7.Basic Spring Boot Application with Spring Data JPA

Aim: To create the Basic Spring Boot Application with Spring Data JPA

Description:

This project demonstrates how to create a basic Spring Boot application integrated with Spring Data JPA. It provides a simple setup for building Java applications that interact with relational databases using JPA and Hibernate. The application includes:

- Spring Boot for rapid application development and simplified configuration.
- Spring Data JPA for database operations such as saving, updating, deleting, and retrieving records without writing boilerplate SQL queries.
- Entity classes to represent database tables.
- Repository interfaces for CRUD operations.
- REST controllers (optional) to expose endpoints for interacting with the database.

This serves as a foundational template for beginners to understand how to set up Spring Boot with JPA and can be extended for real-world applications.

Program:

Student.java

```
package com.example;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Student {
        @Id
        private int sno;
        private String sname;

        public Student() {}
        public Student(int sno, String sname) {
                super();
                this.sno = sno;
                this.sname = sname;
        }
        public int getSno() {return sno;}
        public void setSno(int sno) {this.sno = sno;}
        public String getSname() {return sname;}
        public void setSname(String sname) {this.sname = sname;}

}
```

StudentController.java

```
package com.example;

import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMapping;

@RestController
//@RequestMapping("/students")
public class StudentController {
	private final StudentRepository repo;
	public StudentController(StudentRepository repo) {
		this.repo = repo;

	}

	//@RequestMapping("/students")
	//Add new student
	@PostMapping
	public Student addStudent(@RequestBody Student student) {
		return repo.save(student);
	}
	//Get all students
	@GetMapping
	public List<Student> getAllStudents(){
		return repo.findAll();
	}
}
```

StudentRepository.java

```
package com.example;

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

public interface StudentRepository extends JpaRepository<Student,Integer> {

}
```

U3StudentApplication.java

```
package com.example;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class U3StudentApplication {
	public static void main(String[] args) {
		SpringApplication.run(U3StudentApplication.class, args);
	}

	@Bean
	CommandLineRunner initDatabase(StudentRepository repo) {
		return args ->{
			repo.save(new Student(1,"Swarupa"));
			repo.save(new Student(2,"Gayatri"));
			repo.save(new Student(3,"Sridevi"));
			System.out.println("Students inserted!");
		};
	}
}
```

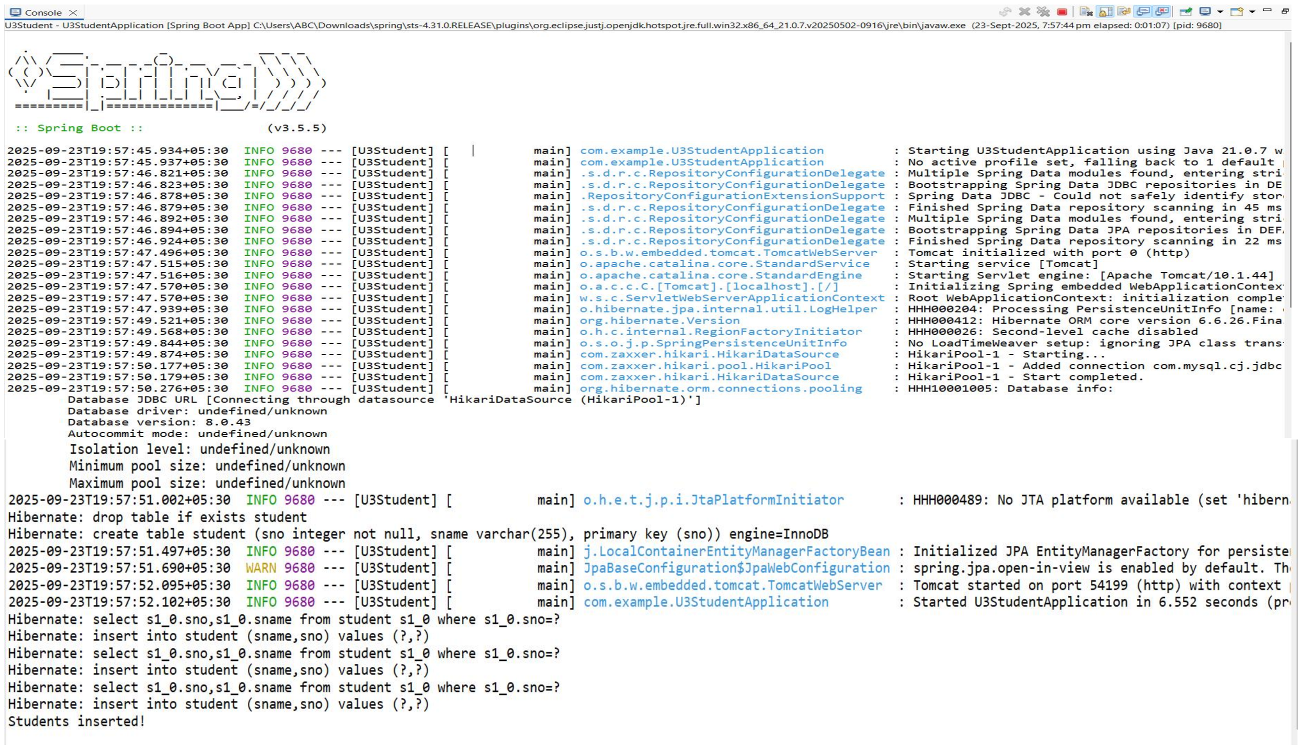
application.properties

```
spring.application.name=U3Student
server.port=0000
spring.datasource.url=jdbc:mysql://localhost:3306/demo
spring.datasource.username=root
spring.datasource.password=Swarupa@123
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
```

