**Spring Boot CRUD Application**

A simple Spring Boot application that demonstrates CRUD (Create, Read, Update, Delete) operations for **Category** and **Product** entities using Spring Data JPA and MySQL.

**Table of Contents**

* [Features](#features)
* [Project Structure](#project-structure)
* [Requirements](#requirements)
* [Setup Instructions](#setup-instructions)
* [API Endpoints](#api-endpoints)
* [Database Configuration](#database-configuration)
* [Running the Application](#running-the-application)
* [Testing the APIs](#testing-the-apis)

**Features**

* **CRUD operations** for Category and Product entities.
* **One-to-Many relationship** between Category and Product.
* **Pagination support** for fetching records.
* Built using **Spring Boot**, **Spring Data JPA**, **MySQL**, and **Hibernate**.
* RESTful APIs implemented with **Spring Web**.

**Project Structure**

springbootcrud/

├── src/

│ └── main/

│ └── java/

│ └── com/

│ └── example/

│ └── springbootcrud/

│ ├── SpringBootCrudApplication.java

│ ├── model/

│ │ ├── Category.java

│ │ └── Product.java

│ ├── repository/

│ │ ├── CategoryRepository.java

│ │ └── ProductRepository.java

│ └── controller/

│ ├── CategoryController.java

│ └── ProductController.java

└── resources/

└── application.properties

**Requirements**

* **Java**: 11 or 17 (recommended)
* **Maven**: To build the project.
* **MySQL**: For database operations.
* **Spring Tool Suite 4** (or any IDE with Spring Boot support).

**Setup Instructions**

**1. Clone the Repository**

git clone https://github.com/yourusername/springbootcrud.git

cd springbootcrud

**2. Configure the Database**

In the src/main/resources/application.properties file, configure your MySQL database:

# Database configuration

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

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Make sure to replace your\_database\_name and your\_password with your actual MySQL credentials.

**3. Build the Project**

Run the following Maven command to build the project:

mvn clean install

**4. Run the Application**

You can run the application using the following command:

mvn spring-boot:run

Alternatively, in Spring Tool Suite 4, right-click the SpringBootCrudApplication.java file and select Run As > Spring Boot App.

The application will start at http://localhost:8080.

**API Endpoints**

**Category APIs**

| **Method** | **Endpoint** | **Description** | **Payload (JSON)** |
| --- | --- | --- | --- |
| GET | /api/categories | Get all categories (paginated) | - |
| GET | /api/categories/{id} | Get category by ID | - |
| POST | /api/categories | Create a new category | { "name": "category\_name"} |
| PUT | /api/categories/{id} | Update an existing category | { "name": "updated\_name"} |
| DELETE | /api/categories/{id} | Delete category by ID | - |

**Product APIs**

| **Method** | **Endpoint** | **Description** | **Payload (JSON)** |
| --- | --- | --- | --- |
| GET | /api/products | Get all products (paginated) | - |
| GET | /api/products/{id} | Get product by ID | - |
| POST | /api/products | Create a new product | { "name": "product\_name", "price": 100.00, "category": { "id": 1 } } |
| PUT | /api/products/{id} | Update an existing product | { "name": "updated\_name", "price": 150.00, "category": { "id": 2 } } |
| DELETE | /api/products/{id} | Delete product by ID | - |

**Pagination**

To enable pagination for both **Categories** and **Products**, the APIs accept the following query parameter:

* page: The page number (starts from 0).

Example: GET /api/categories?page=0

**Database Configuration**

The application uses **MySQL** as the database. You can modify the application.properties file to

# MySQL Database Configuration

spring.datasource.url=jdbc:mysql://localhost:3306/your\_database

spring.datasource.username=root

spring.datasource.password=root\_password

# JPA & Hibernate Configuration

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

If you prefer to use a different database (e.g., PostgreSQL, H2), update the driver and connection settings accordingly.

**Running the Application**

1. Ensure that your MySQL database is running and accessible.
2. Run the application using the instructions mentioned in the [Setup Instructions](#setup-instructions) section.
3. The application will be accessible at http://localhost:8080.

**Testing the APIs**

You can use **Postman** or **curl** to test the API endpoints.

**Example: Create a Category**

POST http://localhost:8080/api/categories \

"Content-Type: application/json" \

'{"name": "Electronics"}'

**Example: Create a Product**

X POST http://localhost:8080/api/products \

"Content-Type: application/json" \

'{"name": "Laptop", "price": 999.99, "category": {"id": 1}}'

**Example: Get All Products (Page 0)**

GET http://localhost:8080/api/products?page=0