algaworks

Conteúdo do treinamento

# ESPECIALISTA JPA

TURMA DO LANÇAMENTO
FEVEREIRO/2020

## Mapeamento do básico ao super avançado

O JPA é uma especificação de persistência de dados bem completa e que ajuda bastante o desenvolvedor Java.

O seu funcionamento é possível graças às inúmeras anotações de mapeamento objeto-relacional disponíveis.

Existem anotações para mapear chaves primárias, relacionamentos, enumerações, datas, etc. Sem contar as configurações que cada anotação pode receber.

Este curso vai te ajudar a lidar com as principais anotações e os mais diversos tipos de mapeamentos.

Você vai aprender a melhor maneira de mapear suas entidades, desde anotações mais básicas, como a @Column, até as mais avançadas, como @JoinTable, @SecondaryTable, @IdClass, @MapsId, @Inheritance, etc.

## Relacionamentos entre entidades

Você vai aprender a mapear todos os tipos de relacionamentos entre entidades e usar as anotações @ManyToOne, @OneToMany, @ManyToMany e @OneToOne.

A maior parte dos programadores Java que estão aprendendo ou já trabalham com JPA tem um nível mais elevado de dificuldade neste ponto.

No curso você irá aprender a usar cada uma dessas anotações e descobrirá técnicas que vão te ajudar a decidir quando usar cada uma.

E ainda vai aprender como encontrar o *owner* e *non-owner* de uma relação, pois você vai precisar saber isso para entender o comportamento do JPA e utilizá-lo da melhor forma, sem imprevistos na sua aplicação.

## Ciclo de vida e os estados

Você vai aprender o ciclo de vida das entidades mapeadas com JPA. Isso é algo importante porque o comportamento do JPA em uma entidade muda de acordo com o estado atual que o objeto se encontra.

Você vai aprender a identificar quando uma instância de uma entidade é considerada transiente, persistente, removida ou desanexada.

Existem operações que dependem do estado da entidade para funcionar. Sem esse conhecimento, comportamentos inesperados vão acontecer e serão difíceis de identificar.

## Gerenciamento de transações

No curso você vai aprender o que é uma transação e qual o objetivo dela dentro da sua aplicação. Vai descobrir também como fazer para cancelar uma transação, caso seja necessário.

Você vai aprender a gerenciar uma transação manualmente, pois é algo essencial para entender de verdade como funciona, e também vai aprender como isso é tratado nas aplicações web profissionais, de forma automatizada.

## Eventos e callbacks

Cada vez que é realizada alguma operação com uma entidade, seja uma inserção, atualização ou pesquisa, um evento do JPA é disparado.

Você vai aprender a "escutar" e reagir a esses eventos no seu código Java através de uma nova classe.

Vai aprender também a configurar métodos de callback que serão baseados nos eventos que você escolher trabalhar.

## Estratégias para chave primária simples e composta

A maioria das tabelas em um banco de dados tem uma chave primária simples, e por isso no treinamento você vai entender as estratégias possíveis para se usar nesse caso.

Vai aprender a usar recursos de autoincremento do banco de dados, usar uma *sequence* e também gerenciar os valores da sua chave primária a partir de uma tabela específica.

Além disso, você também vai aprender como mapear chave primária composta, que possui mais de uma coluna, pois também é um cenário comum no desenvolvimento de software.

## Chave primária e estrangeira com a mesma coluna

É comum também acontecer de termos uma chave primária que usa colunas da tabela que, ao mesmo tempo, também são chaves estrangeiras.

Muitas das vezes precisamos usar essa coluna para mapear nosso atributo que representará o identificador do banco de dados e também para mapear o atributo que fará um relacionamento com outra entidade.

Para fazer esse mapeamento da melhor forma, é necessário dominar as anotações de relacionamento, anotações de chave primária e a anotação @MapsId, que você também aprenderá no curso.