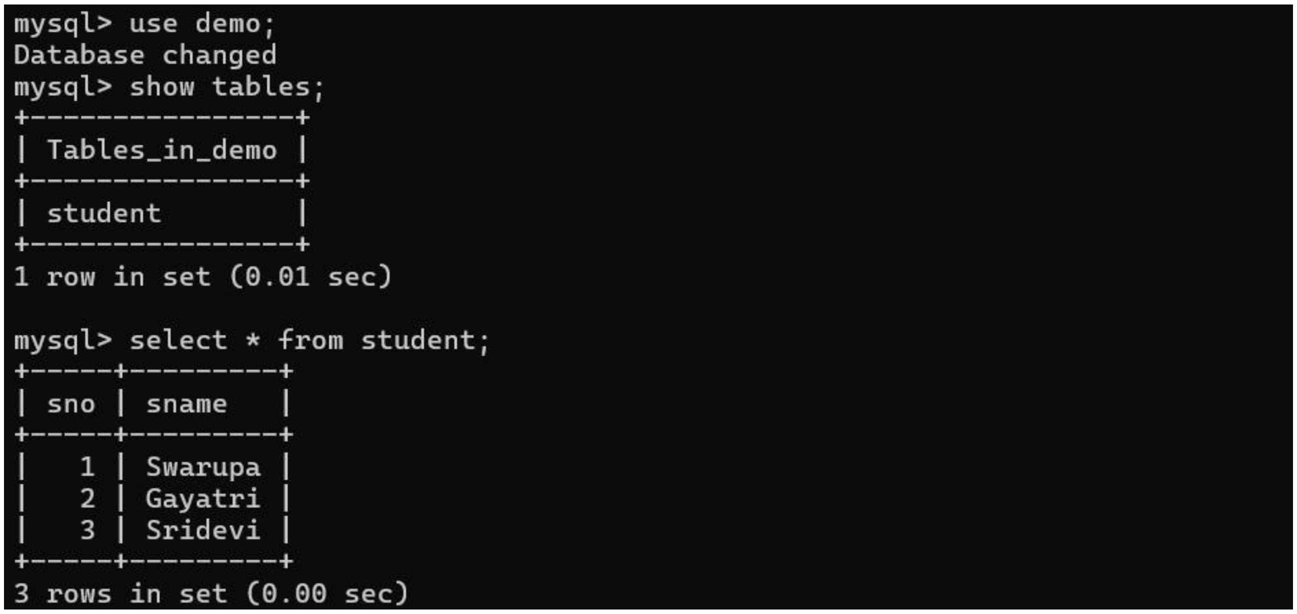
pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>3.5.5</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>U3StudentApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>U3Student</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<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-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
			<version>3.5.2</version>
		</dependency>
	</dependencies>
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Output:



```
Console
U3Student - U3StudentApplication [Spring Boot App] C:\Users\ABC\Downloads\spring\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe  (23-Sept-2025, 7:57:44 pm elapsed: 0:01:07) [pid: 9680]

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.5)

2025-09-23T19:57:45.934+05:30  INFO 9680 --- [U3Student] [           main] com.example.U3StudentApplication         : Starting U3StudentApplication using Java 21.0.7 w
2025-09-23T19:57:45.937+05:30  INFO 9680 --- [U3Student] [           main] com.example.U3StudentApplication         : No active profile set, falling back to 1 default
2025-09-23T19:57:46.821+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering stri
2025-09-23T19:57:46.823+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JDBC repositories in DE
2025-09-23T19:57:46.878+05:30  INFO 9680 --- [U3Student] [           main] .RepositoryConfigurationExtensionSupport : Spring Data JDBC - Could not safely identify stor
2025-09-23T19:57:46.879+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 45 ms
2025-09-23T19:57:46.892+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering stri
2025-09-23T19:57:46.894+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEF
2025-09-23T19:57:46.924+05:30  INFO 9680 --- [U3Student] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 22 ms
2025-09-23T19:57:47.496+05:30  INFO 9680 --- [U3Student] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 0 (http)
2025-09-23T19:57:47.515+05:30  INFO 9680 --- [U3Student] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-09-23T19:57:47.516+05:30  INFO 9680 --- [U3Student] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.44]
2025-09-23T19:57:47.570+05:30  INFO 9680 --- [U3Student] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContex
2025-09-23T19:57:47.570+05:30  INFO 9680 --- [U3Student] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization comple
2025-09-23T19:57:47.939+05:30  INFO 9680 --- [U3Student] [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name:
2025-09-23T19:57:49.521+05:30  INFO 9680 --- [U3Student] [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.6.26.Fina
2025-09-23T19:57:49.568+05:30  INFO 9680 --- [U3Student] [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-09-23T19:57:49.844+05:30  INFO 9680 --- [U3Student] [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class trans
2025-09-23T19:57:49.874+05:30  INFO 9680 --- [U3Student] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-09-23T19:57:50.177+05:30  INFO 9680 --- [U3Student] [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc
2025-09-23T19:57:50.179+05:30  INFO 9680 --- [U3Student] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-09-23T19:57:50.276+05:30  INFO 9680 --- [U3Student] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 8.0.43
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-09-23T19:57:51.002+05:30  INFO 9680 --- [U3Student] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibern
Hibernate: drop table if exists student
Hibernate: create table student (sno integer not null, sname varchar(255), primary key (sno)) engine=InnoDB
2025-09-23T19:57:51.497+05:30  INFO 9680 --- [U3Student] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persiste
2025-09-23T19:57:51.690+05:30  WARN 9680 --- [U3Student] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Th
2025-09-23T19:57:52.095+05:30  INFO 9680 --- [U3Student] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 54199 (http) with context
2025-09-23T19:57:52.102+05:30  INFO 9680 --- [U3Student] [           main] com.example.U3StudentApplication         : Started U3StudentApplication in 6.552 seconds (pr
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Students inserted!
```



```
mysql> use demo;
Database changed
mysql> show tables;
+----------------+
| Tables_in_demo |
+----------------+
| student        |
+----------------+
1 row in set (0.01 sec)

mysql> select * from student;
+-----+---------+
| sno | sname   |
+-----+---------+
|   1 | Swarupa |
|   2 | Gayatri |
|   3 | Sridevi |
+-----+---------+
3 rows in set (0.00 sec)
```

