Name: SASHMITHA R

Superset ID: 6387562

Exercise 5: Configuring the Spring IoC Container

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="bookRepository" class="com.example.BookRepository" />
  <bean id="bookService" class="com.example.BookService">
    <property name="bookRepository" ref="bookRepository" />
  </bean>
</beans>
```

BookService.java

```
package com.example;
public class BookService {
  private BookRepository bookRepository;
  public void setBookRepository(BookRepository bookRepository)
  {
    this.bookRepository = bookRepository;
  }
  public void displayBooks()
  {
    System.out.println("Calling BookRepository from BookService...");
    bookRepository.getBookDetails();
  }
}
```

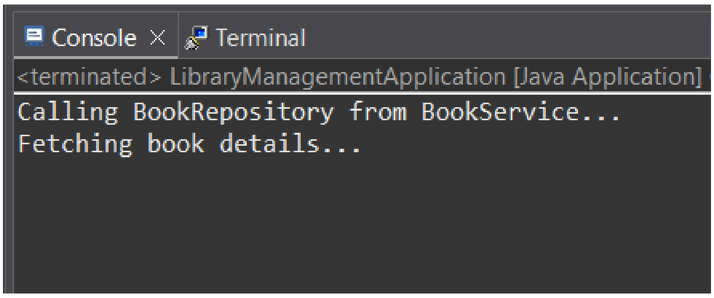
BookRepository.java

```
package com.example;

public class BookRepository

{

   public void getBookDetails()

   {

      System.out.println("Fetching book details...");

   }

}
```

LibraryManagementApplication.java

```
package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication

{

   public static void main(String[] args)

   {

      ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

      BookService bookService = (BookService) context.getBean("bookService");

      bookService.displayBooks();

   }

}
```

Output :



```
<terminated> LibraryManagementApplication [Java Application]
Calling BookRepository from BookService...
Fetching book details...
```

Exercise 7: Implementing Constructor and Setter Injection

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="bookRepository" class="com.example.BookRepository" />
  <bean id="bookService" class="com.example.BookService">
    <constructor-arg value="Library Service Bean" />
    <property name="bookRepository" ref="bookRepository" />
  </bean>
</beans>
```

BookService.java

```
package com.example;
public class BookService
{
  private String serviceName;
  private BookRepository bookRepository;
  public BookService(String serviceName)
  {
    this.serviceName = serviceName;
    System.out.println("Constructor injection: Service Name set to " + serviceName);
  }
  public void setBookRepository(BookRepository bookRepository)
  {
    this.bookRepository = bookRepository;
  }
  public void displayBooks()
```

```
    {
        System.out.println("Service: " + serviceName);
        if (bookRepository != null)
        {
            bookRepository.getBookDetails();
        }
        else
        {
            System.out.println("BookRepository is not injected!");
        }
    }
}
```

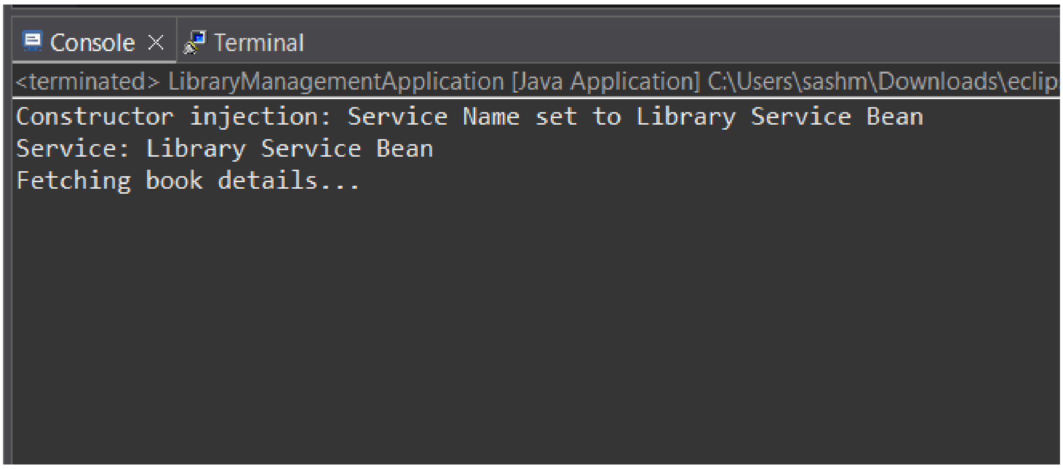
BookRepository.java

```
package com.example;
public class BookRepository
{
    public void getBookDetails()
    {
        System.out.println("Fetching book details...");
    }
}
```

