08

Criando um filtro dinâmico de produtos

O que acontece se um cliente precisar listar todos os **e-book**s que custem R$**15,00**. Precisamos criar uma rota que receba dois parâmetros? Ou deixamos o cliente se virar para filtrar do lado dele? Prefiro uma terceira solução: criar um sistema de filtros dinâmicos, como se o usuário estivesse preenchendo um formulário com diversos campos opcionais. Por exemplo, a URL `localhost:9000/api/produtos?tipo=e-book&preco=15` .

Essa solução é bastante genérica, tanto que merece o lugar da rota raiz da API. Vamos alterar as rotas então, deixando o conjunto assim:

```
GET /api/produtos            controllers.ApiController.comFiltros
GET /api/produtos/todos      controllers.ApiController.todos
GET /api/produtos/tipo/:tipo controllers.ApiController.comTipo(tipo)
```

A primeira rota é a que criamos agora. Como os parâmetros virão de modo dinâmico, não precisamos declará-los em lugar algum. Para receber os parâmetros da URL, usaremos a fábrica de formulários e pegaremos todos eles de uma vez!

```
public Result comFiltros() {
    Map<String, String> parametros = formularios.form().bindFromRequest().data();
    EnvelopeDeProdutos envelope = new EnvelopeDeProdutos(produtoDAO.comFiltros(parametros));
    return ok(Json.toJson(envelope));
}
```

E agora vem a parte interessante: incluir todos os parâmetros na mesma query. Para isso precisamos, no DAO, armazenar a consulta numa variável e percorrer os parâmetros. Como cada chave representa um nome de atributo e cada valor é o valor do atributo que queremos, podemos adicionar todos com poucas linhas de código.

```
public List<Produto> comFiltros(Map<String, String> parametros) {
    ExpressionList<Produto> consulta = produtos.where();
    parametros.entrySet().forEach(entrada -> {
        consulta.eq(entrada.getKey(), entrada.getValue());
    });
    return consulta.findList();
}
```

Se por acaso fizermos a consulta por **tipos**, no entanto, ocorre uma exceção já que o atributo não existe no modelo. Podemos então fazer uma lista apenas com os atributos do modelo e conferir se eles são os únicos presentes na requisição.

```
private static final List<String> ATRIBUTOS = Arrays.asList("id", "titulo", "codigo", "tipo", "
```

Mas Marco, se a gente alterar nosso modelo de Produto precisamos alterar essa lista também? No escopo desse curso, sim! Poderíamos utilizar *Reflexão* para listar os atributos do modelo, como é ensinado no curso **Java Reflection: mágica e meta programação**, mas aqui vamos utilizar essa pequena repetição para validar os parâmetros.

```
private void validaParametros(DynamicForm formulario) {
    Map<String, String> parametros = formulario.data();
    parametros.keySet().forEach(chave -> {
        if (!ATRIBUTOS.contains(chave)) {
            formulario.reject(new ValidationError("Atributos inválidos", chave));
        }
    });
}
```

E então executamos a validação antes de fazer a pesquisa no banco de dados.

```
public Result comFiltros() {
    DynamicForm formulario = formularios.form().bindFromRequest();
    validaParametros(formulario);
    if (formulario.hasErrors()) {
        JsonNode erros = Json.newObject().set("erros", formulario.errorsAsJson());
        return badRequest(erros);
    }
    Map<String, String> parametros = formulario.data();
    EnvelopeDeProdutos envelope = new EnvelopeDeProdutos(produtoDAO.comFiltros(parametros));
    return ok(Json.toJson(envelope));
}
```

Temos então um endereço que aceita filtros dinâmicos, e permite quaisquer combinações dentre os atributos do nosso modelo!