8.Pagination and Sorting in Spring Data JPA

Aim: To create the Pagination and Sorting in Spring Data JPA

Description:

Pagination and sorting are essential features when dealing with large sets of data in a database. Instead of retrieving all records at once, which can affect performance, Spring Data JPA provides built-in support to fetch data in chunks (pages) and in a specific order.
Pagination: It allows dividing the result set into smaller parts (pages). This helps reduce memory usage and speeds up response times. Spring Data JPA provides the Pageable interface and the PageRequest implementation to define the page number, size (number of records per page), and sorting details. The result is returned as a Page<T> or Slice<T> object.
Sorting: Sorting enables arranging the records in ascending or descending order based on one or more columns. Spring Data JPA supports sorting via the Sort class, which can be passed along with the Pageable object or used independently.

Book.java

```
package com.example;
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;
        private String author;
        public Book() {}
        public Book(String title, String author) {
                this.title = title;
                this.author = author;
        }
        @Override
        public String toString() {
                return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";
        }
        //getters and setters
        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.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/books")
public class BookController {
	@Autowired
	private BookRepository bookRepository;

	//Insert sample data if empty
	@GetMapping("/init")
	public String initData() {
		if (bookRepository.count()==0) {
			bookRepository.save(new Book("Spring Boot Basics","John"));
			bookRepository.save(new Book("Java Programming","Alice"));
			bookRepository.save(new Book("Hibernate in Action","Bob"));
			bookRepository.save(new Book("Microservices Guide","Carol"));
			bookRepository.save(new Book("Data Structures","Davidraj"));
		   }
		return "Sample Books added!";
	}

	  // Pagination + Sorting end point
	@GetMapping
	public Page<Book>getBooks(
			@RequestParam(defaultValue="0") int page,
			@RequestParam(defaultValue="3") int size,
			@RequestParam(defaultValue="title") String sortBy,
			@RequestParam(defaultValue="asc") String direction
			){
		Sort sort = direction.equalsIgnoreCase("asc") ?
				Sort.by(sortBy).ascending():
				Sort.by(sortBy).descending();
		 Pageable pageable = PageRequest.of(page,size,sort);
		 return bookRepository.findAll(pageable);
	}
}
```

