7)Aim:Basic Spring Boot Application with Spring Data JPA

Description:

In this experiment, we will create a Spring Boot application that connects to a MySQL database and uses Spring Data JPA to perform basic database operations. The application will allow inserting and retrieving student records through a RESTful API.

- **Student.java** – The entity class representing students.
- **StudentRepository.java** – The JPA repository interface for database operations.
- **StudentController.java** – REST controller for handling HTTP requests.
- **StudentApplication.java** – Main application class for bootstrapping the application.
- **application.properties** – Configuration file for database and server.
- **pom.xml** – Maven configuration file for dependencies.

Program:

1)StudentApplication.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 StudentApplication {
public static void main(String[] args) {
		SpringApplication.run(StudentApplication.class, args);
	}
	@Bean
	CommandLineRunner initDatabase(StudentRepository repo) {
		return args -> {
			repo.save(new Student(1, "Samyuktha"));
			repo.save(new Student(2, "Samyu"));
			repo.save(new Student(3, "Sam"));
			System.out.println("Students inserted!");
		};
	}

	}
```

2)Student.java

```
package com.example;
//import org.springframework.data.annotation.Id;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Student {
	@Id
	private int sno;   //primary key
	private String sname;  //Student name

	public Student() {}
	public Student(int sno, String sname) {
		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;
	}

}
```

3)application.properties

```
spring.application.name=Student
server.port= 9640
spring.datasource.url=jdbc:mysql://localhost:3306/mca
spring.datasource.username=root
spring.datasource.password= Pradeep@79979
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
```

4)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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@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();
  }

}
```

5)StudentRepository.java (Interface)

```
package com.example;

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

public interface StudentRepository extends JpaRepository<Student,Integer> {

}
```

6)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.example</groupId>
	<artifactId>StudentApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>Student</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-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-data-jpa</artifactId>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Output:

```
Problems  Javadoc  Declaration  Console  Progress
Student - StudentApplication [Spring Boot App] C:\Program Files\Java\jdk-17\bin\javaw.exe  (25-Sept-2025, 6:49:02 pm elapsed: 0:00:46) [pid: 32032]
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-09-25T18:49:24.495+05:30  INFO 32032 --- [Student] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availabl
Hibernate:
    drop table if exists student
Hibernate:
    create table student (
        sno integer not null,
        sname varchar(255),
        primary key (sno)
    ) engine=InnoDB
2025-09-25T18:49:24.746+05:30  INFO 32032 --- [Student] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactor
2025-09-25T18:49:25.067+05:30  WARN 32032 --- [Student] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled
2025-09-25T18:49:25.807+05:30  INFO 32032 --- [Student] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8989 (http)
2025-09-25T18:49:25.822+05:30  INFO 32032 --- [Student] [           main] com.example.StudentApplication           : Started StudentApplication in 8.26
Hibernate:
    select
        s1_0.sno,
        s1_0.sname
    from
        student s1_0
    where
        s1_0.sno=?
Hibernate:
    insert
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| employee           |
| information_schema |
| mcab               |
| mysql              |
| performance_schema |
| sample             |
| student            |
| sys                |
+--------------------+
8 rows in set (0.03 sec)

mysql> use mcab;
Database changed
mysql> show tables;
+----------------+
| Tables_in_mcab |
+----------------+
| book           |
| movies         |
| product        |
| student        |
+----------------+
4 rows in set (0.02 sec)

mysql> select * from student;
+-----+-----------+
| sno | sname     |
+-----+-----------+
|   1 | Samyuktha |
|   2 | Samyu     |
|   3 | Sam       |
+-----+-----------+
3 rows in set (0.01 sec)
```

8)Aim:Pagination and Sorting in Spring Data JPA

Description:

In this experiment, we will create a Spring Boot application that demonstrates how to paginate and sort database records using Spring Data JPA. We will use a Book entity with sample data, a JPA repository interface for database operations, and a REST controller to handle requests. Pagination parameters (page, size) and sorting parameters (sortBy, direction) will be passed via URL query parameters to retrieve data in a paginated and sorted manner.

Program:

1)application.properties

```
spring.application.name=Book
server.port=8821
spring.datasource.url=jdbc:mysql://localhost:3306/mcab
spring.datasource.username=root
spring.datasource.password=Samyu@19
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
server.error.whitelabel.enabled=false
```

2)BookApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BookApplication {

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

}
```

3)Book.java

```
package com.example.demo;
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(String title, String author) {
		this.title = title;
		this.author = author;
	}
	@Override
	 public String toString() {
		return "Book [id=" + id + ", title=" + title + ", author=" + author + "]";
	 }
	 //getter and setter methods
	  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;
	 }

}
```

4)BookRepository (Interface)

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

5)BookController.java

```
package com.example.demo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
@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("Micerservices 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);}