## Arquivos no banco de dados

Em alguns cenários, você pode precisar salvar arquivos em uma tabela do banco de dados. O JPA dá suporte para fazer isso através da anotação @Lob.

No curso você vai aprender a usar essa anotação e salvar uma imagem no banco. E claro, também vai aprender a consultar e recuperar os dados do arquivo.

## Tabela secundária de uma mesma entidade

Às vezes pode acontecer de você ter duas tabelas representadas por uma única entidade. Ou seja, você faz o mapeamento de uma entidade, porém possui duas tabelas onde estão armazenados os dados.

Isso pode acontecer por uma questão de normalização do banco de dados ou talvez por estar lidando com um banco legado.

Nesse cenário, você vai precisar conhecer a anotação @SecondaryTable, que irá aprender também no curso, incluindo seus ajustes específicos.

## Estratégias de herança

Herança é um recurso que sempre devemos usar com cuidado na aplicação. Não só com relação ao JPA, mas em nosso sistema como um todo.

O legal é que, nos casos onde ela é realmente necessária, o JPA dá suporte para que isso aconteça.

É possível configurar a classe pai para uma tabela específica e as filhas para outras. É possível também mapear a classe pai e as classes filhas para uma mesma tabela.

Você vai escolher a estratégia de acordo com o modelo que desejar para o seu banco de dados.

No curso você vai aprender a mapear herança usando todas as estratégias detalhadamente (*Single Table*, *Table per Class* e *Joined)*.

E caso você queira utilizar herança de forma que não haja impacto no modelo do banco de dados, também é possível. Você vai aprender a mapear herança com a anotação @MappedSuperclass também.

## Banco de dados legado

Uma hora ou outra você vai precisar construir um sistema novo, porém com um banco de dados legado.

Em casos como esse, pode acontecer do modelo das tabelas e relacionamentos do banco legado não usar estruturas convencionais.

O conhecimento que o curso entrega pra você vai te permitir lidar com esse problema, pois só um programador especialista em JPA vai conseguir criar e mapear as entidades da melhor forma, sem gambiarras.

## Schema generation

Não é uma boa prática usar o JPA para gerar o schema do seu banco de dados de produção, mas esse recurso pode ser útil, especialmente se você estiver no estágio inicial de desenvolvimento ou criando um protótipo.

Mas para o DDL do schema ser gerado corretamente, é preciso configurar os detalhes físicos nas anotações do JPA.

Você vai aprender tudo isso no curso!

## Operações em cascata

Com os recursos de operação em cascata você tem a possibilidade de, com um só comando, executar operações em uma ou mais entidades.

Mas para usar da melhor forma, é preciso ter conhecimento sobre o ciclo de vida das entidades e também sobre como funcionam os relacionamentos.

Como o curso vai te ensinar tudo que precisa, você vai conseguir tirar o melhor proveito possível das operações em cascata.

## JPQL e Criteria API do básico ao avançado

O JPA nos entrega muito poder também na hora de criar nossas consultas no banco de dados. A começar por disponibilizar duas formas para criação das mesmas, a JPQL e Criteria API.

As duas formas são muito poderosas e praticamente tudo que pode ser feito de uma forma, poderá ser executado com a outra também.

No treinamento você vai aprender como criar consultas do básico ao avançado, das duas formas! Vai aprender a executar de consultas simples até as mais complexas.

No meio disso, vai aprender sobre expressões, joins, fetch joins, passagem de parâmetros, operadores lógicos, vários tipos de funções, como agrupar registros com *group by* e muito mais!

## Subqueries com JPQL e com Criteria API

Subqueries é um recurso que aumenta muito as possibilidades para consultamos os dados em nosso banco.

Você pode entender isso como uma consulta dentro de outra. Elas são utilizadas quando é preciso gerar uma informação que ainda não está explícita em nosso banco de dados.

Dominar as subqueries no JPA vai deixar suas pesquisas em outro nível. Por entender isso é que incluímos várias aulas sobre esse tema dentro do curso. Para você aprender tudo que precisa para planejar suas consultas da melhor maneira.

