Lab - 3

Lab Program - 3

Create a Spring Boot Application using Maven Plugin

- Write a sample REST Controller API using Spring Annotations

- Using Postman invoke the REST Controller to demonstrate end to end working

Step 1: Generate a Spring Boot Project(Setting Up the Project on Spring Initializer:)

- Navigate to 'start.spring.io'
- Configure the project details.

```
Project: Maven
Language: Java
Spring Boot: 3.2.2 (SNAPSHOT)
Group: com.lab3
Artifact: my_lab3
Name: my_lab3
Description: Lab Program 3
Package name: com.lab3.my_lab3
Packaging: Jar
Java: 17
```

- Add the following dependencies: Spring Web
- Click on the "Generate" button to download the project as a ZIP file.
- Extract the downloaded ZIP file.

Step 2: Extract and Import the Project into Eclipse

- Open Eclipse.
- Go to "File" -> "Import" -> "Existing Maven Projects."
- Select the root directory of the extracted project and click "Finish."

Step 3: Create Book.java and BookController.java under src/main/java folder

Book.java

```java
package com.lab3.my_lab3;

public class Book {

    private Long id;
    private String title;
    private String author;
    private int publicationYear;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public int getPublicationYear() {
        return publicationYear;
    }

    public void setPublicationYear(int publicationYear) {
        this.publicationYear = publicationYear;
    }
}
```

BookController.java

```java
package com.lab3.my_lab3;

import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/api/books")
public class BookController {

    private final List<Book> books = new ArrayList<Book>();

    @GetMapping
    public List<Book> getAllBooks() {
        return books;
    }

    @PostMapping
    public Book addBook(@RequestBody Book book) {
        books.add(book);
        return book;
    }
}
```

Step 4: Build and Run the Project

- Right-click on the project in the Project Explorer.
- Select "Run As" -> "Java Application"

Step 5: Test the API

- Open Postman
- **Add Book (POST Request):**
  - Set the request type to POST.
  - Enter the URL: http://localhost:8080/api/books
  - Go to the "Body" tab.
  - Select raw and set the data type to JSON (application/json).
  - Copy and paste the following JSON array into the request body:

```
{
    "id":1,
    "title":"My Book  - 1",
    "author":"Sitomi",
    "publicationYear":2024
    }
```

- Click the "Send" button.
- This will add the book to our Spring Boot application with a single request.

- **Retrieve Books(GET Request):**
  - Set the request type to GET.
  - Enter the URL: for example: http://localhost:8080/api/books
  - Click the "Send" button.
  - This will retrieve the details of the book/s.

Lab Program - 4

Write a sample REST App to demonstrate below Concepts with a use-case of your choice, GET, PUT, POST, DELETE

OR

Create a Spring Boot Application using Maven Plugin

- Write a sample REST Controller API using Spring Annotations

- Product details using GET, POST, PUT and DELETE methods.

- Using Postman invoke the REST Controller to demonstrate end to end working

Step 1: Generate a Spring Boot Project(Setting Up the Project on Spring Initializer:)

- Navigate to 'start.spring.io'
- Configure the project details.

```
Project: Maven
Language: Java
Spring Boot: 3.2.2 (SNAPSHOT)
Group: com.lab4
Artifact: my_lab4
Name: my_lab4
Description: Lab Program 4
```

```
Package name: com.lab4.my_lab4
Packaging: Jar
Java: 17
```

- Add the following dependencies: Spring Web
- Click on the "Generate" button to download the project as a ZIP file.
- Extract the downloaded ZIP file.

Step 2: Extract and Import the Project into Eclipse

- Open Eclipse.
- Go to "File" -> "Import" -> "Existing Maven Projects."
- Select the root directory of the extracted project and click "Finish."

Step 3: Create Book.java and BookController.java under src/main/java folder

Book.java

```java
package com.lab4.my_lab4;

public class Book {
    private Long id;
    private String title;
    private String author;
    private int publicationYear;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getAuthor() {
        return author;
    }
```

```
    public void setAuthor(String author) {
        this.author = author;
    }
    public int getPublicationYear() {
        return publicationYear;
    }
    public void setPublicationYear(int publicationYear) {
        this.publicationYear = publicationYear;
    }


}
```

BookController.java

```
package com.lab4.my_lab4;

import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/api/books")
public class BookController {
    private final List<Book> books = new ArrayList<>();
    private Long bookIdCounter = 1L; // Initialize counter to 1

    @GetMapping
    public List<Book> getAllBooks() {
        return books;
    }

    @GetMapping("/{id}")
    public Book getBookById(@PathVariable Long id) {
        return findBookById(id);
    }

    @PostMapping
    public Book addBook(@RequestBody Book book) {
        book.setId(generateBookId());
        books.add(book);
        return book;
    }
```

```
    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody Book
updatedBook) {
        Book existingBook = findBookById(id);
        if (existingBook != null) {
            existingBook.setTitle(updatedBook.getTitle());
            existingBook.setAuthor(updatedBook.getAuthor());

existingBook.setPublicationYear(updatedBook.getPublicationYear());
        }
        return existingBook;
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        Book bookToRemove = findBookById(id);
        if (bookToRemove != null) {
                books.remove(bookToRemove);
        }
    }

    private Book findBookById(Long id) {
        for(Book book:books) {
                if(book.getId().equals(id)) {
                        return book;
                }
        }
        return null;
    }

    private Long generateBookId() {
        return bookIdCounter++;
    }
}
```

Step 4: Build and Run the Project

- Right-click on the project in the Project Explorer.
- Select "Run As" -> "Java Application"

Step 5: Test the API

- Open Postman
- **Add Book (POST Request):**
  - Set the request type to POST.
  - Enter the URL: http://localhost:8080/api/books
  - Go to the "Body" tab.
  - Select raw and set the data type to JSON (application/json).
  - Copy and paste the following JSON array into the request body:

```
{
    "title":"My Book  - 1",
    "author":"Sitomi",
    "publicationYear":2024
    }
```

  - Click the "Send" button.
  - This will add the book to our Spring Boot application with a single request.
- **Retrieve a Specific Book (GET Request):**
  - Set the request type to GET.
  - Enter the URL: http://localhost:8080/api/books/{id}, replacing {id} with the specific ID of the book you want to retrieve.
  - For example: http://localhost:8080/api/books/1 to retrieve the book with ID 1.
  - Click the "Send" button.
  - This will retrieve the details of the specific book.
- **Update a Specific Book (PUT Request):**
  - Set the request type to PUT.
  - Enter the URL: http://localhost:8080/api/books/{id}, replacing {id} with the specific ID of the book you want to update.
  - For example: http://localhost:8080/api/books/1 to update the book with ID 1.
  - Go to the "Body" tab.
  - Select raw and set the data type to JSON (application/json).
  - Copy and paste the updated details of the book into the request body.
  - For example:

```
{
    "title":"My Book  - 1",
    "author":"Sitomi - Momo - Fluffy",
    "publicationYear":2024
}
```

- Click the "Send" button.
- This will update the details of the specified book.

- **Delete a Specific Book (DELETE Request):**
  - Set the request type to DELETE.
  - Enter the URL: http://localhost:8080/api/books/{id}, replacing {id} with the specific ID of the book you want to delete.
  - For example: http://localhost:8080/api/books/1 to delete the book with ID 1.
  - Click the "Send" button.
  - This will delete the specified book from your Spring Boot application.