Índice

Assuntos: Método Consultar Postagens por Id

Abertura Alunos:
Apresentação pitch
Luciana Rocha e Ellen Arruda

Relembrar é viver :

O padrão de arquitetura MVC (Model-View-Controller) é utilizado para organizar e estruturar o código de um software de maneira mais eficiente e fácil de entender. Ele divide as responsabilidades do sistema em três partes distintas: Model (Modelo), View (Visão) e Controller (Controlador).

O **"Modelo"** representa a lógica de negócios e os dados do aplicativo. Ele gerencia o acesso e a manipulação dos dados, garantindo a consistência e a integridade das informações.

A **"Visão"** é responsável pela apresentação dos dados ao usuário. Ela exibe as informações do Modelo de maneira compreensível e interativa, proporcionando uma interface para a interação do usuário.

O **"Controlador"** atua como intermediário entre o Modelo e a Visão. Ele recebe as entradas do usuário, processa essas entradas (como cliques ou comandos) e coordena as interações entre o Modelo e a Visão. Isso ajuda a manter a separação de preocupações e torna o código mais modular e fácil de dar manutenção.

O benefício do padrão MVC está na divisão clara de responsabilidades, o que facilita o desenvolvimento colaborativo, a manutenção do código e a escalabilidade do sistema. Cada componente desempenha um papel específico, permitindo que mudanças em uma parte do

sistema não afetem diretamente as outras, promovendo assim um código mais organizado e sustentável.

Comandos Controller

getById(@PathVariable Long id)

Ele retornará um Objeto específico persistido no Banco de dados, identificado pelo id (Identificador único do Objeto). Traçando um paralelo com o MySQL, seria o equivalente a instrução: SELECT * FROM tb_postagens where id = id;.

Explicação 2:
Retorna um Objeto específico da Classe Postagem persistido no Banco de dados. A Postagem é identificada pelo Atributo id.

@GetMapping("/{id}")

Mapeia todas as Requisições HTTP GET, enviadas para um endereço específico (Endpoint).

@PathVariable

Indica que o valor da variável na URL deve ser atribuído a um parâmetro no método do controlador. (insere o valor enviado no endereço do endpoint, na Variável de Caminho {id}, no parâmetro do Método getById(Long id);)

@GetMapping("/titulo/{titulo}")

Retorna todos os Objetos da Classe Postagem persistidos no Banco de dados, cujo Atributo título contenha a String enviada no parâmetro título do Método.

.map(resposta ⊠ ResponseEntity.ok(resposta))

Método map (Optional) e pode ter uma informação nula ou não (ele também trabalha como If), ele mapeia a resposta pelo findById(id), Se não for encontrado ele vai para orElse.

.orElse(ResponseEntity.notFound().build());:

Se não tiver encontrado a resposta do Map, ele seta o status de 404 não encontrado.

Método personalizado que está funcionando como a query:

SELECT * FROM tb_postagens WHERE titulo LIKE “%título%”;

- IgnoreCase - Ignorar letras minúsculas e maiúsculas.
- Containing - LIKE
- @Param - define que a String (titulo) é uma parâmetro para ser consultado
- Título se refere a coluna/atributo a ser buscado lá no MySQL.

return ResponseEntity.status(HttpStatus.CREATED).body(*postagem*Repository.save(*postagem*));

Comando que cria uma postagem e retorna um status de “postagem criada” no insomnia

@PostMapping

Persiste (salva/cria) um novo Objeto da Classe Postagem no Banco de Dados.

ATENÇÃO: O Endereço do Endpoint será igual ao Endereço do Recurso (@RequestMapping). O Método getAll() utiliza o mesmo endereço, porém como se tratam de verbos diferentes (O primeiro utiliza o verbo GET e o segundo utiliza o verbo POST) o endereço pode ser o mesmo.

@valid

Esta anotação valida o Objeto Postagem enviado no Corpo da Requisição (Request Body), conforme as regras definidas na Model passadas para a tabela Postagem (@NotNull, @NotBlank, @Size e etc).

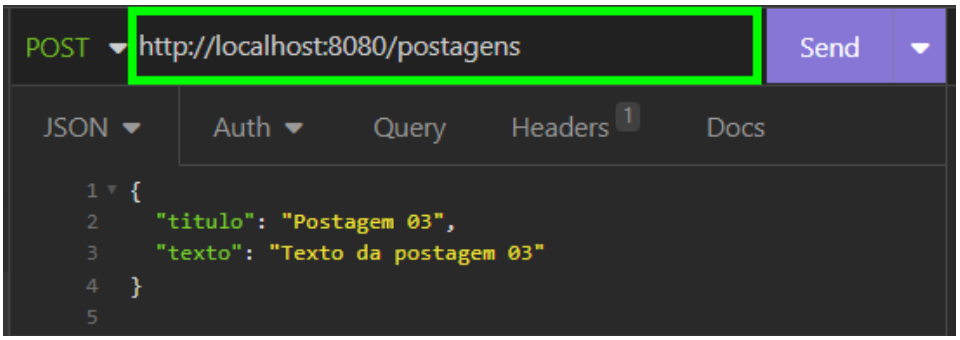
Obs: Para acessar pelo Insomnia usará a busca pelo método POST com o endereço http://localhost:8080/postagens, pois não foi indicado nenhum local, no Body selecione JSON. Abrirá um editável onde deve colocar o post e ele criará um novo post.