## Operações em lote

Em alguns casos, precisamos executar uma mesma operação em vários registros diferentes da nossa base.

No curso você vai aprender a usar tanto a JPQL quanto a Criteria API para remover ou atualizar vários objetos de uma vez só, com ganho de performance.

## Named queries e arquivos XML

Uma named query é uma consulta em JPQL que não pode ser alterada. Ela pode tanto ser colocada em anotações (presentes nas entidades) quanto em um arquivo XML.

A vantagem é que ela é pré-compilada pela implementação do JPA que você estiver usando, possibilitando assim uma melhor performance da aplicação.

Sem contar que deixar as consultas em arquivos XML pode te ajudar a organizar melhor sua aplicação.

E você vai aprender como fazer isso no curso!

## Metamodel

Esse é um recurso para ser utilizado com a Criteria API. O metamodel foi criado com a intenção de deixar o nosso código mais seguro, sendo possível descobrir problemas em nossas consultas em tempo de compilação.

Basicamente, o metamodel são classes que ajudarão na criação das consultas. Para cada entidade dentro do seu projeto, você terá uma classe de metamodel correspondente.

No curso você vai aprender a gerar o metamodel usando Apache Maven, para que fique independente da IDE que estiver usando no desenvolvimento.

## SQL nativo, views, procedures e functions

Você pode precisar executar um SQL nativo no seu banco de dados por algum motivo. Talvez porque precise usar um recurso específico dele.

A boa notícia é que o JPA nos dá suporte para isso. A gente consegue executar um SQL normalmente, executar views, procedures e functions.

O legal é que você vai aprender tudo isso, e com detalhes, dentro do curso.

## Validação com Bean Validation

Bean Validation é uma especificação para validação de objetos que se integra muito bem com o JPA.

Você vai aprender sobre as anotações de validação mais utilizadas que vão te ajudar a não deixar passar informações incorretas para seu banco de dados.

## Pool de conexões

Você vai aprender a usar um gerenciamento profissional do seu pool de conexões com o HikariCP, que vem sendo a biblioteca mais utilizada para esse objetivo.

Em grandes aplicações é crucial a configuração de um pool profissional, para que você consiga a melhor performance da sua aplicação com o banco de dados.

## Entity Graph

Esta é uma parte da API do JPA que vai te ajudar a configurar como você deseja o retorno da sua consulta.

Ela te permite escolher os atributos da sua entidade que você deseja que sejam retornados na consulta.

Você vai aprender no treinamento como usar o Entity Graph e também sobre algumas limitações que existem quanto a essa funcionalidade.

## Cache de segundo nível (ou cache compartilhado)

Este é outro recurso que, se bem utilizado, pode ajudar muito a sua aplicação a diminuir o tempo de resposta dos usuários.

Mais conhecido pelos nomes em inglês que são *Second Level Cache* ou *Shared Cache*, você vai aprender todos detalhes sobre esse recurso do JPA.

Vai aprender como incluir suas entidades no cache, como o cache pode se manter o mais atualizado possível e como remover as entidades do cache. Além disso, vai aprender a configurar uma implementação de cache profissional com o EhCache.

## Concorrência, Lock Otimista e Lock Pessimista

É comum surgirem situações onde duas transações diferentes irão alterar o mesmo objeto e ao mesmo tempo, principalmente em aplicações com muitos acessos.

Existem duas formas de tratarmos isso. Basicamente, com dois tipos de lock (ou trava), que são o Lock Otimista e Lock Pessimista.

O Lock Otimista é menos agressivo e fácil de ser implementado. Ele já poderia resolver problemas de concorrência na maioria das situações.

O Lock Pessimista já é algo mais avançado e serve para situações mais críticas e bem específicas de concorrência.

Estes dois tipos você encontrará no curso e com exemplos bem didáticos, pois não é um assunto simples. Por isso tratamos eles de forma detalhada.