}
```

6)pom.xml

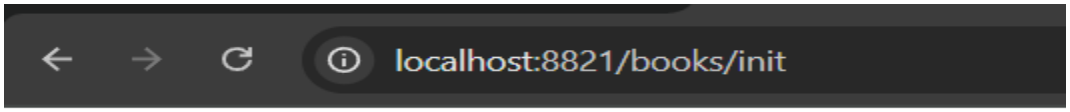
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                        https://maven.apache.org/xsd/maven-4.0.0.xsd">

   <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>PaginationandSortingApplication</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>Book</name>
   <description>Demo project for Spring Boot</description>
<licenses>
      <license/>
   </licenses>

   <developers>
      <developer/>
   </developers>

   <scm>
      <connection/>
      <developerConnection/>
      <tag/>
      <url/>
   </scm>

   <properties>
      <java.version>21</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-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-data-jpa</artifactId>
        </dependency>
    </dependencies>
<build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Output:

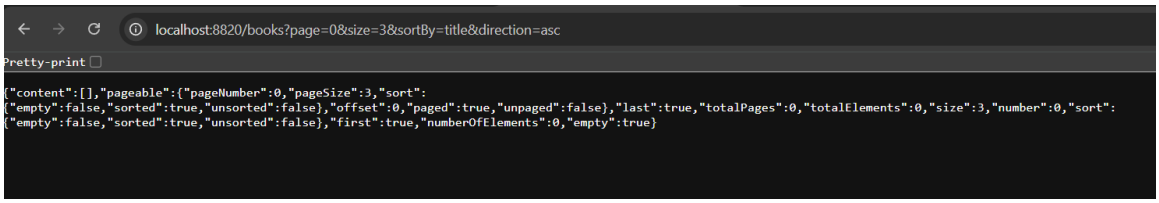
```
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 8.0.40
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-09-25T19:06:58.857+05:30  INFO 18192 --- [Book] [           main] o.h.m.i.EntityInstantiatorPojoStandard   : HHH000182: No default (no-argument) constructor for
2025-09-25T19:06:59.193+05:30  INFO 18192 --- [Book] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernat
Hibernate: drop table if exists book
Hibernate: create table book (id bigint not null auto_increment, author varchar(255), title varchar(255), primary key (id)) engine=InnoDB
2025-09-25T19:06:59.413+05:30  INFO 18192 --- [Book] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistenc
2025-09-25T19:06:59.745+05:30  WARN 18192 --- [Book] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Ther
2025-09-25T19:07:00.523+05:30  INFO 18192 --- [Book] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8820 (http) with context pat
2025-09-25T19:07:00.538+05:30  INFO 18192 --- [Book] [           main] com.example.demo.BookApplication         : Started BookApplication in 8.184 seconds (process r
```

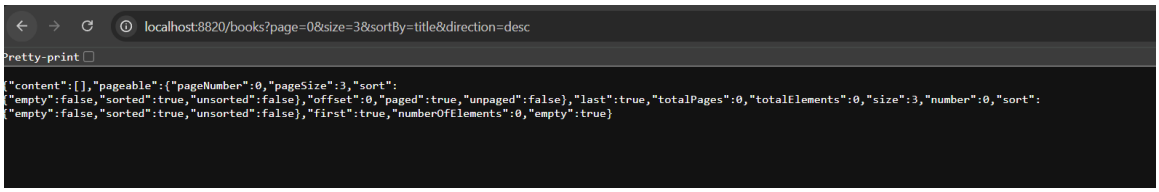
```
mysql> use mcab;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_mcab  |
+-----------------+
| book            |
| movies          |
| product         |
| student         |
+-----------------+
4 rows in set (0.04 sec)

mysql> select * from book;
Empty set (0.02 sec)
```



Sample books added!

```
mysql> use mcab;
Database changed
mysql> select * from book;
+----+----------+----------------------+
| id | author   | title                |
+----+----------+----------------------+
|  1 | John     | Spring Boot Basics   |
|  2 | Alice    | Java Programming     |
|  3 | Bob      | Hibernate in Action  |
|  4 | Carol    | Micerservices Guide  |
|  5 | Davidraj | Data Structures      |
+----+----------+----------------------+
5 rows in set (0.00 sec)
```