LibraryManagementApplication.java

```
package com.example;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class LibraryManagementApplication
{
    public static void main(String[] args)
    {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");

        bookService.displayBooks();

    }

}
```

Output :



```
<terminated> LibraryManagementApplication [Java Application] C:\Users\sashm\Downloads\eclip
Constructor injection: Service Name set to Library Service Bean
Service: Library Service Bean
Fetching book details...
```

Exercise 9: Creating a Spring Boot Application

Book.java

```
package com.example;
import jakarta.persistence.*;
@Entity
public class Book
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String title;
  private String author;
  public Book() {}
  public Book(String title, String author) {
    this.title = title;
    this.author = author;
  }
  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; }
}
```

BookController.java

```
package com.example;
import org.springframework.web.bind.annotation.*;
import java.util.List;
```

```
@RestController
@RequestMapping("/books")
public class BookController
{
    private final BookRepository bookRepository;
    public BookController(BookRepository bookRepository)
    {
        this.bookRepository = bookRepository;
    }
    @PostMapping("/bulk")
    public List<Book> addBooks(@RequestBody List<Book> books)
    {
        return bookRepository.saveAll(books);
    }
}
```

BookRepository.java

```
package com.example;
import org.springframework.data.jpa.repository.JpaRepository;
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

LibraryManagementApplication.java

```
package com.example;
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);
       }
}
```

CorsConfig.java

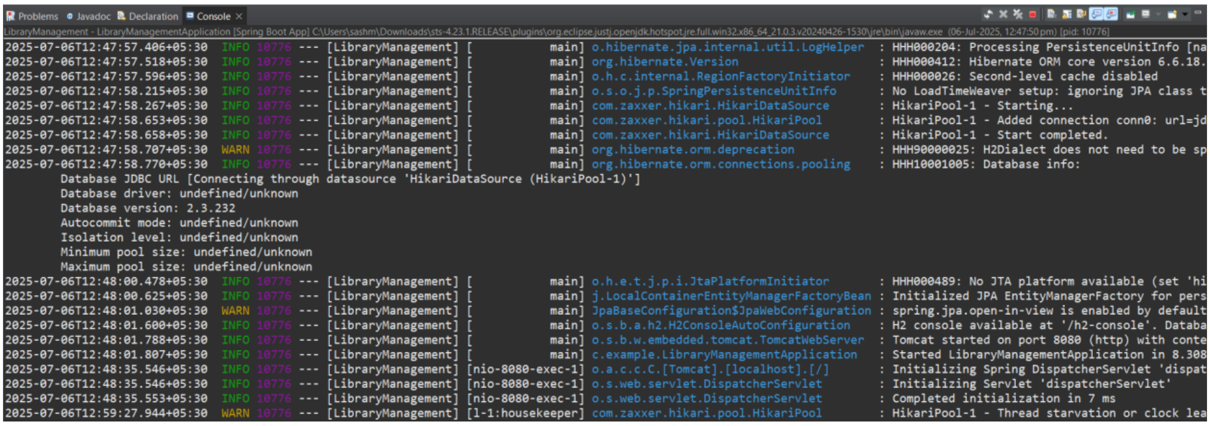
```
package com.example;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.*;
@Configuration
public class CorsConfig
{
  @Bean
  public WebMvcConfigurer corsConfigurer()
  {
    return new WebMvcConfigurer() {
      @Override
      public void addCorsMappings(CorsRegistry registry) {
            registry.addMapping("/**")
            .allowedOrigins("*")
            .allowedMethods("GET", "POST", "PUT", "DELETE")
            .allowedHeaders("*");
      }
    };
  }
}
```

application.properties

```
spring.application.name=LibraryManagement
# H2 In-Memory DB Settings
spring.datasource.url=jdbc:h2:mem:librarydb
spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=


# JPA Settings

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update


# Enable H2 Console

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console
```

Output :

LibraryManagementApplication.java -> Run As -> SpringBoot App



