Spring Data JPA - Quick Example

### **Step 1: Create a Spring Boot Project**

1. Go to [start.spring.io](https://start.spring.io/)
2. Select:
   * Project: Maven
   * Language: Java
   * Spring Boot: 3.2.x (Java 21 compatible)
3. Add dependencies:
   * Spring Data JPA
   * H2 Database (for in-memory testing)
4. Generate and download the project

### **Step 2: Configure application.properties**

Add to *src/main/resources/application.properties*:

# H2 Database

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# H2 Console (Access at http://localhost:8080/h2-console)

spring.h2.console.enabled=true

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

# JPA

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

### **Step 3: Create Entity Class**

// src/main/java/com/example/demo/entity/Product.java

package com.example.demo.entity;

import jakarta.persistence.\*;

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private double price;

// Getters and setters

// Constructor (no-args and all-args)

}

### **Step 4: Create Repository Interface**

// src/main/java/com/example/demo/repository/ProductRepository.java

package com.example.demo.repository;

import com.example.demo.entity.Product;

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

public interface ProductRepository extends JpaRepository<Product, Long> {

// Custom query method

List<Product> findByPriceLessThan(double price);

}

### **Step 5: Create a Service Class**

// src/main/java/com/example/demo/service/ProductService.java

package com.example.demo.service;

import com.example.demo.entity.Product;

import com.example.demo.repository.ProductRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class ProductService {

private final ProductRepository productRepository;

public ProductService(ProductRepository productRepository) {

this.productRepository = productRepository;

}

public Product saveProduct(Product product) {

return productRepository.save(product);

}

public List<Product> getAffordableProducts(double maxPrice) {

return productRepository.findByPriceLessThan(maxPrice);

}

}

### **Step 6: Create a REST Controller**

// src/main/java/com/example/demo/controller/ProductController.java

package com.example.demo.controller;

import com.example.demo.entity.Product;

import com.example.demo.service.ProductService;

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

import java.util.List;

@RestController

@RequestMapping("/api/products")

public class ProductController {

private final ProductService productService;

public ProductController(ProductService productService) {

this.productService = productService;

}

@PostMapping

public Product addProduct(@RequestBody Product product) {

return productService.saveProduct(product);

}

@GetMapping("/affordable")

public List<Product> getAffordableProducts(@RequestParam double maxPrice) {

return productService.getAffordableProducts(maxPrice);

}

}

### **Step 7: Test the Application**

1. Run the Spring Boot application (DemoApplication.java)
2. Use Postman or curl to test.