BookRepository.java

```
package com.example;
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long>{

}
```

U3BookApplication.java

```
package com.example;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class U3BookApplication {
	public static void main(String[] args) {
		SpringApplication.run(U3BookApplication.class, args);
	}
}
```

application.properties

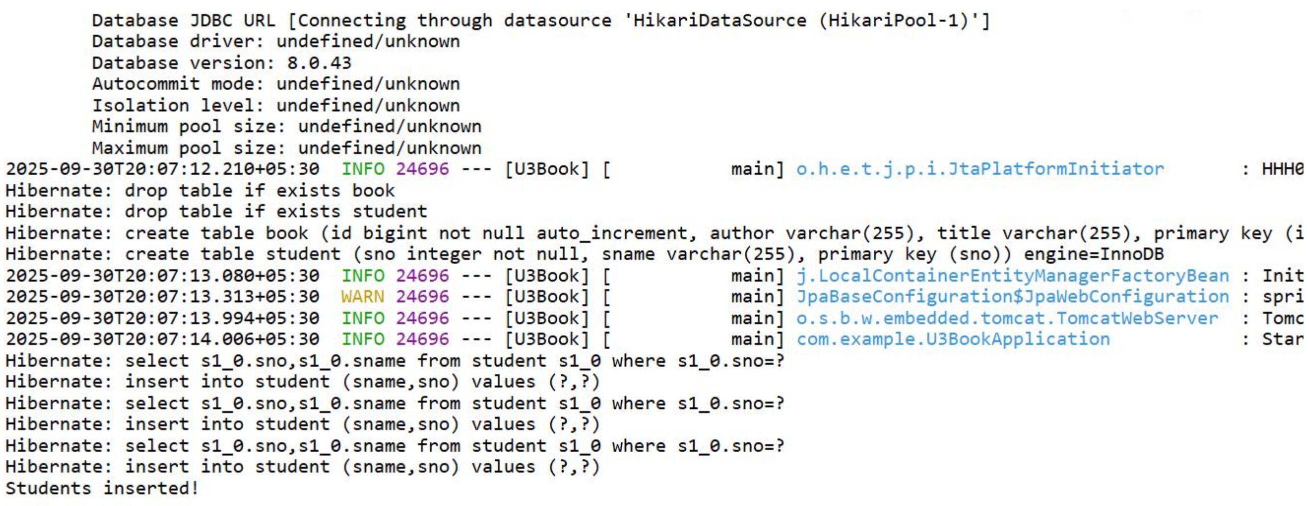
```
spring.application.name=U3Book
server.port=8748
spring.datasource.url=jdbc:mysql://localhost:3306/record
spring.datasource.username=root
spring.datasource.password=Swarupa@123
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
```

pom.xml

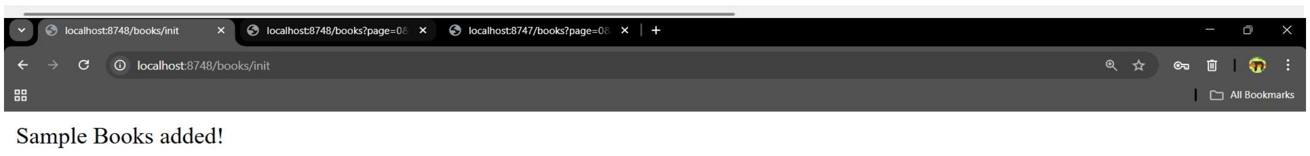
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.6</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>U3BookApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
```

```xml
	<name>U3Book</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<scope>runtime</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>com</groupId>
			<artifactId>U3StudentApplication</artifactId>
			<version>0.0.1-SNAPSHOT</version>
		</dependency>
	</dependencies>
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

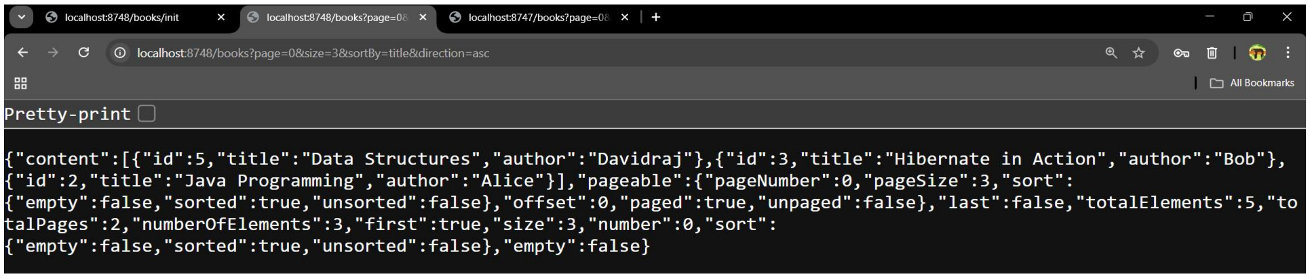
Output:



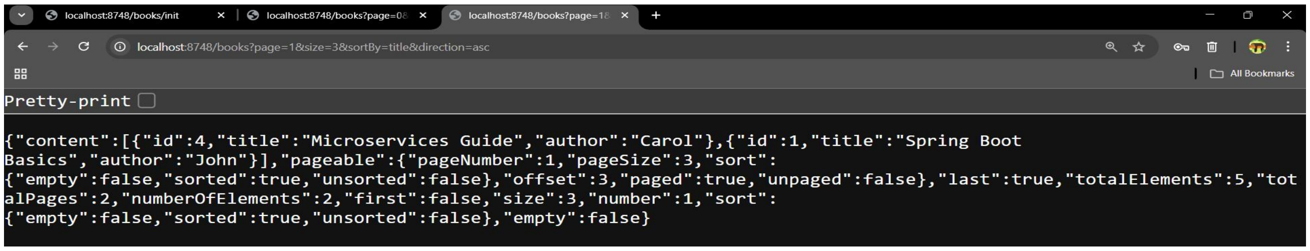
```
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.43
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-09-30T20:07:12.210+05:30  INFO 24696 --- [U3Book] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH0
Hibernate: drop table if exists book
Hibernate: drop table if exists student
Hibernate: create table book (id bigint not null auto_increment, author varchar(255), title varchar(255), primary key (i
Hibernate: create table student (sno integer not null, sname varchar(255), primary key (sno)) engine=InnoDB
2025-09-30T20:07:13.080+05:30  INFO 24696 --- [U3Book] [           main] j.LocalContainerEntityManagerFactoryBean : Init
2025-09-30T20:07:13.313+05:30  WARN 24696 --- [U3Book] [           main] JpaBaseConfiguration$JpaWebConfiguration : spri
2025-09-30T20:07:13.994+05:30  INFO 24696 --- [U3Book] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2025-09-30T20:07:14.006+05:30  INFO 24696 --- [U3Book] [           main] com.example.U3BookApplication            : Star
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Hibernate: select s1_0.sno,s1_0.sname from student s1_0 where s1_0.sno=?
Hibernate: insert into student (sname,sno) values (?,?)
Students inserted!
```


localhost:8748/books/init
localhost:8748/books?page=0&
localhost:8747/books?page=0&
localhost:8748/books/init
All Bookmarks

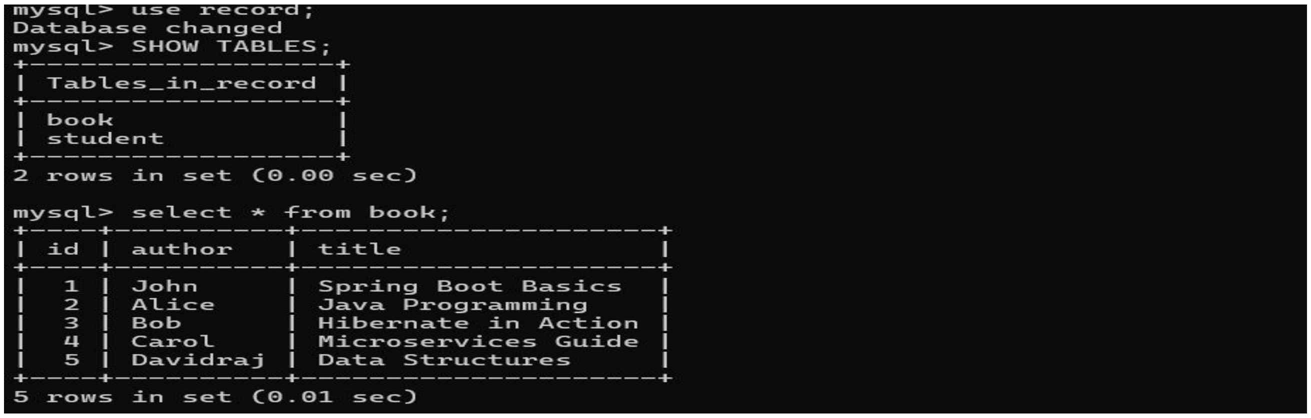
Sample Books added!


