ANKITA ANAND

OHR ID: 850058506

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>2.6.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<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> <version>2.2.2.RELEASE</version> </dependency>

/\*<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jdbc</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>\*/

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

APPLICATION.PROPERTIES:

spring.datasource.url= jdbc:mysql://localhost:3306/testdb?useSSL=false

spring.datasource.username= root

spring.datasource.password= 123456

spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto= update

TUTORIAL.JAVA

package com.org.asssessment.model;

import javax.persistence.\*;

@Entity

@Table(name = "tutorials")

public class Tutorial {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id;

@Column(name = "title")

private String title;

@Column(name = "description")

private String description;

@Column(name = "published")

private boolean published;

public Tutorial() {

}

public Tutorial(String title, String description, boolean published) {

this.title = title;

this.description = description;

this.published = published;

}

public long getId() {

return id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public boolean isPublished() {

return published;

}

public void setPublished(boolean isPublished) {

this.published = isPublished;

}

@Override

public String toString() {

return "Tutorial [id=" + id + ", title=" + title + ", desc=" + description + ", published=" + published + "]";

}

}

TutorialRepository.Java Interface:

package com.org.assessment.repository;

import java.util.List;

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

import com.designmycodes.spring.datajpa.model.Tutorial;

public interface TutorialRepository extends JpaRepository<Tutorial, Long> {

List<Tutorial> findByPublished(boolean published);

List<Tutorial> findByTitleContaining(String title);

}

TutorialController.java(Controller Apis for findby id, post, delete& put):

Package com.org.assessment.controller;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.designmycodes.spring.datajpa.model.Tutorial;

import com.designmycodes.spring.datajpa.repository.TutorialRepository;

@CrossOrigin(origins = "http://localhost:8081")

@RestController

@RequestMapping("/api")

public class TutorialController {

@Autowired

TutorialRepository tutorialRepository;

@GetMapping("/tutorials")

public ResponseEntity<List<Tutorial>> getAllTutorials(@RequestParam(required = false) String title) {

try {

List<Tutorial> tutorials = new ArrayList<Tutorial>();

if (title == null)

tutorialRepository.findAll().forEach(tutorials::add);

else

tutorialRepository.findByTitleContaining(title).forEach(tutorials::add);

if (tutorials.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

return new ResponseEntity<>(tutorials, HttpStatus.OK);

} catch (Exception e) {

return new ResponseEntity<>(null, HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

@GetMapping("/tutorials/{id}")

public ResponseEntity<Tutorial> getTutorialById(@PathVariable("id") long id) {

Optional<Tutorial> tutorialData = tutorialRepository.findById(id);

if (tutorialData.isPresent()) {

return new ResponseEntity<>(tutorialData.get(), HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@PostMapping("/tutorials")

public ResponseEntity<Tutorial> createTutorial(@RequestBody Tutorial tutorial) {

try {

Tutorial \_tutorial = tutorialRepository

.save(new Tutorial(tutorial.getTitle(), tutorial.getDescription(), false));

return new ResponseEntity<>(\_tutorial, HttpStatus.CREATED);

} catch (Exception e) {

return new ResponseEntity<>(null, HttpStatus.EXPECTATION\_FAILED);

}

}

@PutMapping("/tutorials/{id}")

public ResponseEntity<Tutorial> updateTutorial(@PathVariable("id") long id, @RequestBody Tutorial tutorial) {

Optional<Tutorial> tutorialData = tutorialRepository.findById(id);

if (tutorialData.isPresent()) {

Tutorial \_tutorial = tutorialData.get();

\_tutorial.setTitle(tutorial.getTitle());

\_tutorial.setDescription(tutorial.getDescription());

\_tutorial.setPublished(tutorial.isPublished());

return new ResponseEntity<>(tutorialRepository.save(\_tutorial), HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@DeleteMapping("/tutorials/{id}")

public ResponseEntity<HttpStatus> deleteTutorial(@PathVariable("id") long id) {

try {

tutorialRepository.deleteById(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} catch (Exception e) {

return new ResponseEntity<>(HttpStatus.EXPECTATION\_FAILED);

}

}

@DeleteMapping("/tutorials")

public ResponseEntity<HttpStatus> deleteAllTutorials() {

try {

tutorialRepository.deleteAll();

