@Entity

@Table(name="categories")

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer categoryId;

@Column(name="title",length = 100,nullable = false)

private String categoryTitle;

@Column(name="description")

private String categoryDescription;

//getters setters

}

public class CategoryDto {

private Integer categoryId;

@NotBlank

@Size(min = 4,message = "Min size of category title is 4")

private String categoryTitle;

@NotBlank

@Size(min = 10, message = "min size of cateogry desc is 10")

private String categoryDescription;

//getters setters

}

@Repository

public interface CategoryRepo extends JpaRepository<Category, Integer> {

}

public interface CategoryService {

CategoryDto createCategory(CategoryDto categoryDto);

CategoryDto updateCategory(CategoryDto categoryDto, Integer categoryId);

void deleteCategory(Integer categoryId);

CategoryDto getCategory(Integer categoryId);

List<CategoryDto> getCategories();

}

@Service

public class CategoryServiceImpl implements CategoryService {

@Autowired

private CategoryRepo categoryRepo;

@Autowired

private ModelMapper modelMapper;

@Override

public CategoryDto createCategory(CategoryDto categoryDto) {

Category cat = this.modelMapper.map(categoryDto, Category.class);

Category addedCat = this.categoryRepo.save(cat);

return this.modelMapper.map(addedCat, CategoryDto.class);

}

@Override

public CategoryDto updateCategory(CategoryDto categoryDto, Integer categoryId) {

Category cat = this.categoryRepo.findById(categoryId)

.orElseThrow(() -> new ResourceNotFoundException("Category ", "Category Id", categoryId));

cat.setCategoryTitle(categoryDto.getCategoryTitle());

cat.setCategoryDescription(categoryDto.getCategoryDescription());

Category updatedcat = this.categoryRepo.save(cat);

return this.modelMapper.map(updatedcat, CategoryDto.class);

}

@Override

public void deleteCategory(Integer categoryId) {

Category cat = this.categoryRepo.findById(categoryId)

.orElseThrow(() -> new ResourceNotFoundException("Category ", "category id", categoryId));

this.categoryRepo.delete(cat);

}

@Override

public CategoryDto getCategory(Integer categoryId) {

Category cat = this.categoryRepo.findById(categoryId)

.orElseThrow(() -> new ResourceNotFoundException("Category", "category id", categoryId));

return this.modelMapper.map(cat, CategoryDto.class);

}

@Override

public List<CategoryDto> getCategories() {

List<Category> categories = this.categoryRepo.findAll();

List<CategoryDto> catDtos = categories.stream().map((cat) -> this.modelMapper.map(cat, CategoryDto.class))

.collect(Collectors.toList());

return catDtos;

}

}

@RestController

@RequestMapping("/api/categories")

public class CategoryController {

@Autowired

private CategoryService categoryService;

// create

@PostMapping("/save")

public ResponseEntity<CategoryDto> createCategory(@Valid @RequestBody CategoryDto cateogDto) {

CategoryDto createCategory = this.categoryService.createCategory(cateogDto);

return new ResponseEntity<CategoryDto>(createCategory, HttpStatus.CREATED);

}

// update

@PutMapping("/update/{catId}")

public ResponseEntity<CategoryDto> updateCategory(@Valid @RequestBody CategoryDto categoryDto,

@PathVariable Integer catId) {

CategoryDto updatedCategory = this.categoryService.updateCategory(categoryDto, catId);

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

}

// delete

@DeleteMapping("/delete/{catId}")

public ResponseEntity<ApiResponse> deleteCategory(@PathVariable Integer catId) {

this.categoryService.deleteCategory(catId);

return new ResponseEntity<ApiResponse>(new ApiResponse("category is deleted successfully !!", true),

HttpStatus.OK);

}

// get

@GetMapping("/getcategory/{catId}")

public ResponseEntity<CategoryDto> getCategory(@PathVariable Integer catId) {

CategoryDto categoryDto = this.categoryService.getCategory(catId);

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

}

// get all

@GetMapping("/getallcategory")

public ResponseEntity<List<CategoryDto>> getCategories() {

List<CategoryDto> categories = this.categoryService.getCategories();

return ResponseEntity.ok(categories);

}

}









