SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

Controladores

Desenvolvimento de Sistemas

Prof. Me. Reneilson Santos

Abril/2024

SENAI
PELO FUTURO DO TRABALHO

Agenda

- Controladores
  - Injeção de Repositórios
  - Operações de Escrita
  - Atualização de Dados
- Adicionando Swagger
- Bibliografia

Injeção de Repositórios

Injeção de Repositórios

Para injetar o repositório, portanto, basta fazermos uso da anotação **@Autowired**, que será usada para injeção de dependência inversa.

Como essa classe de controlador possui uma anotação do Spring Boot (**@RestController**), a nossa aplicação consegue fazer a injeção de dependência sem precisarmos fazer nada além da declaração da variável.

```
@RestController
@RequestMapping("pets")
public class PetController {
    @Autowired
    private PetRepository repository;
    …
}
```

```
@RestController
@RequestMapping("tags")
public class TagController {
    @Autowired
    private TagRepository repository;
    …
}
```

Usando Repositório

Para usar em nossos método de Request, no nosso Controlador, basta chamarmos a variável do repositório criada com o seu respectivo método (o JPARepository já tem os principais métodos necessários para um CRUD padrão:

- **CREATE** : método **save** passando o objeto que deseja salvar no banco (sem id);
- **READ**: método **findAll** sem passar nada ou passando paginação (*Pageable*);
- **READ** 1 elemento: método **findById** passando o id.
- **UPDATE**: método **save** passando um objeto com id diferente de null;
- **DELETE**: método **delete** passando um objeto ou **deleteById** passando o id;

Controladores

Controladores

Na abordagem do Spring para construir API Web RESTful, as solicitações HTTP são tratadas por um controlador.

Esses componentes são identificados pela anotação **@RestController**.

Nós já utilizamos controladores na aula introdutória do Spring, o que faremos agora é conectá-lo aos nossos serviços para que os dados sejam acessados a partir de um banco de dados.

