Exercícios Práticos: Projeto Livraria com Spring Boot e JPA

Este material didático foi elaborado para ajudá-lo a aprofundar seu conhecimento sobre o projeto da Livraria desenvolvido em Spring Boot, utilizando Spring Web, JPA e Lombok. Os exercícios abrangem desde o entendimento básico da estrutura até a manipulação de dados e a expansão do projeto.

---

1. Entendimento da Estrutura do Projeto

Objetivo: Compreender as responsabilidades de cada camada (Entity, Repository, Service, Controller) e como elas se comunicam.

Perguntas:

1. **Entidade (Livro.java):**
    - Qual é o propósito principal da classe Livro?
    - Para que servem as anotações @Entity, @Table e @Id?
    - Como o Lombok (@Data, @NoArgsConstructor, @AllArgsConstructor) contribui para esta classe?
2. **Repositório (LivroRepository.java):**
    - Qual o objetivo de estender JpaRepository<Livro, Long>?
    - Que tipos de operações este repositório é capaz de realizar sem que você escreva nenhum código específico para elas?
    - Se você precisasse buscar um livro pelo título, como você declararia um método para isso nesta interface?
3. **Serviço (LivroService.java):**
    - Por que existe uma camada de Serviço separada do Controller e do Repository? Qual é a sua responsabilidade principal?
    - O que significa a injeção de dependência via construtor (@Autowired) nesta classe?
    - Explique a lógica do método atualizarLivro. O que acontece se um livro com o id especificado não for encontrado?
4. **Controlador (LivroController.java):**
    - Para que serve a anotação @RestController?
    - Qual a função de @RequestMapping("/api/livros")?

- Diferencie o uso de @PostMapping, @GetMapping, @PutMapping e @DeleteMapping em relação aos métodos HTTP.
- Qual o papel de ResponseEntity nos métodos do controlador? Cite exemplos de códigos de status HTTP retornados.

---

2. Análise e Uso de Métodos

Objetivo: Analisar a implementação dos métodos em cada camada e entender seu fluxo de execução.

Perguntas:

1. **Fluxo de uma Requisição POST:**
   - Descreva passo a passo o que acontece quando uma requisição POST é feita para /api/livros para cadastrar um novo livro. Inclua todas as camadas envolvidas e os métodos chamados.
2. **Método buscarLivroPorId:**
   - No LivroService, o método buscarLivroPorId retorna um Optional<Livro>. Por que usar Optional?
   - Como o LivroController lida com o resultado Optional (quando o livro é encontrado e quando não é)?
3. **Tratamento de Erros:**
   - Observe os blocos try-catch nos métodos atualizarLivro e deletarLivro do LivroController. Qual o objetivo de capturar a RuntimeException?
   - Que tipo de código de status HTTP é retornado quando um livro não é encontrado para atualização ou exclusão?
4. **Diferença entre save e update:**
   - No LivroService, o método salvarLivro e atualizarLivro ambos chamam livroRepository.save(). Explique por que save() pode ser usado para ambas as operações (inserção e atualização) no JPA.

---

3. Extensão do Projeto

Objetivo: Propor melhorias e novas funcionalidades, aplicando os conceitos aprendidos.

Tarefas e Perguntas:

1. **Adicionar Campo de Data de Publicação:**
    - Como você adicionaria um novo campo dataPublicacao (do tipo LocalDate ou Date) à entidade Livro?
    - Que modificações seriam necessárias no DDL do PostgreSQL para incluir este campo?
    - Quais métodos no LivroService e LivroController precisariam ser ajustados para permitir o cadastro e atualização deste novo campo?
2. **Buscar Livros por Editora:**
    - Proponha como você implementaria uma funcionalidade para buscar todos os livros de uma determinada editora.
    - Quais seriam as alterações necessárias no LivroRepository, LivroService e LivroController para expor este novo endpoint? Defina o URL e o método HTTP.
3. **Paginação e Ordenação:**
    - Se a lista de livros se tornar muito grande, listar todos os livros de uma vez pode ser ineficiente. Como você modificaria o endpoint GET /api/livros para suportar paginação (ex: retornar 10 livros por página)?
    - Que parâmetro você esperaria na requisição para indicar a página e o tamanho da página? (Pesquise sobre Pageable no Spring Data JPA).
4. **Validação de Dados:**
    - Atualmente, não há validação no lado do servidor para os dados de entrada. Se você quisesse garantir que o campo título de um livro não seja nulo ou vazio, como você faria isso no Livro Entity e/ou no LivroService? (Pesquise sobre jakarta.validation.constraints).

---

4. Reflexão e Melhorias

Objetivo: Pensar criticamente sobre o design do projeto e possíveis otimizações.

Perguntas:

1. **DTOs (Data Transfer Objects):**
    - Atualmente, o projeto usa a própria entidade Livro para receber e retornar dados no Controller. Quais são os riscos e desvantagens dessa abordagem?
    - Como a introdução de DTOs (LivroRequestDTO, LivroResponseDTO) poderia melhorar a segurança, a flexibilidade e a clareza da sua API?
2. **Tratamento Global de Exceções:**
    - Perceba que a lógica de try-catch para lidar com RuntimeException está replicada em alguns métodos do LivroController. Como você centralizaria o tratamento de exceções em Spring Boot para evitar essa repetição? (Pesquise sobre @ControllerAdvice ou ResponseEntityExceptionHandler).

Observações:

- Utilize o código-fonte do projeto fornecido anteriormente para consultar e responder às perguntas.
- Para as questões de extensão e melhoria, pesquise na documentação do Spring Boot, Spring Data JPA e Jakarta Validation para encontrar as melhores práticas e implementações.
- Tente modificar e executar o projeto localmente para testar suas respostas e implementações.