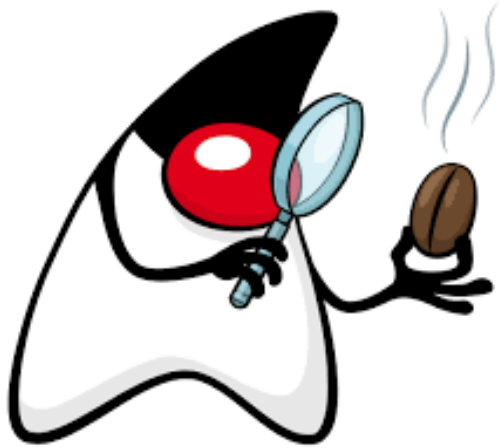
Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 05 – SPRING DATA JPA – Continuação....

Validação com Bean Validation



Persistência

- Acabamos de usar o Flyway migration para criar o schema do BD, porém faltou realizar a validação que estão chegando na api.
- Será que essas informações estão vindo correta.
- Será que existe campos obrigatórios.
- Será que existe campos apenas numéricos ou de apenas letras.
- Tamanho mínimo ou máximo.
- Campo de data.
- Teremos que **validar**

Spring

- Esse tipo de validação é algo corriqueiro, e já existe um modulo no Spring Boot para lidar com as validações, que inclusive já incluirmos no projeto.
- Basta visualizar no pom.xml!!!

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

- Vamos acessar nosso controller e observar o método cadastrar. Ele recebe o DTO DadosCadastroMédico como parâmetro.
- Os campos que estão como parâmetro, são os campos que estão chegando na requisição.
- Se o acessarmos (DadosCadastroMédico), clicando sobre ele com o botão esquerdo + a tecla Ctrl, veremos os campos que estão chegando na requisição.
- São eles: String nome, String email, String crm, Especialidade especialidade, e DadosEndereco endereco.

- É nesses que usaremos o Bean Validation, a partir de anotações.
- Ele vai verificar, no caso, se as informações que chegam estão de acordo com as anotações.
- Vamos adicionar uma anotação a cada um dos atributos, começando pelo atributo nome, que é obrigatório e não pode ser nulo e que, também, não pode ser vazio: precisa de um texto.
- Para informar isso ao Bean Validation, passaremos a anotação @NotBlank.

- Para dar a formatação de e-mail, passaremos também a anotação @Email.
- Acima de String crm, vamos passar @NotBlank e @Pattern, porque ele é um número de 4 a 6 dígitos.
- Dentro da segunda anotação, para esclarecer a quantidade de dígitos passaremos a expressão regular (regexp = "\\d{4,6}").

- Como Especialidade especialidade é um campo obrigatório, vamos adicionar a anotação @NotNull.
- Por fim, acima de DadosEndereco endereco, passaremos a anotação @NotNull, porque ele também não pode ser nulo, e @Valid, para validar o DTO: Como ele é um record, também será validado

Spring

```
public record DadosCadastroMedico(
        @NotBlank
        String nome,
        @NotBlank
        @Email
        String email,

        @NotBlank
        @Pattern(regexp = "\\d{4,6}")
        String crm,
        @NotNull
        Especialidade especialidade,

        @NotNull @Valid DadosEndereco endereco) {
}
```

Spring

```
public record DadosCadastroMedico(
        @NotBlank
        String nome,
        @NotBlank
        @Email
        String email,
        @NotBlank
        @Pattern(regexp = "\\d{4,6}")
        String crm,
        @NotNull
        Especialidade especialidade,

        @NotNull @Valid
    DadosEndereco endereco) {
    }
```

Spring

```
public record DadosEndereco(
        @NotBlank
        String logradouro,
        @NotBlank
        String bairro,
        @NotBlank
        @Pattern(regexp = "\\d{8}")
        String cep,
        @NotBlank
        String cidade,
        @NotBlank
        String uf,
        String complemento,
        String numero) {
}
```

Spring

```
public record DadosEndereco(
        @NotBlank
        String logradouro,
        @NotBlank
        String bairro,
        @NotBlank
        @Pattern(regexp = "\\d{8}")
        String cep,
        @NotBlank
            String cidade,
            @NotBlank
            String uf,
            String complemento,
            String numero) {
        }
```

- Agora vamos acessar "MedicoController.java". Lá, adicionaremos @Valid, para solicitar que o Spring se integre ao Bean Validation e execute as validações.

```
@PostMapping
@Transactional
public void cadastrar(@RequestBody @Valid DadosCadastroMedico dados) {
    repository.save(new Medico(dados));
```

- Vamos abrir o Insomnia para executar um teste.
- Vamos disparar uma requisição, mas deixando o campo "nome" em branco.
- Como resultado, receberemos o erro "400 Bad Request".
- Isso acontece porque informamos, via Bean Validation, que esse campo não pode ser nulo.

Spring

- No Insomnia.


```
{
  "timestamp": "2024-03-17T12:29:57.347+00:00",
  "status": 400,
  "error": "Bad Request",
  "trace":
"org.springframework.web.bind.MethodArgumentNotValidException:
Validation failed for argument [0] in public void
med.voll.api.controller.MedicoController.cadastrar(med.voll.api.medico.
DadosCadastroMedico): [Field error in object 'dadosCadastroMedico' on
field 'nome': rejected value []; codes
[NotBlank.dadosCadastroMedico.nome,NotBlank.nome,NotBlank.java.lang.Str
ing,NotBlank]; arguments
[org.springframework.context.support.DefaultMessageSourceResolvable:
codes [dadosCadastroMedico.nome,nome]; arguments []; default message
[nome]]; default message [não deve estar em branco]] \r\n\tat
```

- Preencha o campo “nome”, e dispare uma nova requisição.
- Como resultado, receberemos o código “200 ok".
- Para confirmar veja o BD....
- Caso você queira saber mais sobre validações

Jakarta Bean Validation specification

- https://jakarta.ee/specifications/bean-validation/3.0/jakarta-bean-validation-spec-3.0.html#builtinconstraints

Nova Migrations

- No cadastro do médico esquecemos de adicionar o campo "Telefone" à entidade médico.
- Vamos aprender a adicionar uma coluna ao banco de dados quando a ***migration*** já existe.
- Vamos acessar "MedicoController.java > DadosCadastroMedico". Precisaremos criar um atributo para receber esse novo campo.
- Vamos adicioná-lo logo após o campo String email. Como ele será obrigatório, passaremos a anotação @NotBlank String telefone.

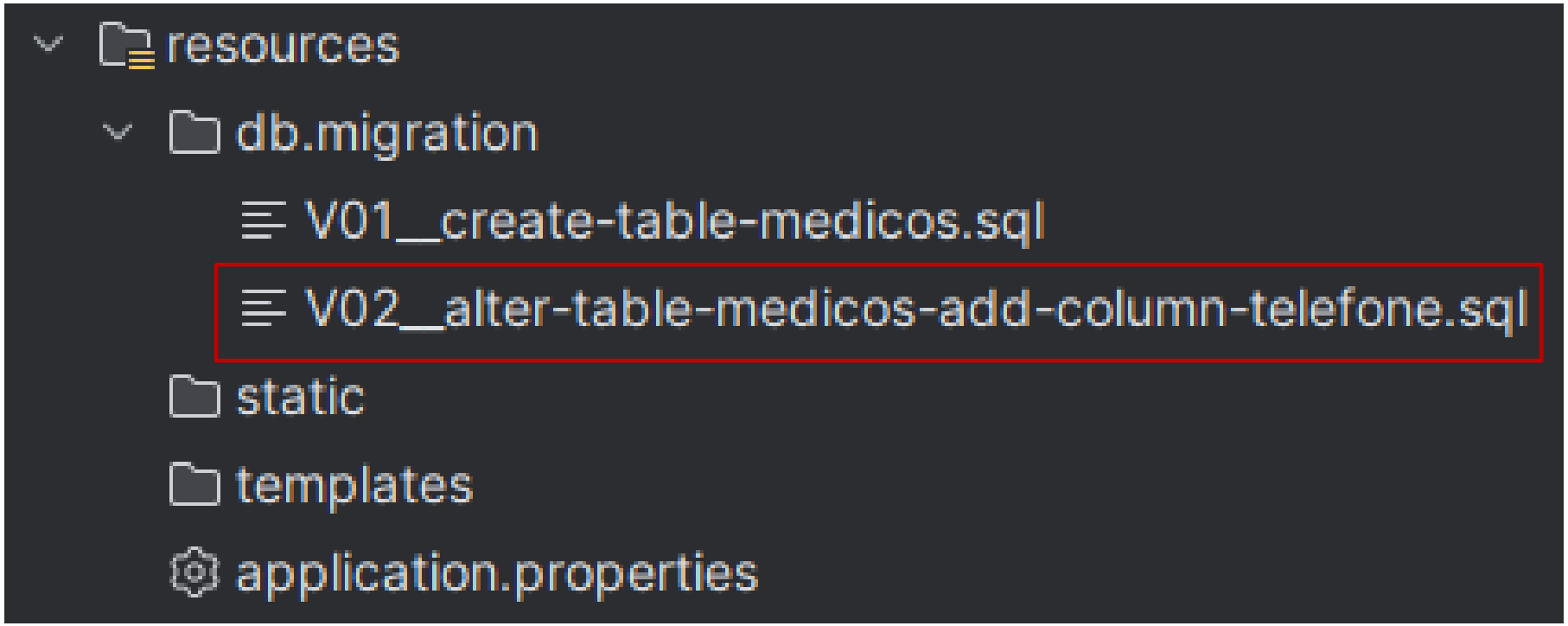
- Vamos precisar fazer alterações também na entidade JPA "Medico.java". Lembrando fica no pacote medico.
- Lá, adicionaremos o atributo abaixo de private String email;. Será o atributo private String telefone;.
- Nesse mesmo arquivo, vamos informar ao construtor public Medico, que recebe DadosCadastroMedico, desse novo atributo. Faremos isso passando this.telefone = dados.telefone():

