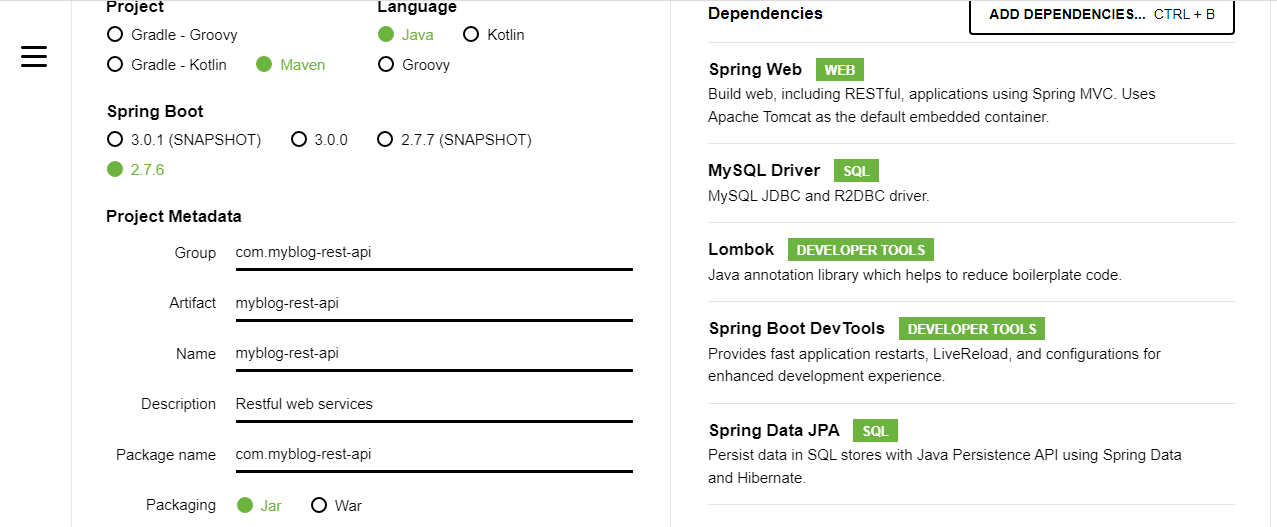
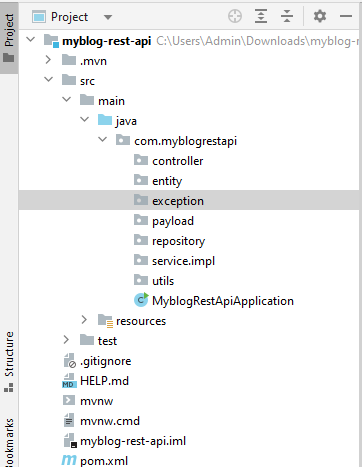
**Developing restful web services in spring boot**

1. **Create Spring boot project with following dependencies:**



1. Create Following Project Structure in IntelliJ Idea



Step 3: Create POST Entity Class

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.\*;

@Data

@AllArgsConstructor

@NoArgsConstructor

@Entity

@Table

(

name = "posts", uniqueConstraints = {@UniqueConstraint(columnNames = {"title"})}

)

public class Post {

@Id

@GeneratedValue( strategy = GenerationType.IDENTITY) )

private Long id;

@Column(name = "title", nullable = false)

private String title;

@Column(name = "description", nullable = false)

private String description;

@Column(name = "content", nullable = false)

private String content;

}

Step 3: Update application.properties file

spring.datasource.url = jdbc:mysql://localhost:3306/myblog?useSSL=false&serverTimezone=UTC

spring.datasource.username = root

spring.datasource.password = root

# hibernate properties

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

# Hibernate ddl auto

spring.jpa.hibernate.ddl-auto = update

Step 4: Create Post Repository Layer:

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

public interface PostRepository extends JpaRepository<Post, Long> {

}

Step 5: Create Payload PostDto class

import lombok.Data;

@Data

public class PostDto {

private long id;

private String title;

private String description;

private String content;

}

Step 6: Create PostService Interface

import java.util.List;

public interface PostService {

PostDto createPost(PostDto postDto);