localhost:8748/books/init
localhost:8748/books?page=0&
localhost:8747/books?page=0&
localhost:8748/books?page=0&size=3&sortBy=title&direction=asc
All Bookmarks
Pretty-print

```
{"content":[{"id":5,"title":"Data Structures","author":"Davidraj"},{"id":3,"title":"Hibernate in Action","author":"Bob"},
{"id":2,"title":"Java Programming","author":"Alice"}],"pageable":{"pageNumber":0,"pageSize":3,"sort":
{"empty":false,"sorted":true,"unsorted":false},"offset":0,"paged":true,"unpaged":false},"last":false,"totalElements":5,"to
talPages":2,"numberOfElements":3,"first":true,"size":3,"number":0,"sort":
{"empty":false,"sorted":true,"unsorted":false},"empty":false}
```


localhost:8748/books/init
localhost:8748/books?page=0&
localhost:8748/books?page=1&
localhost:8748/books?page=1&size=3&sortBy=title&direction=asc
All Bookmarks
Pretty-print

```
{"content":[{"id":4,"title":"Microservices Guide","author":"Carol"},{"id":1,"title":"Spring Boot
Basics","author":"John"}],"pageable":{"pageNumber":1,"pageSize":3,"sort":
{"empty":false,"sorted":true,"unsorted":false},"offset":3,"paged":true,"unpaged":false},"last":true,"totalElements":5,"tot
alPages":2,"numberOfElements":2,"first":false,"size":3,"number":1,"sort":
{"empty":false,"sorted":true,"unsorted":false},"empty":false}
```



```
mysql> use record;
Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_record |
+------------------+
| book             |
| student          |
+------------------+
2 rows in set (0.00 sec)

mysql> select * from book;
+----+----------+---------------------+
| id | author   | title               |
+----+----------+---------------------+
|  1 | John     | Spring Boot Basics  |
|  2 | Alice    | Java Programming    |
|  3 | Bob      | Hibernate in Action |
|  4 | Carol    | Microservices Guide |
|  5 | Davidraj | Data Structures     |
+----+----------+---------------------+
5 rows in set (0.01 sec)
```

9. Implementing AOP for Logging with Spring Data JPA.

Aim: Implementing AOP for Logging with Spring Data JPA

Description:

Aspect-Oriented Programming (AOP) in Spring allows developers to separate cross-cutting concerns (like logging, security, or transaction management) from business logic. When working with Spring Data JPA, AOP is often used to implement logging so that every database-related method call can be monitored without cluttering the core code.

Aspect: A module that encapsulates cross-cutting concerns (e.g., logging).

Advice: The action taken at a particular join point (e.g., before or after a method executes).

Join Point: A point in program execution where an aspect can be applied (e.g., method execution).

Pointcut: An expression that defines which methods the advice should run on.

Product.java

```
package com.example;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Product {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private long id;
   private String name;
   private double price;

   public Product() {}
   public Product(String name, double price) {
      this.name = name;
      this.price = price;
   }

   // getters & setters
   public Long getId() { return id; }
   public void setId(Long id) { this.id = id; }

   public String getName() { return name; }
   public void setName(String name) { this.name = name; }
   public double getPrice() { return price; }
   public void setPrice(double price) { this.price = price; }
}
```

ProductController.java

```
package com.example;

import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/products")
public class ProductController {
   private final ProductService service;

   public ProductController(ProductService service) {
      this.service = service;
   }
   @PostMapping("/add")
   public Product addProduct(@RequestBody Product product) {
      return service.saveProduct(product);
   }
   @GetMapping("/all")
   public List<Product> getAllProducts() {
      return service.getAllProducts();
   }
}
```

ProductRepository.java

```
package com.example;

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

public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

ProductService.java

```
package com.example;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class ProductService {
   private final ProductRepository repo;
   public ProductService(ProductRepository repo) {
      this.repo = repo;
   }
   public Product saveProduct(Product product) {
      return repo.save(product);
   }
   public List<Product> getAllProducts() {
      return repo.findAll();
   }
}
```

LoggingAspect.java

```
package com.example;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;
@Aspect
@Component
public class LoggingAspect {
   // Logs before executing any ProductService method
   @Before("execution(* com.example.demo.ProductService.*(..))")
   public void logBefore(JoinPoint joinPoint) {
      System.out.println(">>> Entering method: " + joinPoint.getSignature().getName());
   }
}
```

U3AoploggingApplication.java

