

Logo da Orange Talents

# Demarcação de transação

## Antes de começar

Para complementar o curso, recomendamos estudar a documentação oficial do Micronaut sobre [Transactions](https://micronaut-projects.github.io/micronaut-data/latest/guide/#transactions) caso você não tenho estudado ainda. Nela é possível ter uma idéia mais clara do poder e das possibilidades oferecidas pelo framework.

Aqui é um momento importante de conectar vários conteúdos que estudamos e treinamos durante nossa jornada.

O que estou querendo dizer é que alguns conhecimentos sobre persistência de dados, controle transacional com Spring e bancos de dados relacionais que adquirimos através de alguns cursos da ALURA podem ser bastante úteis para este formulário, por exemplo:

* curso de [Spring Data JPA](https://www.alura.com.br/curso-online-spring-data-jpa), mais especificamente o tópico **Operações CRUD**;
* curso de [Spring Boot API REST: Construa uma API](https://www.alura.com.br/curso-online-spring-boot-api-rest), mais especificamente o tópico **Usando Spring Data**;
* curso de [Formação SQL com MySQL Server da Oracle](https://www.alura.com.br/formacao-oracle-mysql), mais especificamente o módulo **Avançando em manipulação de dados**;
* curso de [Persistência com JPA](https://www.alura.com.br/curso-online-persistencia-jpa-introducao-hibernate), mais especificamente o tópico **Ciclo de vida de uma entidade**;

Além disso, por o Micronaut ser inspirado no Spring Boot, seu controle transacional via @Transactional funciona da mesma maneira, portanto a documentação oficial do Spring sobre [Propagação de Transações](https://docs.spring.io/spring-framework/docs/current/reference/html/data-access.html#tx-propagation) possui conteúdo útil que pode ser facilmente transferido para o ecossistema do Micronaut.

## Cenário:

Temos um microsserviço responsável por adicionar comentários aos artigos publicados por nossos usuários. Para isso, o time de backend implementou a seguinte API REST com Micronaut, JPA/Hibernate e um banco de dados relacional:

@Validated  
@Controller  
class ArtigoController(@Inject val service: ArtigoService) {  
  
 @Transactional  
 @Post("/api/artigos/{id}/comentarios")  
 fun novoComentario(@PathVariable id: UUID, @Body @valid request: ComentarioRequest): HttpResponse<Any> {  
  
 val comentario = request.toModel()  
 if (isDuplicado(comentario)) {  
 return HttpResponse.unprocessableEntity() // status 422  
 }  
  
 service.comenta(id = id, comentario = comentario)  
 return HttpResponse.ok()  
 }  
  
 private fun isDuplicado(request: ComentarioRequest): Boolean {  
 // implementação não importante  
 }  
}  
  
@Singleton  
class ArtigoService(@Inject val repository: ArtigoRepository) {  
  
  
 @Transactional  
 fun comenta(id: UUID, comentario: Comentario) {  
  
 val artigo = repository.findById(id)  
 if (artigo == null) {  
 throw ArtigoNaoEncontradoException("Artigo não encontrado")  
 }  
  
 artigo.comenta(comentario)  
  
 // atualiza entidade no banco  
 repository.update(artigo)  
 }  
}  
  
@Repository  
interface ArtigoRepository : JpaRepository<Artigo, UUID> {  
  
}  
  
@Entity  
class Artigo(  
 @Id @GeneratedValue   
 val id: UUID,   
 @OneToMany(mappedBy = "artigo", cascade = CascadeType.ALL, ...)  
 val comentarios: MutableList<Comentario>,  
 // outros atributos  
) {  
  
 /\*\*  
 \* Adiciona novo comentário ao artigo  
 \*/  
 fun comenta(comentario: Comentario) {  
 comentario.artigo = this  
 this.comentarios.add(comentario)  
 }  
  
}  
  
@Entity  
class Comentario(  
 @Id @GeneratedValue   
 val id: Long,   
 @ManyToOne  
 var artigo: Artigo,  
 // outros atributos  
)  
  
class ArtigoNaoEncontradoException(message: String?) : RuntimeException(message) {  
   
}

Um novo desenvolvedor(a) junior entrou no time e ficou responsável por fazer alterações no código acima, mas ele(a) resolveu tirar algumas dúvidas com você por ser iniciante com Micronaut e JPA. Por esse motivo, com base no código acima, como você responderia as perguntas abaixo feitas pelo desenvolvedor(a) junior:

1. Quantas transações são abertas ao executar a API REST para adicionar um novo comentário no artigo? Explique sua resposta;
2. De acordo com a resposta anterior, quais classes e métodos são responsáveis por iniciar e comitar as transações? Explique sua resposta;
3. O que acontece com a transação quando o controller retorna o status de erro HTTP 422 (Unprocessable Entity)?
4. O que acontece com a transação quando uma exceção ArtigoNaoEncontradoException é lançada da classe service? Explique sua resposta;
5. Imagine que a anotação @Transactional seja removida do controller, a lógica para inserir um novo comentário continuaria funcionando ou algo quebraria? Explique sua resposta;
6. Imagine que o trecho de código repository.update(artigo) seja removido da classe service, o comentário ainda seria inserido corretamente no banco de dados? Explique sua resposta;