}

Step 7: Create PostServiceImpl class

@Service

public class PostServiceImpl implements PostService {

private PostRepository postRepository;

public PostServiceImpl(PostRepository postRepository) {

this.postRepository = postRepository;

}

@Override

public PostDto createPost(PostDto postDto) {

// convert DTO to entity

Post post = mapToEntity(postDto);

Post newPost = postRepository.save(post);

// convert entity to DTO

PostDto postResponse = mapToDTO(newPost);

return postResponse;

}

// convert Entity into DTO

private PostDto mapToDTO(Post post){

PostDto postDto = new PostDto();

postDto.setId(post.getId());

postDto.setTitle(post.getTitle());

postDto.setDescription(post.getDescription());

postDto.setContent(post.getContent());

return postDto;

}

// convert DTO to entity

private Post mapToEntity(PostDto postDto){

Post post = new Post();

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

return post;

}

}

Step 8: Create PostController Class:

@RestController

@RequestMapping("/api/posts")

public class PostController {

private PostService postService;

public PostController(PostService postService) {

this.postService = postService;

}

// create blog post rest api

@PostMapping

public ResponseEntity<PostDto> createPost(@RequestBody PostDto postDto){

return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);

}

}

Step 9: Create Exception class

import org.springframework.http.HttpStatus;

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

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public class ResourceNotFoundException extends RuntimeException{

private String resourceName;

private String fieldName;

private long fieldValue;

public ResourceNotFoundException(String resourceName, String fieldName, long fieldValue) {

super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue)); // Post not found with id : 1

this.resourceName = resourceName;

this.fieldName = fieldName;

this.fieldValue = fieldValue;

}

public String getResourceName() {

return resourceName;

}

public String getFieldName() {

return fieldName;

}

public long getFieldValue() {

return fieldValue;

}

}

Step 10: Create GetMapping in controller layer:

import java.util.List;

@RestController

@RequestMapping("/api/posts")

public class PostController {

private PostService postService;

public PostController(PostService postService) {

this.postService = postService;

}

// create blog post rest api

@PostMapping

public ResponseEntity<PostDto> createPost(@RequestBody PostDto postDto){

return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);

}

// get all posts rest api

@GetMapping

public List<PostDto> getAllPosts(){

return postService.getAllPosts();

}

}

Step 11: Update PostService interface:

import com.springboot.blog.payload.PostDto;

import java.util.List;

public interface PostService {

PostDto createPost(PostDto postDto);

List<PostDto> getAllPosts();

}

Step 12: Update PostServiceImpl class:

import com.springboot.blog.entity.Post;

import com.springboot.blog.exception.ResourceNotFoundException;

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.repository.PostRepository;

import com.springboot.blog.service.PostService;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.stream.Collectors;

@Service

public class PostServiceImpl implements PostService {

private PostRepository postRepository;

public PostServiceImpl(PostRepository postRepository) {

this.postRepository = postRepository;

}

@Override

public PostDto createPost(PostDto postDto) {

// convert DTO to entity

Post post = mapToEntity(postDto);

Post newPost = postRepository.save(post);

// convert entity to DTO

PostDto postResponse = mapToDTO(newPost);

return postResponse;

}

@Override

public List<PostDto> getAllPosts() {

List<Post> posts = postRepository.findAll();

return posts.stream().map(post -> mapToDTO(post)).collect(Collectors.toList());

}

// convert Entity into DTO

private PostDto mapToDTO(Post post){

PostDto postDto = new PostDto();

postDto.setId(post.getId());

postDto.setTitle(post.getTitle());

postDto.setDescription(post.getDescription());

postDto.setContent(post.getContent());

return postDto;

}

// convert DTO to entity

private Post mapToEntity(PostDto postDto){

Post post = new Post();

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

return post;

}

}

Step 13: Create DeleteMapping By Id:

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.service.PostService;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/posts")

public class PostController {

private PostService postService;

public PostController(PostService postService) {

this.postService = postService;

}

// create blog post rest api

@PostMapping

public ResponseEntity<PostDto> createPost(@RequestBody PostDto postDto){

return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);