```
LibraryManagement - LibraryManagementApplication [Spring Boot App] C:\Users\sashm\Downloads\sts-4.23.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe  (06-Jul-2025, 12:47:50 pm) [pid: 10776]
2025-07-06T12:47:57.406+05:30  INFO 10776 --- [LibraryManagement] [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [na
2025-07-06T12:47:57.518+05:30  INFO 10776 --- [LibraryManagement] [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.6.18.
2025-07-06T12:47:57.596+05:30  INFO 10776 --- [LibraryManagement] [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-07-06T12:47:58.215+05:30  INFO 10776 --- [LibraryManagement] [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class t
2025-07-06T12:47:58.267+05:30  INFO 10776 --- [LibraryManagement] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-06T12:47:58.653+05:30  INFO 10776 --- [LibraryManagement] [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection conn0: url=jd
2025-07-06T12:47:58.658+05:30  INFO 10776 --- [LibraryManagement] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-06T12:47:58.707+05:30  WARN 10776 --- [LibraryManagement] [           main] org.hibernate.orm.deprecation            : HHH90000025: H2Dialect does not need to be sp
2025-07-06T12:47:58.770+05:30  INFO 10776 --- [LibraryManagement] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 2.3.232
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-07-06T12:48:00.478+05:30  INFO 10776 --- [LibraryManagement] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hi
2025-07-06T12:48:00.625+05:30  INFO 10776 --- [LibraryManagement] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for pers
2025-07-06T12:48:01.030+05:30  WARN 10776 --- [LibraryManagement] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default
2025-07-06T12:48:01.600+05:30  INFO 10776 --- [LibraryManagement] [           main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Databa
2025-07-06T12:48:01.788+05:30  INFO 10776 --- [LibraryManagement] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with conte
2025-07-06T12:48:01.807+05:30  INFO 10776 --- [LibraryManagement] [           main] c.example.LibraryManagementApplication   : Started LibraryManagementApplication in 8.308
2025-07-06T12:48:35.546+05:30  INFO 10776 --- [LibraryManagement] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispat
2025-07-06T12:48:35.546+05:30  INFO 10776 --- [LibraryManagement] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-06T12:48:35.553+05:30  INFO 10776 --- [LibraryManagement] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 7 ms
2025-07-06T12:59:27.944+05:30  WARN 10776 --- [LibraryManagement] [l-1:housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Thread starvation or clock lea
```

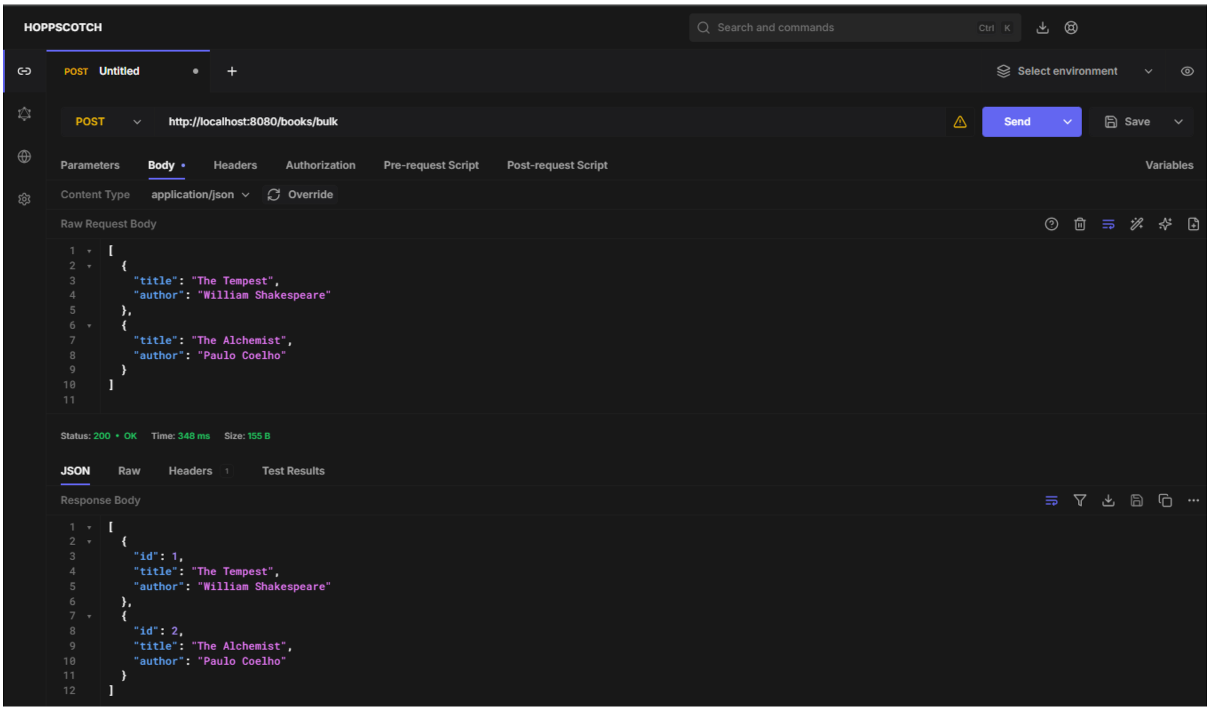
Hoppscotch

Method: POST

URL: http://localhost:8080/books/bulk

Header: Content-Type: application/json

