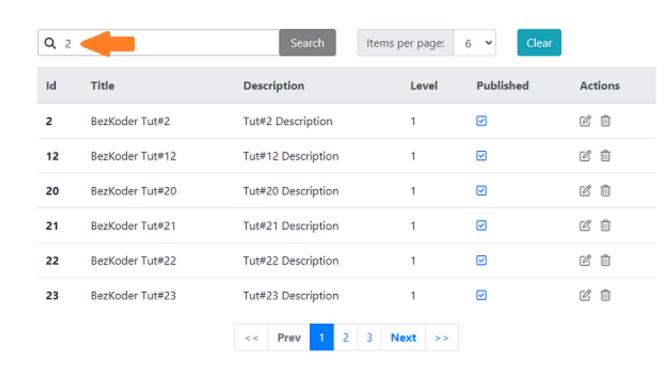
# **AULA 17 - PAGINAÇÃO E ORDENAÇÃO**

Disciplina de Backend - Professor Ramon Venson - SATC 2024

# Paginação e Ordenação

- Evitar o excesso de tráfego de saída (limite de resultados);
- Organizar os dados para o cliente (ordenação);
- Utilizada (geralmente) com requisições GET;



# Paginação

Para realizar a paginação de uma requisição, vamos criar um PageRequest (que implementa a interface Pageable ).

```
Pageable page = PageRequest.of(paginaAtual, tamanhoPagina);
Page<Animal> list = animalRepository.findAll(page);
```

No método, podemos passar a página atual (começando do 0) e o tamanho da página (quantos resultados em cada página).

#### Paginação no Controllerlista

No controlador, vamos precisar solicitar do cliente a página e o número de resultados que ele pretende acessar, repassando essa configuração ao serviço.

A paginação é feita usando de *Query Strings* e, portanto, o @RequestParam . Ex.:

http://localhost:8080/animais?pagina=5&resultados=5

```
@GetMapping
@ResponseStatus(HttpStatus.OK)
public List<AnimalDto> getAllAnimaisByPageAndSort(
          @RequestParam(defaultValue = "0") Integer pagina,
          @RequestParam(defaultValue = "10") Integer resultados) {
    return animalService.getAllAnimais(pagina, resultados);
}
```

O uso do default Value serve para tornar o uso facultativo na Query String

#### Paginação no Service

No service, vamos criar um objeto Pageable com as configurações de paginação.

```
public List<AnimalDto> getAllAnimais(Integer pagina, Integer resultados) {
    // Gera uma configuração de paginação
    Pageable page = PageRequest.of(pagina, resultados);
    // Consulta no repositório
    Page<Animal> list = animalRepository.findAll(page);
    // Conversão de todos os objetos da página para um DTO
    return list.stream().map(animalMapper::toDto).toList();
}
```

# Ordenação

Podemos adicionar à configuração de paginação uma ordenação para que a lista de retorno esteja organiza por um ou mais atributos:

```
Pageable page = PageRequest.of(
    paginaAtual,
    tamanhoPagina,
    Sort.by('nome').ascending() // Ordena por nome em ordem ascendente
);
Page<Animal> list = animalRepository.findAll(page);
```

Os métodos ascending e descending podem definir a direção da ordenação.

A ordenação é realizada antes da paginação pelo Spring Data

Podemos utilizar também uma lista com mais de um atributo para ordenação, porém, precisaremos converter/mapear a lista para o tipo de dado adequado:

Sort.Order:

```
// List<String> sortList -> ["nome", "idade"]
List<Sort.Order> ordemLista = sortList.stream().map((sort) -> {
    return new Sort.Order(Sort.Direction.ASC, sort);
}).toList();
Pageable page = PageRequest.of(
    paginaAtual,
    tamanhoPagina,
    Sort.by(ordemLista)
);
Page<Animal> list = animalRepository.findAll(page);
```

#### Ordenação no Controller

No controller, vamos receber mais uma Query String usando o @RequestParam. A URL da requisição será algo parecido com isso: http://localhost:8080/animais?

pagina=0&resultados=5&ordenar=idade, nome

```
@GetMapping
@ResponseStatus(HttpStatus.OK)
public List<AnimalDto> getAllAnimaisByPageAndSort(
          @RequestParam(defaultValue = "-1") Integer pagina,
          @RequestParam(defaultValue = "-1") Integer resultados,
          @RequestParam(defaultValue = "") List<String> ordenar) {
    return animalService.getAllAnimais(page, results, ordenar);
}
```

#### Ordenação no Service

Por fim, nosso service ficará da seguinte maneira:

```
public List<AnimalDto> getAllAnimais(Integer pagina, Integer resultados, List<String> ordenar ) {
    // Mapeia a lista de strings para Sort.Order
    List<Sort.Order> ordemFinal = sortList.stream()
    .map((ordenar) -> {
        return new Sort.Order(Sort.Direction.ASC, sort);
    }).toList();
    // Cria a configuração de paginação
    Pageable page = PageRequest.of(
        pagina,
        resultados,
        Sort.by(ordemFinal)
    // Faz a requisição
    Page<Animal> list = animalRepository.findAll(page);
    // Converte a página para um DTO
    return list.stream().map(animalMapper::toDto).toList();
```

### O que aprendemos hoje

- Realizar a paginação de resultados
- Ordenar resultados