Scenario :

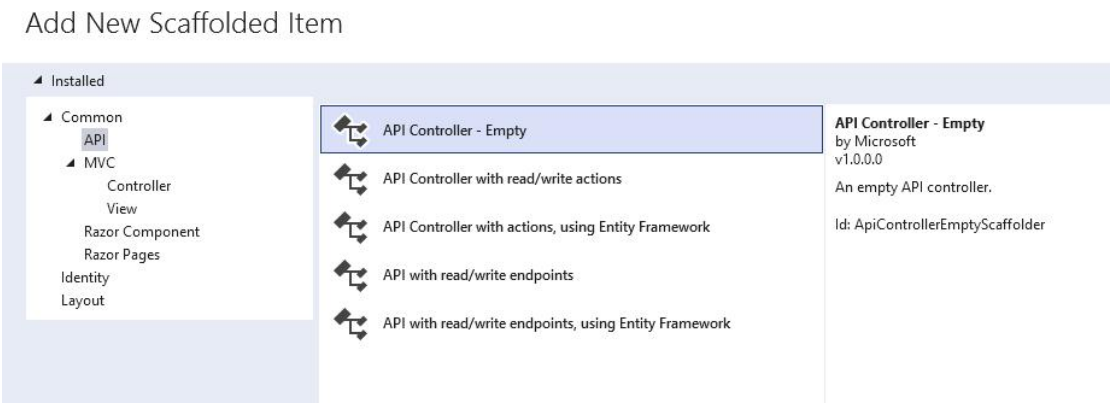
Sarah is an avid reader and has decided to open a small independent bookshop in her neighborhood. She wants to create a Web API application to manage and showcase the available books to customers online.

Help her by creating a web API application and making her work more efficient.

Exercise Steps :

1. Create a new web API core project in visual studio and choose the .net 6.0 framework.

2. Add a new controller in the Controller folder called "**BookController**".



Make sure that the controller is an empty API Controller.

3. Create a new folder called **Models**. Inside that folder, create a class named **Book.cs**. Declare the mentioned properties in the Book.cs class.

```
public class Book
{
    public int Id { get; set; }

    public string Title { get; set; }

    public string Author { get; set; }

    public double Price { get; set; }
}
```

4. Create a new folder called **Interface**. Inside that folder, create an interface named **IBookRepository.cs**.

Declare the given method in the IBookRepository.cs interface.

Method
public bool AddBook(Book book)

```
public interface IBookRepository
{
    bool AddBook(Book book);
}
```

5. Create a new folder called **Repository**. Inside that folder, create two classes, namely **StaticData**.cs and **BookRepository**.cs.

Declare the given list of objects in the **StaticData**.cs class.

```
public class StaticData
{
    public static List<Book> IsBooks  = new List<Book>
    {
        new Book
        {
            Id = 1,
            Title = "To Kill a Mockingbird",
            Author = "Harper Lee",
            Price = 100.99
        },
        new Book
        {
            Id = 2,
            Title = "1984",
            Author = "George Orwell",
            Price = 90.99
        },
        new Book
        {
            Id = 3,
            Title = "The Great Gatsby",
            Author = "F.Scott Fitzgerald",
            Price = 85.49
        },
        new Book
        {
            Id = 4,
            Title = "Pride and Prejudice",
            Author = "Jane Austen",
            Price = 78.99
        },
        new Book
        {
            Id = 5,
            Title = "The Catcher in the Rye",
            Author = "J.D.Salinger",
            Price = 65.79
        }
    };
}
```

6. Implement the **interface** methods in the **BookRepository**.cs. The logic must add the Book object to the list of objects **IsBooks**.

Method	Functionality
public bool AddBook(Book book)	This method is used to add the book object to the **IsBooks** list. If it's added successfully, it returns **true**; if not, it returns **false**.

```
public class BookRepository: IBookRepository
{
    public bool AddBook(Book book)
    {
        Book existingBook = StaticData.IsBooks.Find(b => b.Id == book.Id);

        if (existingBook == null)
        {
            StaticData.IsBooks.Add(book);
            return true;
        }
        return false;
    }
}
```

7. Now all the declarations and definitions have been given and done.

Let us move to the **BookController**.cs and invoke the method present in the interface **IBookRepository**.cs using the dependency injection.

In the controller, we are passing the **Book** object as an input parameter from the body and this is an **HttpPost** method with the route of **[Route("api/[controller]/AddBook")]**.