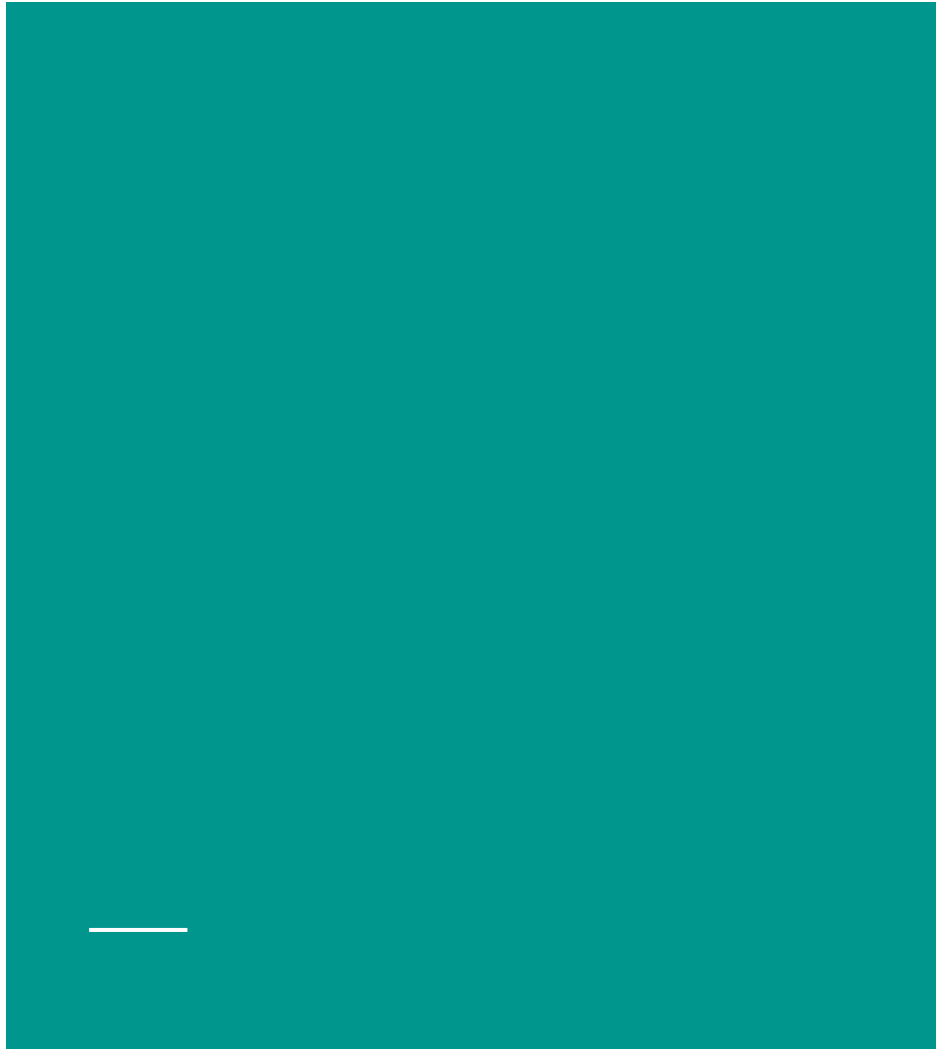
http://localhost:8080/categories

Controladores

Para envolver seu repositório com uma camada da web, você deve recorrer ao Spring MVC. Graças ao Spring Boot, há pouca infraestrutura para codificar.

- **@RestController** indicará que os dados retornados por cada método serão gravados diretamente no corpo da resposta em vez de renderizar um modelo.
- Temos **rotas para cada operação** (@GetMapping, @PostMapping, @PutMapping e @DeleteMapping, correspondendo a chamadas HTTP GET, POST, PUT e DELETE).

Response Entity



Response Entity

É importante retornar os resultados corretos (padronizados) para os métodos HTTP. O **ResponseEntity** é a classe que permitirá definir qual o status de retorno de determinada requisição bem como retornar o objeto (serializável) na forma de JSON.

Para isso o retorno da função é o ResponseEntity<?> (podendo identificar qual a classe de retorno entre < > ou deixar como opcional (?).

Response Entity

O retorno ResponseEntity pode receber como parâmetros no construtor o objeto de retorno e o status Http referente (utilizando a classe **HttpStatus** pode-se utilizar qualquer um dos status HTTP como já visto).

Também é possível usar os métodos estáticos do ResponseEntity para criar alguns retornos.

Passagem de Parâmetros

Passagem de Parâmetros

Existem 3 formas de passar parâmetros em uma API Rest:

1. Através do caminho da URL [PathVariable]
    a. http://localhost:8080/pets/**1**
2. Uma query através da URL [RequestParam]
    a. http://localhost:8080/pets**?category=DOG**
3. Através do body (envio de JSON) [RequestBody]
    a. {"name": "Spike", "status": "AVAILABLE", "category": "DOG"}

@PathVariable

O @PathVariable é usado para mapear variáveis da URL para parâmetros de método em um controlador Spring.

Ele é geralmente usado para capturar valores de variáveis de caminho (por exemplo, /pets/1, onde "1" é uma variável de caminho).

Você pode usá-lo em métodos de controlador para extrair valores de variáveis de caminho e usá-los como argumentos no método.

```
@GetMapping("/{id}")
public ResponseEntity<Pet> read(@PathVariable Long id){
    Pet pet = repository.findById(id).get();
    return ResponseEntity.ok(pet);
}
```

```
@DeleteMapping("/{id}")
public ResponseEntity<?> delete(@PathVariable Long id){
    repository.deleteById(id);
    return ResponseEntity.noContent().build();
}
```

@RequestParam

O @RequestParam é usado para mapear parâmetros de consulta (query parameters) da URL para parâmetros de método em um controlador Spring.

Parâmetros de consulta são aqueles que vêm após o ponto de interrogação na URL (por exemplo, /pets?category=DOG).

O @RequestParam permite que você obtenha valores desses parâmetros de consulta e os utilize em métodos de controlador, geralmente é usado para filtragem de dados, como por exemplo em um método GET.

@RequestBody

O @RequestBody é usado para mapear o corpo da solicitação HTTP para um objeto Java em métodos de controlador Spring.

É frequentemente usado ao lidar com solicitações POST e PUT, onde os dados são enviados no corpo da solicitação em vez de parâmetros de consulta no formato JSON.

