Spring Boot CRUD Application with MySQL

Project Documentation

2026

Contents

1 Introduction

This document outlines the implementation of a RESTful API using Spring Boot and MySQL. The application manages a "Product" inventory, showcasing Create, Read, Update, and Delete (CRUD) operations.

2 Prerequisites

- Java Development Kit (JDK) 17 or higher.
- VS Code with Spring Boot Extension Pack.
- MySQL Server.
- Maven for dependency management.

3 Database Configuration

Create a database named `springboot_crud` in MySQL. The configuration in `src/main/resources/applica` is as follows:

```
spring.datasource.url=jdbc:mysql://localhost:3306/springboot_crud
spring.datasource.username=root
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Listing 1: application.properties

4 Project Components

4.1 Entity Layer

The `Product` class serves as the JPA entity mapped to the MySQL table.

```
package com.example.demo.model;

import jakarta.persistence.*;
import lombok.Data;

@Entity
@Table(name = "products")
@Data
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private double price;
```

```
    private int quantity;
}
```

Listing 2: Product.java

4.2 Repository Layer

The repository interface extends `JpaRepository` to provide standard database operations.

```
package com.example.demo.repository;

import com.example.demo.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;

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

Listing 3: ProductRepository.java

4.3 Service Layer

The service layer contains the business logic.

```
@Service
public class ProductService {
    @Autowired
    private ProductRepository repository;

    public Product saveProduct(Product product) {
        return repository.save(product);
    }

    public List<Product> getProducts() {
        return repository.findAll();
    }

    public String deleteProduct(Long id) {
        repository.deleteById(id);
        return "Product removed " + id;
    }
}
```

Listing 4: ProductService.java

4.4 Controller Layer

Exposes REST endpoints for external clients.

```
@RestController
@RequestMapping("/api/products")
public class ProductController {
    @Autowired
```

```
    private ProductService service;

    @PostMapping
    public Product addProduct(@RequestBody Product product) {
        return service.saveProduct(product);
    }

    @GetMapping
    public List<Product> findAllProducts() {
        return service.getProducts();
    }
}
```

Listing 5: ProductController.java

5 API Endpoint Summary

Method	Endpoint	Action
POST	/api/products	Create Product
GET	/api/products	Get All Products
GET	/api/products/{id}	Get Product by ID
PUT	/api/products	Update Product
DELETE	/api/products/{id}	Delete Product