```
[{
   "title": "The Tempest",
   "author": "William Shakespeare"
 },
 {
   "title": "The Alchemist",
   "author": "Paulo Coelho"
 }
]
```



POST http://localhost:8080/books/bulk

Content Type application/json

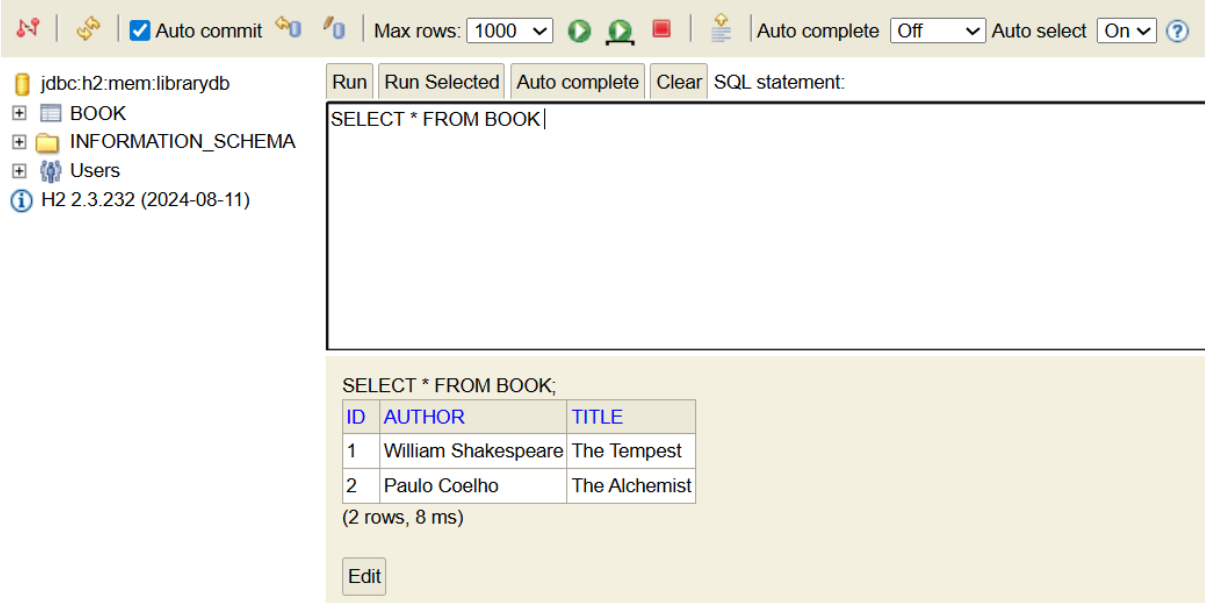
```
[
  {
    "title": "The Tempest",
    "author": "William Shakespeare"
  },
  {
    "title": "The Alchemist",
    "author": "Paulo Coelho"
  }
]
```

Status: 200 • OK Time: 348 ms Size: 155 B

```
[
  {
    "id": 1,
    "title": "The Tempest",
    "author": "William Shakespeare"
  },
  {
    "id": 2,
    "title": "The Alchemist",
    "author": "Paulo Coelho"
  }
]
```

H2 Console

Select * from BOOK;



```
SELECT * FROM BOOK
```

SELECT * FROM BOOK;

ID	AUTHOR	TITLE
1	William Shakespeare	The Tempest
2	Paulo Coelho	The Alchemist

(2 rows, 8 ms)