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} catch (Exception e) {

return new ResponseEntity<>(HttpStatus.EXPECTATION\_FAILED);

}

}

@GetMapping("/tutorials/published")

public ResponseEntity<List<Tutorial>> findByPublished() {

try {

List<Tutorial> tutorials = tutorialRepository.findByPublished(true);

if (tutorials.isEmpty()) {

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

return new ResponseEntity<>(tutorials, HttpStatus.OK);

} catch (Exception e) {

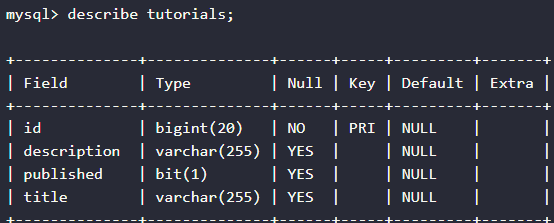
return new ResponseEntity<>(HttpStatus.EXPECTATION\_FAILED);

}

}

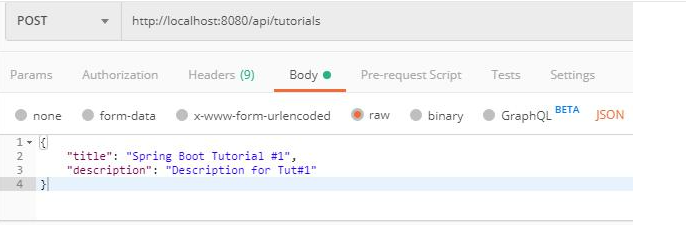
}

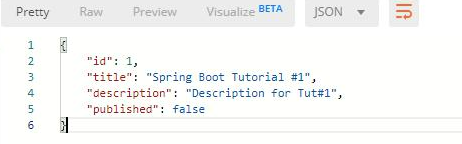
OUTPUT:

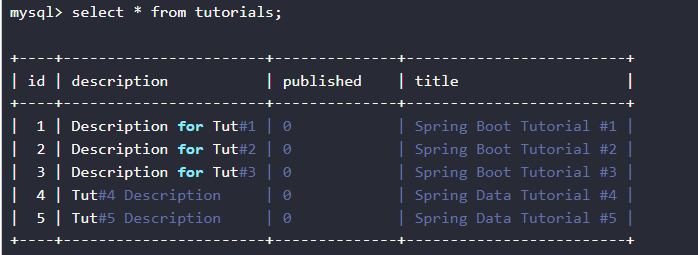


* Tutorials table in database on running the spring boot application

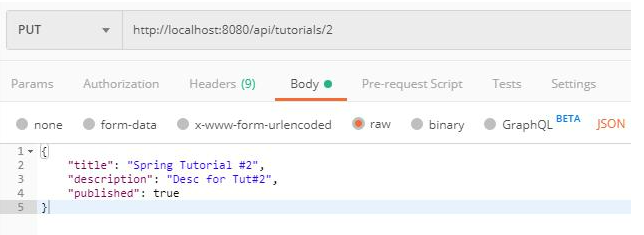
CREATING SOME TUTORIALS:



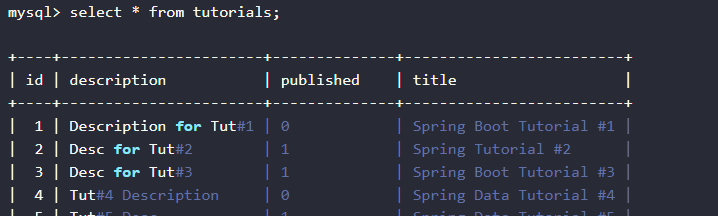




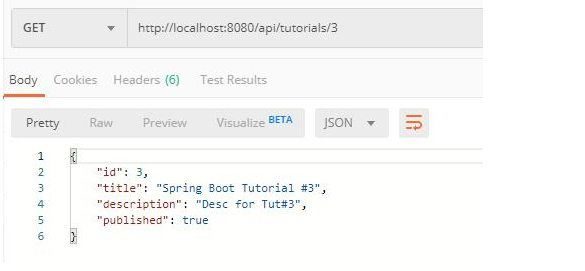
UPDATE SOME TUTORIALS:







FIND A TUTORIAL BY ID:



DELETE TUTORIAL BY ID:

