

Logo da Orange Talents

# Conversão de um objeto de entrada para um objeto de domínio

## Cenário:

Com o conhecimento sobre Micronaut, imagine que precisamos implementar uma operação de finalização de compra em um E-Commerce via uma API REST, esta API receberá todos os pedidos com os itens que foram selecionados pelo usuário até o momento de concluir a compra. Este pedido deve possuir as seguintes informações com suas respectivas restrições:

* todo pedido deve ter o nome e email do cliente;
* o email deve ser válido;
* o pedido deve possuir um ou mais itens de pedido;
* todo item de pedido deve ter código do produto, preço do produto e quantidade;

Não iremos gravar nada no banco de dados, mas para finalizarmos a compra precisamos converter os dados de entrada da API REST para o nosso modelo de domínio da aplicação, que por sua vez segue esse desenho:

class Pedido(  
 val cliente: Cliente,  
 val total: BigDecimal,  
 val itens: List<ItemDePedido>  
)  
  
class Cliente(val nome: String, val email: String)  
  
class ItemDePedido(  
 val codigo: String,  
 val preco: BigDecimal,  
 val quantidade: Int  
)

Em caso de sucesso, nossa API REST deve retornar o status HTTP 200, caso contrário ela deve retornar o status HTTP 400 referente aos erros de validação dos dados submetidos para aplicação.

Como você implementaria uma API REST para a atividade acima?

## O que seria bom ver nessa resposta?

* **Peso 4**: Criação de um controller com um endpoint do tipo POST recebendo uma request com os dados do pedido, com o DTO de request anotado com as anotações da Bean Validation, e por fim retornando um status HTTP 200 em caso de sucesso ou status HTTP 400 em caso de erro de validação;
* **Peso 2**: Criação de um método toModel() ou similar no DTO de request para converter o DTO para uma instância de Pedido do nosso modelo de domínio;
* **Peso 3**: Implementação da lógica para calcular o valor total do pedido deve estar dentro do DTO de pedido, e do valor total do item de pedido deve estar dentro do DTO de item de pedido;
* **Peso 1**: Implementação da validação do tamanho máximo da lista de itens de pedidos;

## O que penaliza sua resposta?

* **Penalidade -5**: Utilização do modelo de domínio (Pedido, ItemDePedido e Cliente) como entrada dos dados da API REST;

## Resposta do Especialista:

* Crio uma classe de controller, anoto ela com @Controller, crio um método finalizar() anotado com @Post("/api/pedidos/finaliza"). Para o payload JSON recebido como parâmetro, eu crio um DTO NovoPedidoRequest que recebe **nome** e **email**, ambos como String, e um atributo **itens** do tipo List, onde cada instância de item de pedido é um DTO do tipo ItemDePedidoRequest contendo os atributos **codigo** como String, **preco** como BigDecimal e um atributo **quantidade** como Int. Por se tratar de DTOs, eu os crio como data class do Kotlin;
* Na classe NovoPedidoRequest, eu a anoto com @Introspected do Micronaut por causa do AOT (Ahead of Time), e também anoto os atributos da classe com as anotações da Bean Validation:
  + atributo **nome** com @field:NotBlank;
  + atributo **email** com @field:NotBlank @field:Email;
  + atributo **itens** com @field:Size(min = 1) para garantir seu tamanho mínimo e com @field:Valid para habilitar a validação dos itens da coleção;
* Enquando na classe ItemDePedidoRequest eu a anoto com @Introspected e anoto seus atributos:
  + atributo **codigo** com @field:NotBlank;
  + atributo **preco** com @field:NotNull @field:Positive;
  + atributo **quantidade** com @field:NotNull @field:Positive;
* Ainda no controller, eu anoto a classe com @Validated e o parâmetro do método finalizar() com @Valid para que o Micronaut consiga disparar as validações de forma automática;
* Na classe NovoPedidoRequest eu crio um método toModel() para converter o DTO para uma instância de Pedido. Aproveito e faço o mesmo para a classe de ItemDePedidoRequest, que será convertida em ItemDePedido. Ainda no método toModel(), eu instancio Pedido recebendo os dados que vieram na request, onde converto a lista de ItemDePedidoRequest em uma lista de ItemDePedido através do método map() da coleção;
* Para preencher o atributo **cliente** do Pedido, basta instanciar a classe Cliente passando os atributos **nome** e **email** do DTO;
* Para preencher o atributo **total** do Pedido, eu crio um método getTotal() (ou property do Kotlin) na classe NovoPedidoRequest para calcular o valor total daquele pedido, que basicamente se resume em iterar pela lista de items somando o valor de cada item do pedido. Lembrando que o valor total de cada item de pedido se dá por meio da multiplicação dos atributos **preco** e **quantidade** existentes na classe ItemDePedidoRequest, por isso aproveito para encapsular essa lógica num método getTotal() na classe ItemDePedidoRequest;
* Agora, dentro do método finalizar() do controller, invoco o método toModel() do DTO da request para transformá-lo numa instância válida de Pedido;
* Por fim retorno um status HTTP 200 via HttpResponse.ok() ou simplesmente implemento o método finalizar() sem retorno;