Recebendo Dados com DTOs

Usando Records e Enums no Spring Boot

Rômulo C. Silvestre

September 8, 2025

O Problema e a Solução: Padrão DTO

O Problema

Receber um JSON como uma `String` única é ineficiente e difícil de manipular.

A Solução: Padrão DTO (Data Transfer Object)

- Criar uma classe (ou `record`) que representa os dados da requisição.
- O Spring converte o JSON para um objeto desta classe automaticamente.

Antes

```
public void cadastrar(
  @RequestBody String json)
```

Depois (com DTO)

```
public void cadastrar(
  @RequestBody
  DadosCadastroMedico dados)
```

Construindo os DTOs com record e enum

DTO Principal: DadosCadastroMedico.java

```
public record DadosCadastroMedico(
    String nome,
    String email,
    String crm,
    Especialidade especialidade, // Usando um Enum
    DadosEndereco endereco       // Usando outro Record
) {}
```

DTO Aninhado: DadosEndereco.java

```
public record DadosEndereco(
    String logradouro,
    String bairro,
    String cep,
    ...
) {}
```

Enum: Especialidade.java

```
public enum Especialidade {
    ORTOPEDIA,
    CARDIOLOGIA,
    GINECOLOGIA,
    DERMATOLOGIA;
}
```

Teste, Erro Comum e Conclusão

Erro Comum: `400 Bad Request`

Ocorreu um erro de "desserialização" ao enviar o JSON.

- **Causa:** O valor no JSON (`"ortopedia"`) não correspondia ao valor no `enum` (`ORTOPEDIA`).
- **Solução:** Ajustar o valor no JSON para corresponder ao `enum` (letras maiúsculas).

Resultado no Console Após a Correção

O `System.out.println(dados)` exibe o objeto DTO completo:

```
DadosCadastroMedico[nome=..., especialidade=ORTOPEDIA, endereco=DadosEndereco[...]]
```

Conclusão

O padrão DTO é a maneira correta de representar e validar os dados que entram e saem da API, garantindo um código mais limpo, seguro e organizado.