}

// get all posts rest api

@GetMapping

public List<PostDto> getAllPosts(){

return postService.getAllPosts();

}

// get post by id

@GetMapping("/{id}")

public ResponseEntity<PostDto> getPostById(@PathVariable(name = "id") long id){

return ResponseEntity.ok(postService.getPostById(id));

}

}

Step 14: Update PostServiceImpl interface:

import com.springboot.blog.payload.PostDto;

import java.util.List;

public interface PostService {

PostDto createPost(PostDto postDto);

List<PostDto> getAllPosts();

PostDto getPostById(long id);

}

Step 15: Update PostServiceImpl class

import com.springboot.blog.entity.Post;

import com.springboot.blog.exception.ResourceNotFoundException;

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.repository.PostRepository;

import com.springboot.blog.service.PostService;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.stream.Collectors;

@Service

public class PostServiceImpl implements PostService {

private PostRepository postRepository;

public PostServiceImpl(PostRepository postRepository) {

this.postRepository = postRepository;

}

@Override

public PostDto createPost(PostDto postDto) {

// convert DTO to entity

Post post = mapToEntity(postDto);

Post newPost = postRepository.save(post);

// convert entity to DTO

PostDto postResponse = mapToDTO(newPost);

return postResponse;

}

@Override

public List<PostDto> getAllPosts() {

List<Post> posts = postRepository.findAll();

return posts.stream().map(post -> mapToDTO(post)).collect(Collectors.toList());

}

@Override

public PostDto getPostById(long id) {

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

return mapToDTO(post);

}

// convert Entity into DTO

private PostDto mapToDTO(Post post){

PostDto postDto = new PostDto();

postDto.setId(post.getId());

postDto.setTitle(post.getTitle());

postDto.setDescription(post.getDescription());

postDto.setContent(post.getContent());

return postDto;

}

// convert DTO to entity

private Post mapToEntity(PostDto postDto){

Post post = new Post();

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

return post;

}

}

Step 16: Create UpdateMapping Controller

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.service.PostService;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/posts")

public class PostController {

private PostService postService;

public PostController(PostService postService) {

this.postService = postService;

}

// create blog post rest api

@PostMapping

public ResponseEntity<PostDto> createPost(@RequestBody PostDto postDto){

return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);

}

// get all posts rest api

@GetMapping

public List<PostDto> getAllPosts(){

return postService.getAllPosts();

}

// get post by id

@GetMapping("/{id}")

public ResponseEntity<PostDto> getPostById(@PathVariable(name = "id") long id){

return ResponseEntity.ok(postService.getPostById(id));

}

// update post by id rest api

@PutMapping("/{id}")

public ResponseEntity<PostDto> updatePost(@RequestBody PostDto postDto, @PathVariable(name = "id") long id){

PostDto postResponse = postService.updatePost(postDto, id);

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

}

}

Step 17: Update PostService Interface:

import com.springboot.blog.payload.PostDto;

import java.util.List;

public interface PostService {

PostDto createPost(PostDto postDto);

List<PostDto> getAllPosts();

PostDto getPostById(long id);

PostDto updatePost(PostDto postDto, long id);

}

Step 18: Update PostServiceImpl class:

import com.springboot.blog.entity.Post;

import com.springboot.blog.exception.ResourceNotFoundException;

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.repository.PostRepository;

import com.springboot.blog.service.PostService;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.stream.Collectors;

@Service

public class PostServiceImpl implements PostService {

private PostRepository postRepository;

public PostServiceImpl(PostRepository postRepository) {

this.postRepository = postRepository;

}

@Override

public PostDto createPost(PostDto postDto) {

// convert DTO to entity

Post post = mapToEntity(postDto);

Post newPost = postRepository.save(post);

// convert entity to DTO

PostDto postResponse = mapToDTO(newPost);

return postResponse;

}

@Override

public List<PostDto> getAllPosts() {

List<Post> posts = postRepository.findAll();

return posts.stream().map(post -> mapToDTO(post)).collect(Collectors.toList());

