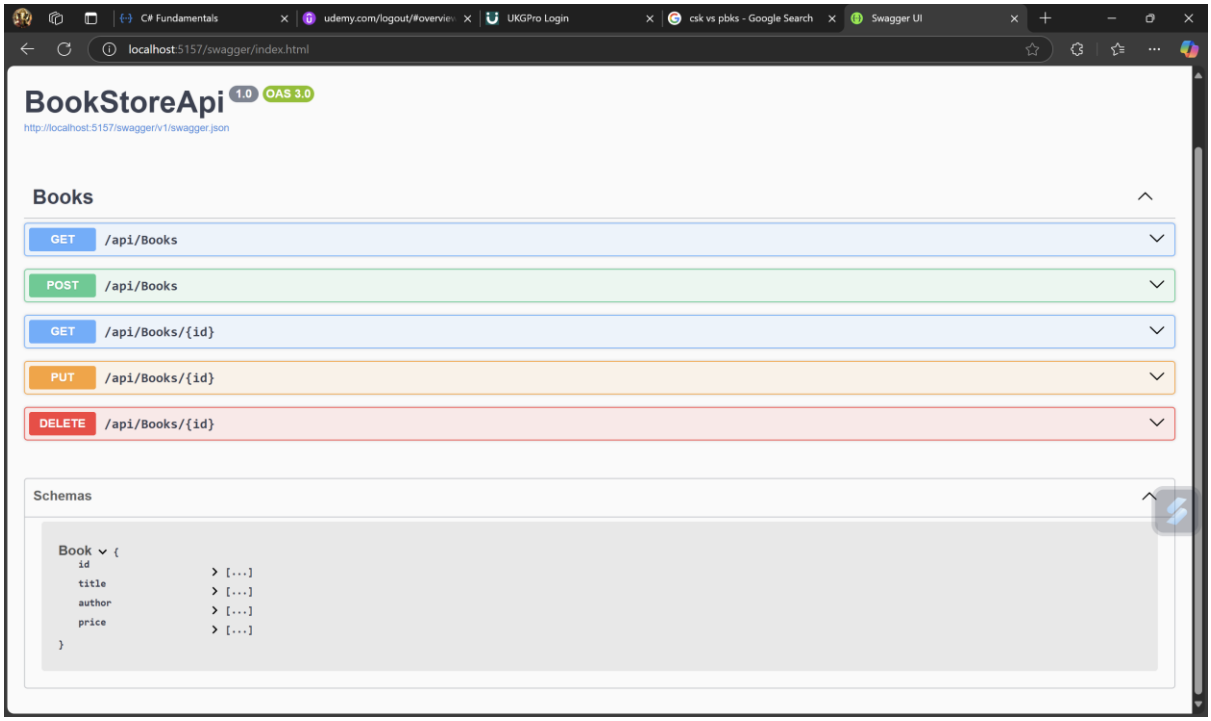




Now check the changes using GET /api/Books




C# Fundamentals
udemy.com/logout/#overvie
UKGPro Login
csk vs pbks - Google Search
Swagger UI
localhost:5157/swagger/index.html
curl -X 'GET' \
'http://localhost:5157/api/Books' \
-H 'accept: text/plain'
Request URL
http://localhost:5157/api/Books
Server response
Code
Details
200
Response body
[]
Download
Response headers
content-type: application/json; charset=utf-8
date: Tue,08 Apr 2025 18:19:06 GMT
server: Kestrel
transfer-encoding: chunked
Responses
Code
Description
Links
200
OK
No links
Media type
text/plain
Controls Accept header.
Example Value
Schema
[
{
"id": 0,
"title": "string",
"author": "string",
"price": 0
}
]


C# Fundamentals
udemy.com/logout/#overvie
UKGPro Login
csk vs pbks - Google Search
Swagger UI
localhost:5157/swagger/index.html
BookStoreApi
1.0
OAS 3.0
http://localhost:5157/swagger/v1/swagger.json
Books
GET /api/Books
POST /api/Books
GET /api/Books/{id}
PUT /api/Books/{id}
DELETE /api/Books/{id}
Schemas
Book {
id
title
author
price
}
[...]
[...]
[...]
[...]