O Spring Boot converte automaticamente os dados do corpo da solicitação (JSON) no objeto Java especificado.

```java
@PostMapping
public ResponseEntity<?> create(@RequestBody Pet pet){
    Pet createdPet = repositoy.save(pet);
    return new ResponseEntity<Pet>(createdPet, HttpStatus.CREATED);
}

public ResponseEntity<?> update(@RequestBody Pet pet){
    Pet updatedPet = repositoy.save(pet);
    return ResponseEntity.ok(updatedPet);
}
```

Exemplo Controlador Pet

Adicionando Swagger

Adicionando Documentação

Para adicionar a documentação Swagger no nosso projeto Spring Boot e facilitar os nossos testes é bem simples, basta adicionarmos a biblioteca do open API dentro do nosso pom.xml e pronto.

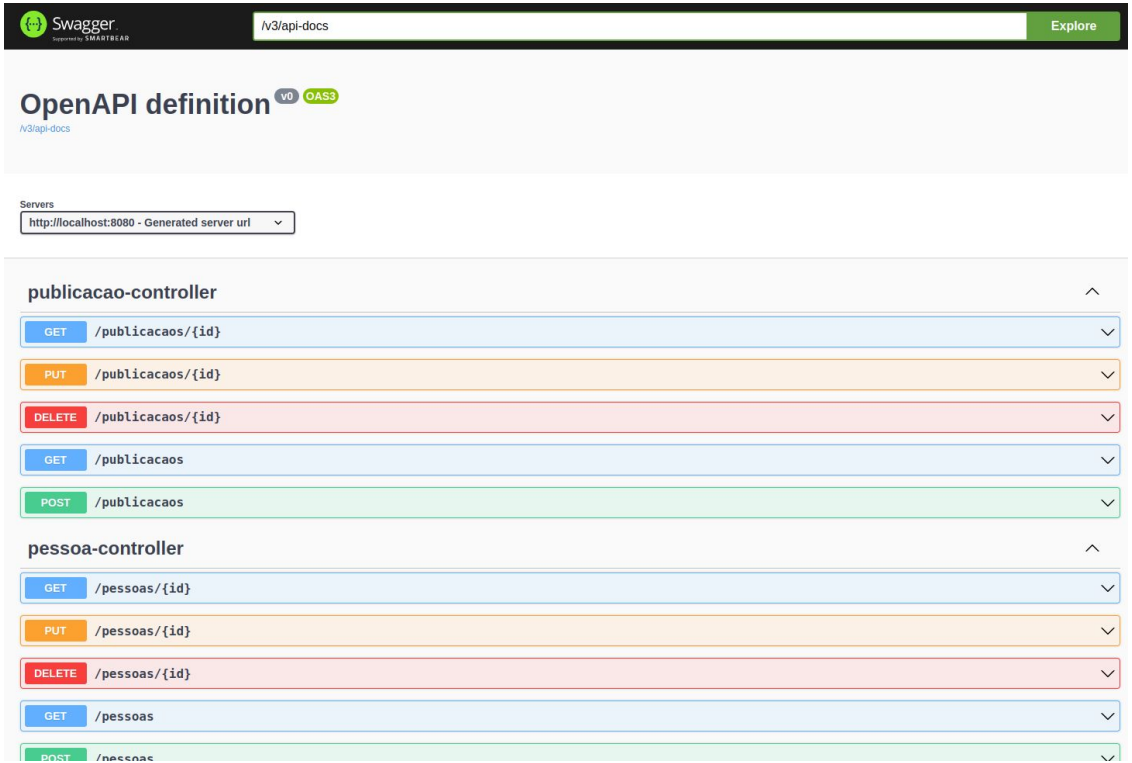
https://springdoc.org/

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.2.0</version>
 </dependency>
```

Adicionando Documentação

Ao adicionar esta dependência e reexecutar o projeto, basta acessar o endpoint:

http://localhost:8080/swagger-ui.html



Exercício

Criar repositórios e controladores com métodos GET, PUT, POST e DELETE para os modelos.

Bibliografia

Bibliografia

- JPA Repositories (https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/html/jpa.repositories.html)
- Accessing Data with JPA (https://spring.io/guides/gs/accessing-data-jpa/)
- Spring Data Partial Update (https://www.baeldung.com/spring-data-partial-update)
-