```
package com.example;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class U3AoploggingApplication {
        public static void main(String[] args) {
                SpringApplication.run(U3AoploggingApplication.class, args);
        }
         @Bean
           CommandLineRunner runner(ProductRepository repo) {
              return args -> {
                 repo.save(new Product("Laptop", 55000));
                 repo.save(new Product("Mobile", 20000));
                 repo.save(new Product("Tablet", 30000));
                 repo.save(new Product("Mouse", 35000));
              };
         }
}
```

application.properties

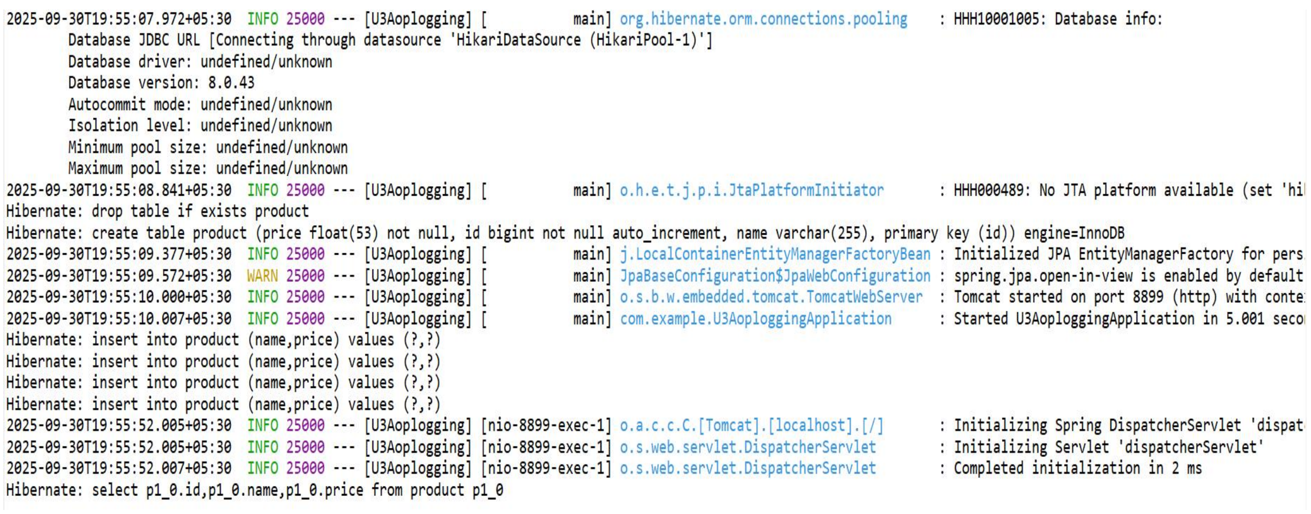
```
spring.application.name=U3Aoplogging
spring.datasource.url=jdbc:mysql://localhost:3306/mca
spring.datasource.username=root
spring.datasource.password=Swarupa@123
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
server.port=8888
```

pom.xml

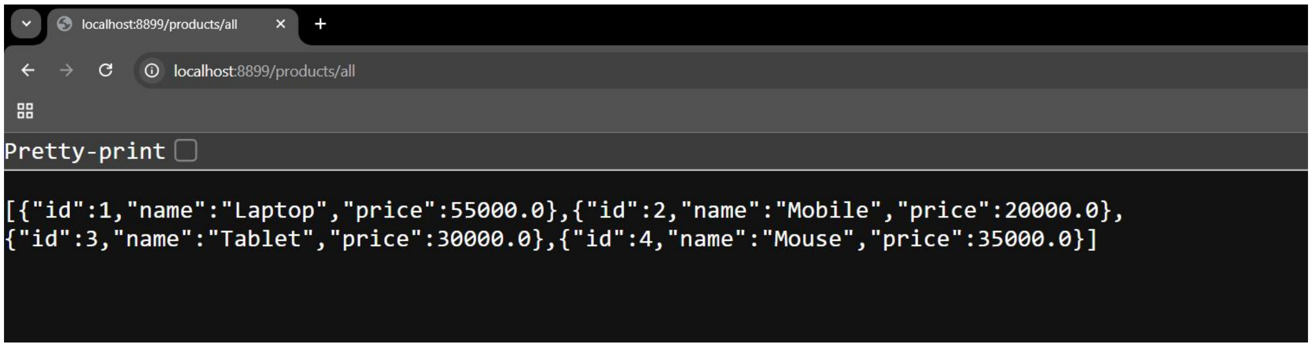
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.5.6</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>U3BookApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>U3Aoplogging</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>com.mysql</groupId>
			<artifactId>mysql-connector-j</artifactId>
			<scope>runtime</scope>
		</dependency>
```

```
			<dependency>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-starter-test</artifactId>
				<scope>test</scope>
			</dependency>
			  <dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-aop</artifactId>
	    </dependency>
	    <dependency>
		<groupId>org.projectlombok</groupId>
		<artifactId>lombok</artifactId>
		<optional>true</optional>
	    </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>
```