## Multitenancy (Multi-inquilinos)

Hoje em dia é muito comum desenvolver uma aplicação e disponibilizar a mesma instância dela para todos os clientes. Isso é conhecido como SaaS (*Software as a Service, ou Software como um Serviço*).

Tecnicamente, para isso se tornar possível e não correr o risco de misturar os dados dos clientes, precisamos implementar o suporte a multitenancy (multi-inquilinos) na aplicação.

Existem várias abordagens de multitenancy para separar os dados dos inquilinos, como abordagem por máquina (banco), por schema e por coluna (identificador).

O JPA não tem suporte a esse recurso, mas no curso você vai aprender a implementar essas 3 abordagens com o suporte que a implementação do Hibernate possui e também de forma customizada/manual (para quando o Hibernate não dá nenhum suporte).

## Hibernate e EclipseLink

Durante o curso nós vamos usar o Hibernate como implementação do JPA, porque é a mais usada no mercado e muito madura.

Mas você vai aprender a substituir a implementação pelo EclipseLink também, que é a implementação de referência do JPA.

Esse tipo de alteração não acontece de forma 100% transparente, como esperaríamos ser. Precisamos fazer pequenos ajustes para o EclipseLink funcionar.

Mas não se preocupe, você vai aprender isso nas aulas.

## MySQL e PostgreSQL

Durante as aulas do curso, vamos usar o MySQL como banco de dados, por ser muito popular entre todos os tipos de desenvolvedores e muito confiável para aplicações em produção.

Mas pode ser que você queira mudar o sistema de gerenciamento de banco de dados para algum outro, já que o JPA permite isso.

E está tudo bem, você vai aprender a substituir para um banco PostgreSQL. Logicamente, você poderia substituir por qualquer outro banco de dados relacional suportado pelo JPA.

## JPA em uma aplicação web

Para fechar com chave de ouro, você vai ver como o JPA se comporta em uma aplicação web.

Vai conseguir identificar como utilizar cada recurso do JPA, que aprendeu no curso, dentro de uma aplicação web profissional.

Vai ver o JPA funcionando tanto em uma aplicação web com Jakarta EE quanto também em uma aplicação web com Spring, e ainda vai aprender a usar o data source do servidor de aplicações Wildfly.

# Conteúdo programático

**1. Introdução**
1.1. Boas-vindas e como fazer o curso
1.2. Instalando o MySQL Server e o MySQL Workbench
1.3. O que é persistência de dados?
1.4. Criando tabelas e persistindo dados pelo MySQL Workbench
1.5. Usando o MySQL Client
1.6. Instalando o JDK
1.7. Instalando a IDE IntelliJ IDEA
1.8. Importando o projeto do GitHub
1.9. Entendendo e configurando o JUnit
1.10. Acessando o banco de dados com JDBC
1.11. O que é JPA?
1.12. O que é Mapeamento Objeto-Relacional (ORM)?
1.13. Jakarta Persistence vs Java Persistence API

**2. Iniciando com JPA**
2.1. Criando um projeto com Apache Maven
2.2. Mapeando a primeira entidade com JPA
2.3. Criando o persistence.xml
2.4. Criando o EntityManager
2.5. Montando a classe para testes com o JUnit
2.6. Buscando objetos por identificador
2.7. Criando uma classe genérica para testes
2.8. Abrindo e fechando uma transação
2.9. Inserindo o primeiro objeto com o método persist
2.10. Removendo objetos com o método remove
2.11. Atualizando objetos com o método merge
2.12. Atualizando objetos gerenciados
2.13. Adicionando objetos com o método merge
2.14. Entendendo a diferença entre os métodos persist e merge
2.15. Exercício: implementando um CRUD
2.16. Desanexando objetos do contexto de persistência com o método detach
2.17. Conhecendo e usando Lombok

