

**Funcionalidade: Cadastro de país e estados**

|  |
| --- |
| **Cenário:** |
| Descreva os passos que você planeja executar para a implementação da funcionalidade. É muito importante que você descreva aqui antes da implementação em si. |

|  |
| --- |
| **O que seria bom ver nessa resposta?** |

* **Para o cadastro do país(total 5 pontos)**
* **Peso 0.8 :**Criação do controller e método mapeado para ser invocado via Post com os dados de um novo País configurado para validar entrada de dados e também para receber os dados como JSON.
* **Peso 1 :**Criação da classe Pais com suas devidas anotações de mapeamento Objeto Relacional.
* **Peso  0.8:**Utilização do construtor para receber o nome do país.
* **Peso 0.8:**Criaçãoda classe que representa a request de entrada com os dados do novo País.
* **Peso 0.4:**Utilização da Bean Validation na classe que representa a request de entrada
* **Peso 0.4:**Utilização da Bean Validation na classe de domínio.
* **Peso 0.8:**Utilização do EntityManager ou Repository para salvar o país.
* **Para o cadastro do país(total 5 pontos)**
* **Peso 0.5:**Criação do método para receber a requisição de um novo Estado anotado com a configuração para receber um POST, configurado para validar entrada de dados e também para receber os dados como JSON.
* **Peso 0.5:**Utilização de uma classe específica para receber os dados do novo Esatdo. O famoso DTO.
* **Peso 0.4:**Deve-se usar os id para referenciar o País no DTO
* **Peso 1**: Declaração dos outros atributos na classe
* **Peso 0.5:**Método de conversão dos dados do novo Estado para um Estado de fato dentro na própria classe DTO ou criação de uma classe de conversão de Dados. O método de conversão precisa ter acesso ao EntityManager/Repositories. Caso seja uma classe específica de conversão, ela precisa ter um atributo do tipo EntityManager ou os Repositories adequados.
* **Peso 0.5:**Utilização das annotations de validação da Bean Validation na classe do DTO
* **Peso 0.5:**Criação da classe que representa o Estado com as informações
* **Peso 0.4:**Precisa ter a referência ao atributo do tipo Pais
* **Peso 0.1:**Outros atributos
* **Peso 0.5:**Utilização das annotations de validação da Bean Validation na classe do Estado
* **Peso 0.5:**Utilização do construtor na classe Estado para deixar claro os dados que são obrigatórios
* **Peso 0.5:**Precisa receber parâmetros do tipo Pais no construtor aqui.
* **Peso 0.5:**Utilização da annotation @ManyToOne para mapear o relacionamento com Pais
* **Peso 0.5:**Utilização do EntityManager ou Repository específico para a inserção do Autor

|  |
| --- |
| **O que penaliza sua resposta?** |

* **Penalidade - 5:**Existência de Service no projeto. Não é necessária dada a especificação. Não tem nada lá que sugira a necessidade da flexibilidade que essa indireção traria.
* **Penalidade - 5:**Existência de biblioteca de geração de código compilado estilo Lombok. Não é necessário porque entendemos que o benefício trazido pelo código gerado não compensa o esforço de entendimento necessário que existe por trás de tal geração.
* **Penalidade -5:**Utilização de bibliotecas de mapeamento entre objetos tipo ModelMapper, MapStruct e afins.

|  |
| --- |
| **Resposta do Especialista:** |
| * **Para o cadastro de país** * Crio um controller novo * Coloco um método que vai receber os dados no novo país. Este método recebe um argumento do tipo NovoPaisRequest. Ali eu uso a annotation @RequestBody e @Valid para indicar que a informação vem no corpo da request e precisa ser validada. * Na classe NovoPaisRequest eu utilizo as validações da bean validation, inclusive a customizada que criei para barrar cadastro duplicado. * No método do controller agora, eu invoco o método toModel que criei na classe NovoPaisRequest para transformar aqueles dados de entrada num objeto do tipo Pais * Crio a classe Pais com o nome e id, recebendo o nome pelo construtor. Já que essa é uma informação obrigatória e tudo que é obrigatório e natural entra pelo construtor. * Faço as devidas anotações na classe para configurá-la para o Hibernate. Neste caso é básico com @Entity, @Id e @GeneratedValue * Utilizo o EntityManager para realizar a persistência lá no método do controller. Importante lembrar de deixar a anotação @Transactional no método para demarcar a transação. * **Para o cadastro de um Estado** * Crio um controller novo * Coloco um método que vai receber os dados no novo estado. Este método recebe um argumento do tipo NovoEstadoRequest. Ali eu uso a annotation @RequestBody e @Valid para indicar que a informação vem no corpo da request e precisa ser validada. * Na classe NovoEstadoRequest eu utilizo as validações da bean validation. Aqui não dá para usar a de valor duplicado que criei, já que é nome único dentro de um mesmo país. * Crio um validador novo pelo Spring Validator(poderia ser via BeanValidation também). Por conta deste validador eu crio um Repository para Estado de modo a usar as facilidades do SpringData JPA. Neste validador eu preciso acessar o objeto que representa a request de novo estado, acessar o nome e o id do estado. Com essas informações na mão, crio e invoco o método findByNomeAndPaisId no repositório do estado. Isso me retorna uma Optional. Agora eu olho se a optional está preenchida e, se estiver, quer dizer que já estado com aquele nome para aquele país. Rejeito a requisição. * Adiciono o validador lá no controller, para que ele possa fazer parte do fluxo de validação. * Por conta dessa validação, eu já crio a classe estado que tem um atributo que representa o nome e também um atributo do tipo Pais que representa o País. * Já coloco as devidas anotações. Lembrando aqui do @ManyToOne em cima do atributo do tipo País para ensinar o hibernate como se comportar na hora de mapear aquela classe para a tabela do banco de dados. * Outra coisa importante sobre a classe Estado é que o nome e país entram pelo construtor, já que são informações obrigatórias. * De volta ao método do controller, eu invoco o método toModel que criei na classe NovoEstadoRequest para transformar aqueles dados de entrada num objeto do tipo Estado. Aqui tem um detalhe especial, é necessário de novo receber o EntityManager ou Repository do pais para carregar o país em função do id do país que veio via request. * Utilizo o EntityManager para realizar a persistência lá no método do controller. Importante lembrar de deixar a anotação @Transactional no método para demarcar a transação. |