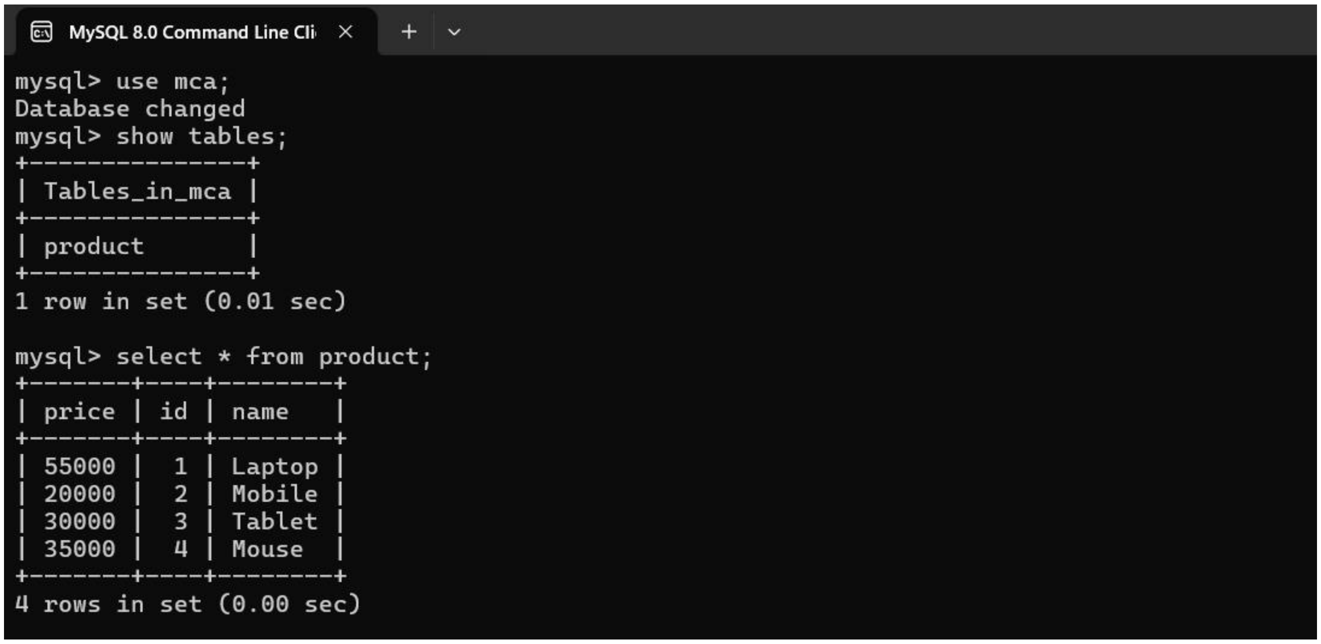
Output:



```
2025-09-30T19:55:07.972+05:30  INFO 25000 --- [U3Aoplogging] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 8.0.43
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-09-30T19:55:08.841+05:30  INFO 25000 --- [U3Aoplogging] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hi
Hibernate: drop table if exists product
Hibernate: create table product (price float(53) not null, id bigint not null auto_increment, name varchar(255), primary key (id)) engine=InnoDB
2025-09-30T19:55:09.377+05:30  INFO 25000 --- [U3Aoplogging] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for pers
2025-09-30T19:55:09.572+05:30  WARN 25000 --- [U3Aoplogging] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default
2025-09-30T19:55:10.000+05:30  INFO 25000 --- [U3Aoplogging] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8899 (http) with conte
2025-09-30T19:55:10.007+05:30  INFO 25000 --- [U3Aoplogging] [           main] com.example.U3AoploggingApplication      : Started U3AoploggingApplication in 5.001 seco
Hibernate: insert into product (name,price) values (?,?)
Hibernate: insert into product (name,price) values (?,?)
Hibernate: insert into product (name,price) values (?,?)
Hibernate: insert into product (name,price) values (?,?)
2025-09-30T19:55:52.005+05:30  INFO 25000 --- [U3Aoplogging] [nio-8899-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispat
2025-09-30T19:55:52.005+05:30  INFO 25000 --- [U3Aoplogging] [nio-8899-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-09-30T19:55:52.007+05:30  INFO 25000 --- [U3Aoplogging] [nio-8899-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
Hibernate: select p1_0.id,p1_0.name,p1_0.price from product p1_0
```



```
[{"id":1,"name":"Laptop","price":55000.0},{"id":2,"name":"Mobile","price":20000.0},
{"id":3,"name":"Tablet","price":30000.0},{"id":4,"name":"Mouse","price":35000.0}]
```



```
mysql> use mca;
Database changed
mysql> show tables;
+---------------+
| Tables_in_mca |
+---------------+
| product       |
+---------------+
1 row in set (0.01 sec)

mysql> select * from product;
+-------+----+--------+
| price | id | name   |
+-------+----+--------+
| 55000 |  1 | Laptop |
| 20000 |  2 | Mobile |
| 30000 |  3 | Tablet |
| 35000 |  4 | Mouse  |
+-------+----+--------+
4 rows in set (0.00 sec)
```