**3. Mapeamento básico**
3.1. Conhecendo o modelo de domínio do projeto e criando as entidades
3.2. Mapeando as entidades e customizando os nomes das tabelas e colunas
3.3. Exercício: mapeando a classe Pedido
3.4. Entendendo a diferença entre mapear atributos ou métodos
3.5. Mapeando enumerações com @Enumerated
3.6. Mapeando objetos embutidos com @Embeddable
3.7. Conhecendo as estratégias para geração de identificador com @GeneratedValue
3.8. Configurando a geração de identificador com @SequenceGenerator
3.9. Configurando a geração de identificador com @TableGenerator

3.10. Configurando geração de identificador com a estratégia IDENTITY
3.11. Exercício: corrigindo classes de testes

**4. Mapeamento de relacionamentos**
4.1. Conhecendo os tipos de relacionamentos entre entidades
4.2. Mapeando relacionamentos muito-para-um com @ManyToOne
4.3. Exercício: mapeando relacionamentos muitos-para-um
4.4. Mapeando relacionamentos um-para-muitos com @OneToMany
4.5. Exercício: mapeando relacionamentos um-para-muitos
4.6. Mapeando autorelacionamentos com @ManyToOne e @OneToMany
4.7. Removendo objetos referenciados por outras entidades
4.8. Mapeando relacionamentos muitos-para-muitos com @ManyToMany e @JoinTable
4.9. Mapeamento relacionamentos um-para-um com @OneToOne
4.10. Exercício: mapeando relacionamentos um-para-um
4.11. Mapeando relacionamentos um-para-um com @JoinTable
4.12. Entendendo o funcionamento de Eager e Lazy Loading
4.13. Para o que serve o atributo optional?
4.14. Exercício: usando o atributo optional

**5. Conhecendo o EntityManager**
5.1. Estados e ciclo de vida dos objetos
5.2. Entendendo o cache de primeiro nível
5.3. Gerenciamento de transações
5.4. Funcionamento do método flush
5.5. Contexto de persistência e o dirty checking
5.6. Callbacks para eventos do ciclo de vida
5.7. Listeners para eventos do ciclo de vida

**6. Mapeamento avançado**
6.1. Conhecendo detalhes da anotação @Column
6.2. Exercício: anotação @Column
6.3. Mapeando chave composta com @IdClass
6.4. Exercício: usando @IdClass
6.5. Mapeando chave composta com @EmbeddedId
6.6. Mapeando chave primária e estrangeira na mesma coluna com @MapsId
6.7. Exercício: usando @MapsId
6.8. Declarando propriedades transientes com @Transient
6.9. Mapeando coleções de tipos básicos com @ElementCollection
6.10. Mapeando coleções de objetos embutidos com @ElementCollection
6.11. Mapeando mapas com @ElementCollection
6.12. Mapeando e persistindo dados de arquivos com @Lob
6.13. Exercício: persistindo fotos de produtos
6.14. Mapeando tabela secundária com @SecondaryTable
6.15. Mapeando herança com @MappedSuperclass
6.16. Entendendo a diferença entre estender uma entidade abstrata e usar a anotação @MappedSuperclass
6.17. Mapeando herança com estratégia de tabela única (single table)
6.18. Mapeando herança com estratégia de uma tabela por classe (table per class)

6.19. Mapeando herança com a estratégia Joined Table
6.20. Exercício: voltando o mapeando de herança para tabela única

**7. Mapeando entidades para geração do DDL**
7.1. Quando criar o schema do banco usando JPA?
7.2. Configurando detalhes da tabela com @Table
7.3. Exercício: usando @Table
7.4. Configurando colunas com @Column
7.5. Exercício: usando @Column
7.6. Corrigindo os testes do JUnit
7.7. Usando a anotação @Lob em strings
7.8. Configurando chaves estrangeiras com @JoinColumn
7.9. Exercício: usando @JoinColumn
7.10. Entendendo alguns detalhes de @JoinTable
7.11. Configurando tabelas secundárias com @SecondaryTable
7.12. Conhecendo as estratégias de schema generation
7.13. Gerando o schema do banco com arquivos de scripts SQL
7.14. Gerando o schema do banco com metadados e scripts
7.15. Exportando os scripts de schema generation para arquivos externos
7.16. Configurando propriedades da unidade de persistência dinamicamente para schema generation