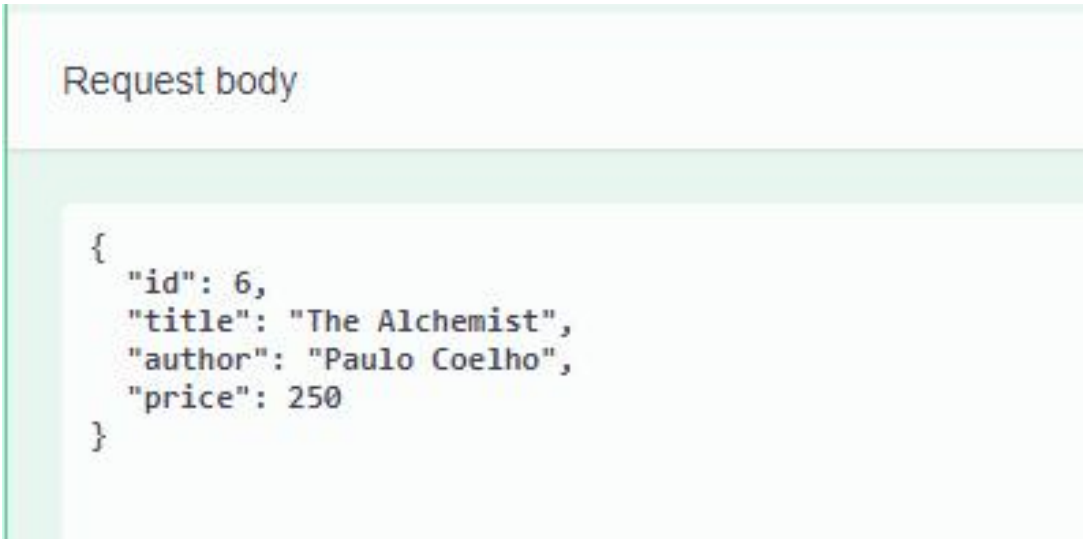
Then we call the method **AddBook** present in the interface. If the returned result is **true**, then return **Ok** with the returned **result**. Otherwise, return the **Status code 500**.

Service	Http Type & Return Type	Functionality	Repository (Check BookRepository Section)
api/[controller]/ AddBook	POST Method IActionResult	This service is used to add the book into the book list. If the result is false, it should return the status code 500	Call "AddBook"

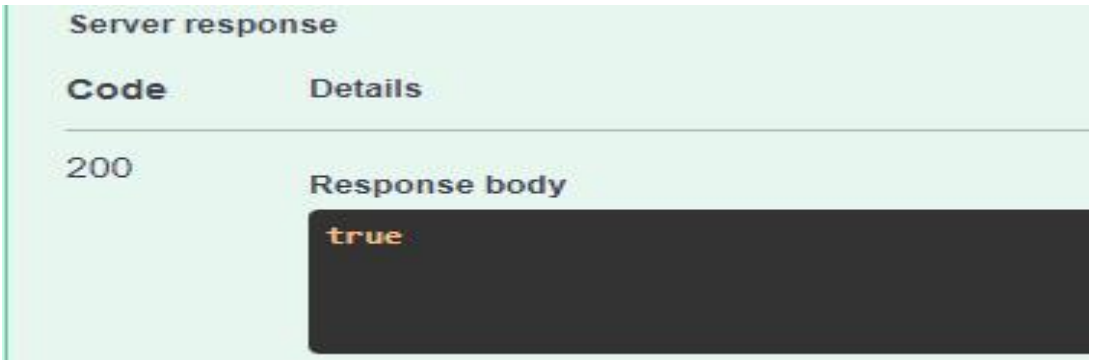
```
[Route("api/[controller]")]
[ApiController]
public class BookController : ControllerBase
{
    private readonly IBookRepository _bookRepository;

    public BookController(IBookRepository bookRepository)
    {
        _bookRepository = bookRepository;
    }

    [HttpPost("AddBook")]
    public IActionResult AddBook([FromBody] Book book)
    {
        var result = _bookRepository.AddBook(book);
        if (result)
        {
            return Ok(result);
        }
        return StatusCode(500);
    }
}
```

8. We have completed a **POST** web API with the necessary declarations and definitions. Let us run the project and see the output.

Pass the input in the body,



We can see the output below,



So, the input is added to the IsBooks list of objects.

To Summarize,

We have learned about how to add data using the web API POST method and about the dependency injection concept. We also learned about how to navigate a web API and pass data through the body, and about the status codes.