- Já temos uma migration criada, "V1__create-table-medicos.sql". Porém, não podemos alterá-la, porque migrations executadas no banco de dados não podem mais ser alteradas.
- Portanto, para passar o atributo telefone para o banco de dados, precisaremos criar uma nova migration.

- Mas, antes disso, precisamos parar o projeto, clicando no ícone do quadrado vermelho, no canto superior direito.

- **Obs: Sempre interrompa o projeto ao usar migrations.**

- Vamos criar um novo arquivo, chamado "**V02__alter-table-medicos-add-column-telefone.sql**", dentro da pasta "db.migration".



- Nele vamos digitar o código SQL abaixo, para alterar o banco de dados:

- alter table medicos add telefone varchar(20) not null;

- **é importante sempre parar o projeto ao criar os arquivos de migrations, para evitar que o Flyway os execute antes da hora**

Spring

- Vamos iniciar o projeto.
- Veja o log no terminal, note que a tabela foi atualizada corretamente.

```
Database: jdbc:mysql://localhost:3306/vollmed_api (MySQL 8.0)
Successfully validated 2 migrations (execution time 00:00.065s)
Current version of schema `vollmed_api`: 01
Migrating schema `vollmed_api` to version "02 - alter-table-medicos-add-column-telefone"
Successfully applied 1 migration to schema `vollmed_api`, now at version v02 (execution time 00:00.173s)
```

- Vamos no Insomnia, para cadastrar um nove registro.
- Tente cadastrar um novo médico sem o telefone (“erro 400 Bad Request”).
- Depois passando o telefone.

Erro na Migrations

Spring

- Eventualmente pode acontecer de esquecermos de parar o projeto e algum erro acontecer ao tentar inicializar a aplicação.
- Nesse caso será exibido o seguinte erro ao tentar inicializar a aplicação:

```
Exception encountered during context initialization - cancelling
refresh attempt:
org.springframework.beans.factory.BeanCreationException: Error
creating bean with name 'flywayInitializer' defined in class path
resource
[org/springframework/boot/autoconfigure/flyway/FlywayAutoConfiguration
$FlywayConfiguration.class]: Validate failed: Migrations have failed
validation
```

- Perceba na mensagem de erro que é indicado que alguma migration falhou, impedindo assim que o projeto seja inicializado corretamente.
- Esse erro também pode acontecer se o código da migration estiver inválido, contendo algum trecho de SQL digitado de maneira incorreta.

- Para resolver esse problema será necessário acessar o banco de dados da aplicação e executar o seguinte comando sql:

```sql
delete from flyway_schema_history where success = 0;
```

- O comando serve para apagar na tabela do Flyway todas as migrations cuja execução falhou.
- Após isso, basta corrigir o código da migration e executar novamente o projeto.

- Obs: Pode acontecer de alguma migration ter criado uma tabela e/ou colunas e com isso o problema vai persistir, pois o flyway não vai apagar as tabelas/colunas criadas em migrations que falharam.
- Nesse caso você pode apagar o banco de dados e criá-lo novamente:

```
drop database vollmed_api;
create database vollmed_api;
```

Finalizamos o Spring Data JPA