**8. Operações em cascata**
8.1. Configurando operações em cascata
8.2. Fazendo inserções de objetos em cascata
8.3. Exercício: fazendo inserções em cascata
8.4. Fazendo atualizações em cascata
8.5. Exercício: fazendo atualizações em cascata
8.6. Fazendo remoções em cascata
8.7. Entendendo a remoção em cascata com @ManyToMany
8.8. Removendo objetos órfãos com a propriedade orphanRemoval
8.9. Quando configurar operações em cascata?

**9. JPQL do básico ao avançado**
9.1. Introdução à JPQL (Java Persistence Query Language)
9.2. Entendendo as diferenças entre TypedQuery e Query
9.3. Selecionando um atributo da entidade como retorno da consulta
9.4. Trabalhando com projeções
9.5. Trabalhando com projeções e DTO
9.6. Fazendo inner join entre as entidades
9.7. Usando left outer join
9.8. Fazendo o join e usando o fetch
9.9. Entendendo as Path Expressions
9.10. Exercício: consultando pedidos com produto específico
9.11. Passando parâmetros para as consultas
9.12. Usando expressão condicional like
9.13. Usando expressões condicionais is null e is empty
9.14. Usando expressões condicionais de maior e menor

9.15. Exercício: usando expressões de maior e menor com datas
9.16. Usando expressão condicional between
9.17. Usando expressão de diferente
9.18. Usando operadores lógicos
9.19. Ordenando os resultados da consulta
9.20. Fazendo paginação de resultados
9.21. Limitando a quantidade de registros retornados
9.22. Usando funções para strings
9.23. Usando funções para datas
9.24. Usando funções para números
9.25. Usando funções para coleções
9.26. Usando funções nativas
9.27. Usando funções de agregação
9.28. Agrupando o registros com group by
9.29. Exercício: usando group by
9.30. Usando a cláusula where com group by
9.31. Usando o having para condicionar o agrupamento
9.32. Usando a expressão case
9.33. Usando a expressão in
9.34. Usando o distinct para evitar duplicações
9.35. Criando subqueries
9.36. Criando subqueries com a expressão in
9.37. Criando subqueries com a expressão exists
9.38. Exercício: usando a expressão in
9.39. Exercício: criando subqueries
9.40. Exercício: criando subqueries com exists
9.41. Criando subqueries com all
9.42. Criando subqueries com any
9.43. Exercício: criando subqueries com all
9.44. Fazendo operações em lote
9.45. Atualizando objetos em lote
9.46. Removendo objetos em lote
9.47. Configurando uma dynamic query
9.48. Configurando uma query nomeada com @NamedQuery
9.49. Externalizando queries em um arquivo XML
9.50. Abordagem híbrida para dynamic e named queries

**10. Criteria API do básico ao avançado**
10.1. Introdução à Criteria API do JPA
10.2. Selecionando um atributo da entidade como retorno da consulta
10.3. Exercício: retornando todos os produtos
10.4. Trabalhando com projeções
10.5. Usando tuple para uma projeção
10.6. Trabalhando com projeções e DTO
10.7. Fazendo inner join entre as entidades
10.8. Usando a cláusula on no join
10.9. Usando left outer join
10.10. Fazendo o join e usando o fetch