JSON

"JavaScript Object Notation", é um formato amplamente utilizado em serviços web, APIs e configurações de dados. A resposta de uma API REST, por exemplo, é frequentemente formatada como JSON.

Verde: atributo
Amarelo: String

Exemplo:



JSON: o texto antes dos 2 pontos (:) é o Atributo da Classe e o texto depois dos 2 pontos (:) é o dado que será cadastrado no Atributo. Os Atributos são separados por virgula, como mostra a imagem acima.

A segunda parte do endereço é o endpoint configurado na anotação @RequestMapping, em nosso caso /postagens.

@ResponseStatus

Indica que o Método delete(Long id), terá um Status HTTP específico quando a Requisição for bem sucedida, ou seja, será retornado o HTTP Status NO_CONTENT 204, ao invés do HTTP Status OK 200 como resposta padrão do Método

@DeleteMapping("/{id}")

Mapeia todas as Requisições HTTP DELETE, enviadas para um endereço específico (Endpoint).

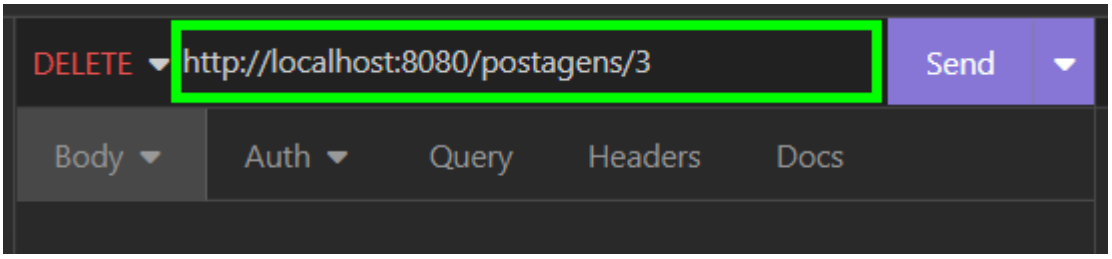
delete(@PathVariable Long id)

Foi definido com o tipo void, porque ao deletar um Objeto da Classe Postagem do Banco de dados, ela deixa de existir, logo não existe um Objeto para ser Retornado.

```
64
65      @ResponseStatus(HttpStatus.NO_CONTENT)
66      @DeleteMapping("/{id}")
67      public void delete(@PathVariable Long id) {
68          Optional<Postagem> postagem = postagemRepository.findById(id);
69
70          if(postagem.isEmpty())
71              throw new ResponseStatusException(HttpStatus.NOT_FOUND);
72
73          postagemRepository.deleteById(id);
74      }
75
```

Insonia Delete

Configure a requisição conforme a imagem abaixo:



- A primeira parte do endereço (http://localhost:8080) é o endereço do nosso servidor local. Quando a API estiver na nuvem, ele será substituído pelo endereço da aplicação na nuvem.
- A segunda parte do endereço é o endpoint configurado na anotação @RequestMapping, em nosso caso /postagens/.
- A terceira parte (/3) é a variável de caminho (@PathVariable) id. Informe o id da postagem que você deseja apagar.

Query Métodos

Palavra	Instrução SQL
find	SELECT
All	*
By	WHERE
Titulo	Atributo da Classe Postagem
Containing	LIKE "%título%"
IgnoreCase	Ignorando letras maiúsculas ou minúsculas
@Param("titulo")	Define a variável String titulo como um parâmetro da consulta. Esta anotação é obrigatória em consultas do tipo Like.
String titulo	Parâmetro do Método contendo o título que você deseja procurar.

Bibliotecas adicionadas no Blog Pessoal

- org.springframework.http.HttpStatus;
- org.springframework.web.bind.annotation.PathVariable;
- org.springframework.web.bind.annotation.PostMapping;
- org.springframework.web.bind.annotation.RequestBody;
- import.jakarta.validation.Valid;
- import.java.util.list;
- org.springframework.data.repository.query.Param;
- org.springframework.web.bind.annotation.DeleteMapping;
- org.springframework.web.bind.annotation.ResponseStatus;
- org.springframework.web.server.ResponseStatusException;
- org.springframework.web.bind.annotation.PutMapping;

Links e afins

CookBook :

Id:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/08.md

Titulo:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/09.md

Post:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/10.md

Atualizar Postagem:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/11.md

Método Deletar:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/04_spring/12.md

Documentação JPA Query Methods -
https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html

Código fonte BLOG PESSOAL:
https://github.com/conteudoGeneration/backend_blogpessoal_v3/tree/07_Classe_PostagemController_post/blogpessoal

Documentações de apoio:

Documentação: @GetMapping
Documentação: @PathVariable
Documentação: @ResponseEntity
Documentação: .HttpStatus
Documentação: .findById(Long id)
Documentação: Optional
Documentação: Expressões Lambdas
Documentação: Long
Documentação: Artigo: Classes Wrappers (Long)
Documentação: .save(T entidade)
Documentação: @RequestBody
Documentação: @Valid
Documentação: Java Generics
Documentação:Query Methods
Documentação:@DeleteMapping
Documentação: .deleteById()
Documentação: @PutMapping

Dúvidas