**SPRING CORE AND MAVEN**

**Exercise 1: Configuring a Basic Spring Application**

**Exercise 2: Implementing Dependency Injection**

**Exercise 4: Creating and Configuring a Maven Project**

**Code :**

**Book.java**

package com.library;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    public Book() {}

    public Book(String title) {

        this.title = title;

    }

    public Long getId() {

        return id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

}

**BookController.java**

package com.library;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

    private final BookRepository repo;

    public BookController(BookRepository repo) {

        this.repo = repo;

    }

    @GetMapping

    public List<Book> getAll() {

        return repo.findAll();

    }

    @PostMapping

    public ResponseEntity<Book> create(@RequestBody Book book) {

        return new ResponseEntity<>(repo.save(book), HttpStatus.CREATED);

    }

    @GetMapping("{id}")

    public ResponseEntity<Book> getById(@PathVariable Long id) {

        return repo.findById(id)

                .map(ResponseEntity::ok)

                .orElse(ResponseEntity.notFound().build());

    }

    @PutMapping("{id}")

    public ResponseEntity<Book> update(@PathVariable Long id, @RequestBody Book book) {

        return repo.findById(id)

                .map(existing -> {

                    existing.setTitle(book.getTitle());

                    return new ResponseEntity<>(repo.save(existing), HttpStatus.OK);

                })

                .orElse(ResponseEntity.notFound().build());

    }

    @DeleteMapping("{id}")

    public ResponseEntity<Void> delete(@PathVariable Long id) {

        if (repo.existsById(id)) {

            repo.deleteById(id);

            return ResponseEntity.noContent().build();

        }

        return ResponseEntity.notFound().build();

    }

}

**BookRepository.java**

package com.library;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {}

**LibraryManagementApplication.java**

package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryManagementApplication.class, args);

    }

}

**Resources**

**application.properties**

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

**Classes**

**Book.class**

package com.library;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

   @Id

   @GeneratedValue(

      strategy = GenerationType.IDENTITY

   )

   private Long id;

   private String title;

   public Book() {

   }

   public Book(String title) {

      this.title = title;

   }

   public Long getId() {

      return this.id;

   }

   public String getTitle() {

      return this.title;

   }

   public void setTitle(String title) {

      this.title = title;

   }

}

**BookController.class**

package com.library;

import java.util.List;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

@RestController

@RequestMapping({"/api/books"})

public class BookController {

   private final BookRepository repo;

   public BookController(BookRepository repo) {

      this.repo = repo;

   }

   @GetMapping

   public List<Book> getAll() {

      return this.repo.findAll();

   }

   @PostMapping

   public ResponseEntity<Book> create(@RequestBody Book book) {

      return new ResponseEntity((Book)this.repo.save(book), HttpStatus.CREATED);

   }

   @GetMapping({"{id}"})

   public ResponseEntity<Book> getById(@PathVariable Long id) {

      return (ResponseEntity)this.repo.findById(id).map(ResponseEntity::ok).orElse(ResponseEntity.notFound().build());

   }

   @PutMapping({"{id}"})

   public ResponseEntity<Book> update(@PathVariable Long id, @RequestBody Book book) {

      return (ResponseEntity)this.repo.findById(id).map((existing) -> {

         existing.setTitle(book.getTitle());

         return new ResponseEntity((Book)this.repo.save(existing), HttpStatus.OK);

      }).orElse(ResponseEntity.notFound().build());

   }

   @DeleteMapping({"{id}"})

   public ResponseEntity<Void> delete(@PathVariable Long id) {

      if (this.repo.existsById(id)) {

         this.repo.deleteById(id);

         return ResponseEntity.noContent().build();

      } else {

         return ResponseEntity.notFound().build();

      }

   }

}

**BookRepository.class**

package com.library;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**LibraryManagementApplication.class**

package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

   public LibraryManagementApplication() {

   }

   public static void main(String[] args) {

      SpringApplication.run(LibraryManagementApplication.class, args);

   }

}

**Resources**

**application.properties**

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>3.4.0</version>

        <relativePath/>

    </parent>

    <groupId>com.library</groupId>

    <artifactId>library-management-boot</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

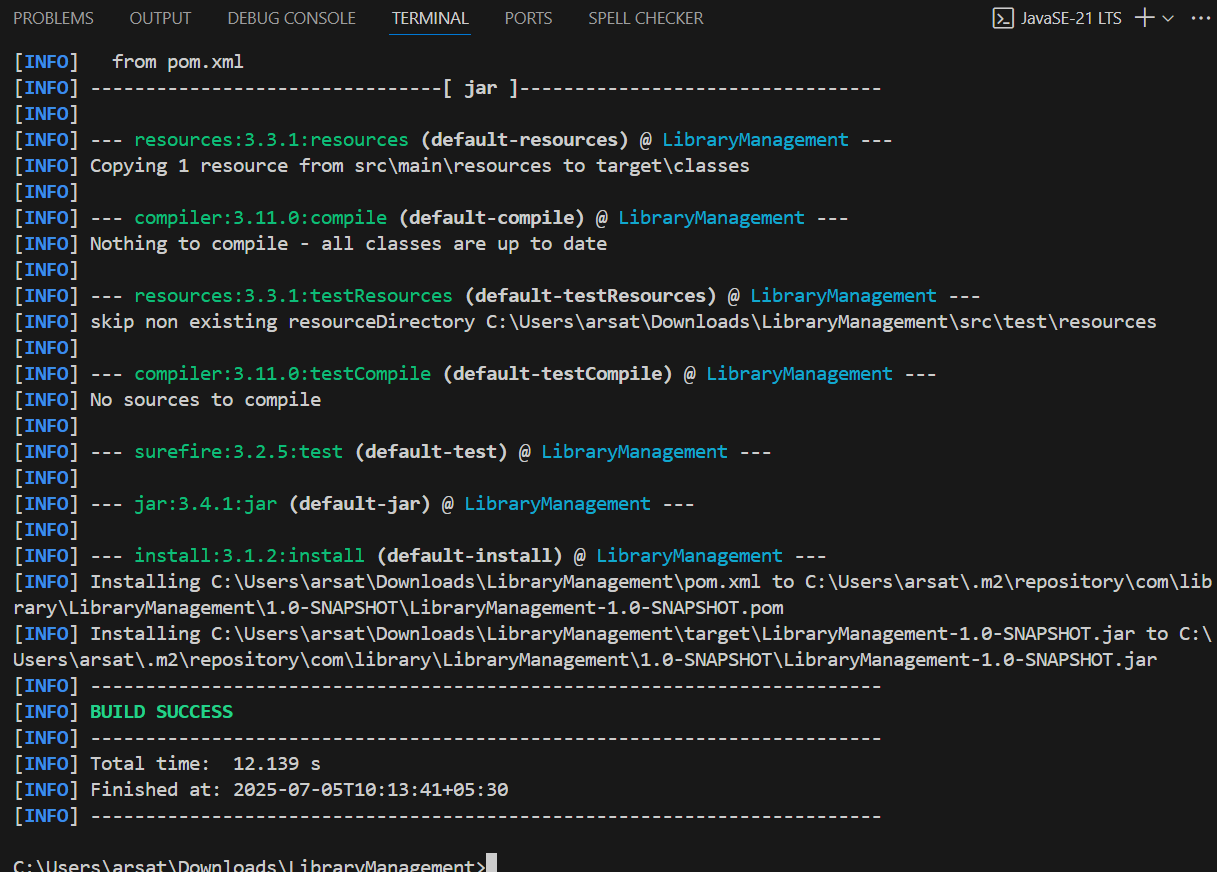
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT :**

****