```
Hibernate: drop table if exists book
Hibernate: create table book (id bigint not null auto_increment, author varchar(255), title varchar(255), primary key (id)) engine=InnoDB
2025-09-25T19:06:59.413+05:30  INFO 18192 --- [Book] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistenc
2025-09-25T19:06:59.745+05:30  WARN 18192 --- [Book] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Ther
2025-09-25T19:07:00.523+05:30  INFO 18192 --- [Book] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8820 (http) with context pat
2025-09-25T19:07:00.538+05:30  INFO 18192 --- [Book] [           main] com.example.demo.BookApplication         : Started BookApplication in 8.184 seconds (process r
2025-09-25T19:13:13.965+05:30  INFO 18192 --- [Book] [nio-8820-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherSe
2025-09-25T19:13:13.965+05:30  INFO 18192 --- [Book] [nio-8820-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-09-25T19:13:13.968+05:30  INFO 18192 --- [Book] [nio-8820-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 0 ms
Hibernate: select b1_0.id,b1_0.author,b1_0.title from book b1_0 order by b1_0.title desc limit ?,?
2025-09-25T19:13:14.682+05:30  WARN 18192 --- [Book] [nio-8820-exec-1] ration$PageModule$WarningLoggingModifier : Serializing PageImpl instances as-is is not support
        For a stable JSON structure, please use Spring Data's PagedModel (globally via @EnableSpringDataWebSupport(pageSerializationMode = VIA_DTO))
        or Spring HATEOAS and Spring Data's PagedResourcesAssembler as documented in https://docs.spring.io/spring-data/commons/reference/repositories/core-extension

Hibernate: select b1_0.id,b1_0.author,b1_0.title from book b1_0 order by b1_0.title limit ?,?
Hibernate: select b1_0.id,b1_0.author,b1_0.title from book b1_0 order by b1_0.title limit ?,?
```

9)Aim:Implementing AOP for Logging with Spring Data JPA

Description:

In this experiment, we create a Spring Boot application to manage products. The application includes:

- **Entity** – Product with id, name, and price.
- **Repository** – ProductRepository for database operations.
- **Service** – ProductService to handle business logic.
- **Controller** – ProductController for REST APIs.
- **Aspect** – LoggingAspect to log method calls in ProductService.
- **Database** – H2 in-memory DB or MySQL.
- **Dependency Management** – Managed via Maven (pom.xml).

This demonstrates the use of **Spring Data JPA**, **Spring AOP**, and **RESTful API development**.

Program:

1)ProductRepository.java (Interface)

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

2)ProductService.java

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

3)application.properties

```
spring.application.name=product

spring.datasource.url=jdbc:mysql://localhost:3306/mcab

spring.datasource.username=root

spring.datasource.password=Samyu@19

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

server.port=8902
```

4)ProductController.java

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

5)product.java

```
package com.example.demo;
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; }
}
```

6)LoggingAspect.java

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

7)main.java

```
package com.example.demo;
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 ProductApplication {
	public static void main(String[] args) {
		SpringApplication.run(ProductApplication.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));
	      };
	 }
	 }
```

8)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>3.5.4</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com</groupId>
	<artifactId>productApplication</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>product</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>21</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-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>
                    <!-- Spring AOP -->
        <dependency>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-aop</artifactId>
        </dependency>
     <!-- Lombok (optional, just to reduce boilerplate) -->
        <dependency>
          <groupId>org.projectlombok</groupId>
          <artifactId>lombok</artifactId>
          <optional>true</optional>
        </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
         <!-- H2 Database (in-memory, no need for MySQL setup) -->
        <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:

```
        Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
        Database driver: undefined/unknown
        Database version: 8.0.40
        Autocommit mode: undefined/unknown
        Isolation level: undefined/unknown
        Minimum pool size: undefined/unknown
        Maximum pool size: undefined/unknown
2025-09-25T15:03:58.723+05:30  INFO 20204 --- [product] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hiber
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
2025-09-25T15:03:58.844+05:30  INFO 20204 --- [product] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persist
2025-09-25T15:03:59.031+05:30  WARN 20204 --- [product] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. T
2025-09-25T15:03:59.381+05:30  INFO 20204 --- [product] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8902 (http) with context
2025-09-25T15:03:59.387+05:30  INFO 20204 --- [product] [           main] com.example.demo.ProductApplication      : Started ProductApplication in 3.977 seconds (pro
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 (?,?)
2025-09-25T15:07:47.076+05:30  INFO 20204 --- [product] [nio-8902-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatche
2025-09-25T15:07:47.077+05:30  INFO 20204 --- [product] [nio-8902-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-09-25T15:07:47.080+05:30  INFO 20204 --- [product] [nio-8902-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 3 ms
>>> Entering method: getAllProducts
Hibernate: select p1_0.id,p1_0.name,p1_0.price from product p1_0
2025-09-25T17:51:40.011+05:30  WARN 20204 --- [product] [l-1:housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Thread starvation or clock leap d
2025-09-25T18:27:33.161+05:30  WARN 20204 --- [product] [l-1:housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Thread starvation or clock leap d
```

```
mysql> use mcab;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_mcab  |
+-----------------+
| book            |
| movies          |
| product         |
+-----------------+
3 rows in set (0.03 sec)

mysql>
mysql> select * from product;
+-------+----+---------+
| price | id | name    |
+-------+----+---------+
| 55000 |  1 | Laptop  |
| 20000 |  2 | Mobile  |
| 30000 |  3 | Tablet  |
| 35000 |  4 | Mouse   |
+-------+----+---------+
4 rows in set (0.04 sec)
```