Swagger JSON specifications for a bookstore web API

Table of Contents

Description

The Bookstore API is a .NET Core application featuring RESTful endpoints, GET, POST, PUT, DELETE, for managing bookstore inventory. This project demonstrates my ability to integrate Swagger for automatic API documentation using Swashbuckle.

Tools and technologies used

- Microsoft Visual Studio 2022
- .NET 8.0
- ASP.NET Core MVC
- Swashbuckle.AspNetCore 6.6.2

Code snippets

Program.cs

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(s =>
{
s.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo
{
Title = "API documentation for Bookstore",
Version = "v1",
Description = "API documentation for Bookstore",
Contact = new Microsoft.OpenApi.Models.OpenApiContact()
{
Name = "Priya Dalvi",
Email = "dalvipriya09@gmail.com"
},
});
var xmlPath = Path.Combine(System.AppContext.BaseDirectory, "bookstore.xml");
s.IncludeXmlComments(xmlPath);
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
```

Bookstore.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
                <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.6.2" />
  </ItemGroup>
</Project>
```

BooksController.cs

```
namespace Bookstore.Controllers
{
   [ApiController]
   [Route("[controller]")]
   public class BookStoreController : Controller

      // In-memory list to simulate a database
      private static List<Book> Books = new List<Book>
      {
         new Book { Id = 1, Title = "The Catcher in the Rye", Author = "J.D. Salinger", Price = 10.99m },
         new Book { Id = 2, Title = "To Kill a Mockingbird", Author = "Harper Lee", Price = 12.99m },
         new Book { Id = 3, Title = "1984", Author = "George Orwell", Price = 14.99m },
         new Book { Id = 4, Title = "The Great Gatsby", Author = "F. Scott Fitzgerald", Price = 9.99m },
         new Book { Id = 5, Title = "Moby Dick", Author = "Herman Melville", Price = 15.99m },
         new Book { Id = 6, Title = "Pride and Prejudice", Author = "Jane Austen", Price = 8.99m },
         new Book { Id = 7, Title = "War and Peace", Author = "Leo Tolstoy", Price = 20.99m },
         new Book { Id = 8, Title = "The Hobbit", Author = "J.R.R. Tolkien", Price = 13.99m },
         new Book { Id = 9, Title = "Crime and Punishment", Author = "Fyodor Dostoevsky", Price = 16.99m },
         new Book { Id = 10, Title = "The Alchemist", Author = "Paulo Coelho", Price = 11.99m }
      };

      // GET: api/BookStore
      /// <summary>
      /// Get details of all the books from the store.
      /// </summary>
      /// <remarks>This API should be used to fetch all books from the store.</remarks>
      /// <returns></returns>
      /// <response code="200">Successfull excecution</response>
      /// <response code="500">Internal server error found</response>

      [HttpGet]
      public IActionResult GetAllBooks()
      {
         return Ok(Books);
      }
   }
```

Output: Swagger JSON specifications


```
{
  "openapi": "3.0.1",
  "info": {
    "title": "API documentation for Bookstore",
    "description": "API documentation for Bookstore",
    "contact": {
      "name": "Priya Dalvi",
      "email": "dalvipriya09@gmail.com"
    },
    "version": "v1"
  },
  "paths": {
    "/BookStore": {
      "get": {
        "tags": [
          "BookStore"
        ],
        "summary": "Get details of all the books from the store.",
        "description": "This API should be used to fetch all books from the
store.",
        "responses": {
          "200": {
            "description": "Successfull excecution"
          },
          "500": {
            "description": "Internal server error found"
          }
        }
      },
      "post": {
        "tags": [
          "BookStore"
```

```
        ],
        "summary": "Create a new book.",
        "description": "This API should be used to add details of a new
book in the system.",
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            },
            "text/json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            },
            "application/*+json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      }
    },
    "/BookStore/{id}": {
      "get": {
        "tags": [
          "BookStore"
        ],
        "summary": "Get details of the books using a book ID.",
        "description": "This API should be used to fetch books using a book
ID.",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int32"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      },
      "put": {
        "tags": [
          "BookStore"
        ],
        "summary": "Update details of the books.",
```

```
        "description": "This API should be used to update a book that you
can first search using a book ID.",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int32"
            }
          }
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            },
            "text/json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            },
            "application/*+json": {
              "schema": {
                "$ref": "#/components/schemas/Book"
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      },
      "delete": {
        "tags": [
          "BookStore"
        ],
        "summary": "Delete a book.",
        "description": "This API should be used to delete a books using a
book ID.",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "integer",
              "format": "int32"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK"
          }
        }
```

```
      }
    },
    "/BookStore/author/{author}": {
      "get": {
        "tags": [
          "BookStore"
        ],
        "summary": "Get details of the books using the author name.",
        "description": "This API should be used to fetch books using
author's name.",
        "parameters": [
          {
            "name": "author",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      }
    },
    "/WeatherForecast": {
      "get": {
        "tags": [
          "WeatherForecast"
        ],
        "operationId": "GetWeatherForecast",
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "text/plain": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/WeatherForecast"
                  }
                }
              },
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/WeatherForecast"
                  }
                }
              },
              "text/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/WeatherForecast"
                  }
                }
              }
```

```
            }
          }
        }
      }
    }
  },
  "components": {
    "schemas": {
      "Book": {
        "type": "object",
        "properties": {
          "id": {
            "type": "integer",
            "format": "int32"
          },
          "title": {
            "type": "string",
            "nullable": true
          },
          "author": {
            "type": "string",
            "nullable": true
          },
          "price": {
            "type": "number",
            "format": "double"
          }
        },
        "additionalProperties": false
      },
      "WeatherForecast": {
        "type": "object",
        "properties": {
          "date": {
            "type": "string",
            "format": "date"
          },
          "temperatureC": {
            "type": "integer",
            "format": "int32"
          },
          "temperatureF": {
            "type": "integer",
            "format": "int32",
            "readOnly": true
          },
          "summary": {
            "type": "string",
            "nullable": true
          }
        },
        "additionalProperties": false
      }
    }
  }
}
```