}

@Override

public PostDto getPostById(long id) {

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

return mapToDTO(post);

}

@Override

public PostDto updatePost(PostDto postDto, long id) {

// get post by id from the database

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

Post updatedPost = postRepository.save(post);

return mapToDTO(updatedPost);

}

// convert Entity into DTO

private PostDto mapToDTO(Post post){

PostDto postDto = new PostDto();

postDto.setId(post.getId());

postDto.setTitle(post.getTitle());

postDto.setDescription(post.getDescription());

postDto.setContent(post.getContent());

return postDto;

}

// convert DTO to entity

private Post mapToEntity(PostDto postDto){

Post post = new Post();

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

return post;

}

}

Step 19: Create DeleteMapping controller:

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.service.PostService;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/posts")

public class PostController {

private PostService postService;

public PostController(PostService postService) {

this.postService = postService;

}

// create blog post rest api

@PostMapping

public ResponseEntity<PostDto> createPost(@RequestBody PostDto postDto){

return new ResponseEntity<>(postService.createPost(postDto), HttpStatus.CREATED);

}

// get all posts rest api

@GetMapping

public List<PostDto> getAllPosts(){

return postService.getAllPosts();

}

// get post by id

@GetMapping("/{id}")

public ResponseEntity<PostDto> getPostById(@PathVariable(name = "id") long id){

return ResponseEntity.ok(postService.getPostById(id));

}

// update post by id rest api

@PutMapping("/{id}")

public ResponseEntity<PostDto> updatePost(@RequestBody PostDto postDto, @PathVariable(name = "id") long id){

PostDto postResponse = postService.updatePost(postDto, id);

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

}

// delete post rest api

@DeleteMapping("/{id}")

public ResponseEntity<String> deletePost(@PathVariable(name = "id") long id){

postService.deletePostById(id);

return new ResponseEntity<>("Post entity deleted successfully.", HttpStatus.OK);

}

}

Step 20: Update PostService Interface:

import com.springboot.blog.payload.PostDto;

import java.util.List;

public interface PostService {

PostDto createPost(PostDto postDto);

List<PostDto> getAllPosts();

PostDto getPostById(long id);

PostDto updatePost(PostDto postDto, long id);

void deletePostById(long id);

}

Step 21: Create PostServiceImpl class:

import com.springboot.blog.entity.Post;

import com.springboot.blog.exception.ResourceNotFoundException;

import com.springboot.blog.payload.PostDto;

import com.springboot.blog.repository.PostRepository;

import com.springboot.blog.service.PostService;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.stream.Collectors;

@Service

public class PostServiceImpl implements PostService {

private PostRepository postRepository;

public PostServiceImpl(PostRepository postRepository) {

this.postRepository = postRepository;

}

@Override

public PostDto createPost(PostDto postDto) {

// convert DTO to entity

Post post = mapToEntity(postDto);

Post newPost = postRepository.save(post);

// convert entity to DTO

PostDto postResponse = mapToDTO(newPost);

return postResponse;

}

@Override

public List<PostDto> getAllPosts() {

List<Post> posts = postRepository.findAll();

return posts.stream().map(post -> mapToDTO(post)).collect(Collectors.toList());

}

@Override

public PostDto getPostById(long id) {

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

return mapToDTO(post);

}

@Override

public PostDto updatePost(PostDto postDto, long id) {

// get post by id from the database

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

Post updatedPost = postRepository.save(post);

return mapToDTO(updatedPost);

}

@Override

public void deletePostById(long id) {

// get post by id from the database

Post post = postRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Post", "id", id));

postRepository.delete(post);

}

// convert Entity into DTO

private PostDto mapToDTO(Post post){

PostDto postDto = new PostDto();

postDto.setId(post.getId());

postDto.setTitle(post.getTitle());

postDto.setDescription(post.getDescription());

postDto.setContent(post.getContent());

return postDto;

}

// convert DTO to entity

private Post mapToEntity(PostDto postDto){

Post post = new Post();

post.setTitle(postDto.getTitle());

post.setDescription(postDto.getDescription());

post.setContent(postDto.getContent());

return post;

}

}