10.11. Consultando pedidos com um produto específico
10.12. Passando parâmetros para as consultas
10.13. Tipagem forte com metamodel
10.14. Usando expressão condicional like
10.15. Usando as expressões condicionais is null e is empty
10.16. Usando expressões condicionais de maior e menor
10.17. Exercício: usando expressões de maior e menor com datas
10.18. Usando expressão condicional between
10.19. Usando expressão de diferente
10.20. Usando operadores lógicos
10.21. Ordenando os resultados da consulta
10.22. Fazendo paginação e limitando resultados
10.23. Usando funções para string
10.24. Usando funções para datas
10.25. Usando funções para números
10.26. Usando funções para coleções
10.27. Usando funções nativas
10.28. Usando funções de agregação
10.29. Agrupando registros com o group by
10.30. Exercício: usando group by
10.31. Diferença entre expressions, paths e predicates
10.32. Exercício: consultando pedidos com produto específico
10.33. Agrupando registros com funções no group by
10.34. Usando o having para condicionar o agrupamento
10.35. Usando a expressão case
10.36. Usando a expressão in
10.37. Usando distinct para evitar duplicações
10.38. Criando subqueries
10.39. Relacionando a subquery com a query principal
10.40. Criando subquery com a expressão in
10.41. Criando subquery com a expressão exists
10.42. Exercício: criando subqueries
10.43. Exercício: criando subqueries com in
10.44. Exercício: criando subqueries com exists
10.45. Criando subqueries com all
10.46. Criando subqueries com any
10.47. Exercício: criando subqueries com all
10.48. Atualizando objetos em lote
10.49. Removendo objetos em lote

**11. Consultas nativas**
11.1. Por que usar query nativa?
11.2. Executando SQL e retornando uma lista de arrays
11.3. Executando SQL e retornando uma entidade
11.4. Passando parâmetros para consulta nativa
11.5. Mapeando resultado de queries nativas com @SqlResultSetMapping
11.6. Usando @SqlResultSetMapping com @FieldResult
11.7. Usando @SqlResultSetMapping com @ColumnResult e retornando DTO

11.8. Usando uma @NamedNativeQuery
11.9. Adicionando consultas no arquivo XML
11.10. Exercício: mapeando retorno para um DTO
11.11. Invocando stored procedures com parâmetros in e out
11.12. Recebendo uma lista de registros da procedure
11.13. Exercício: atualizando registros com procedures
11.14. Configurando uma procedure com a anotação @NamedStoredProcedureQuery
11.15. Invocando uma view do banco de dados

**12. Bean Validation, pool de conexões, Entity Graph e detalhes avançados**
12.1. Validando objetos com Bean Validation
12.2. Exercício: validando objetos
12.3. Analisando anotações utilizadas
12.4. Entendendo o Pool de Conexões
12.5. Usando o HikariCP como gerenciador do pool de conexões
12.6. Buscando conexões de um nome JNDI
12.7. Criando um conversor de atributo
12.8. O problema do @OneToOne com o lazy no Hibernate
12.9. Entendendo e configurando um Entity Graph
12.10. Adicionando um Subgraph na consulta
12.11. Utilizando metamodel com Entity Graph
12.12. Configurando o Entity Graph através da anotação @NamedEntityGraph
12.13. Ajustando a configuração da entidade Pedido
12.14. Resolvendo o problema do N+1

**13. Second Level Cache (cache compartilhado)**
13.1. Entendendo o cache de segundo nível (ou shared cache)
13.2. Incluindo as entidades no cache
13.3. Removendo entidades do cache
13.4. Verificando se uma entidade está no cache
13.5. Modos de cache e a anotação @Cacheable
13.6. Fazendo controle dinâmico do cache
13.7. Configurando o EhCache como provedor
13.8. Fechando as instância de EntityManager dos exemplos de cache

**14. Concorrência e locking**
14.1. O que é concorrência e início da configuração dos exemplos
14.2. Resolvendo problemas de concorrência com Lock Otimista
14.3. Tipos que o atributo com @Version pode ter
14.4. Entendendo a diferença entre Lock Otimista e Lock Pessimista
14.5. Fazendo Lock Pessimista com PESSIMISTIC_READ
14.6. Fazendo Lock Pessimista com PESSIMISTIC_WRITE
14.7. Entendendo o que acontece se misturarmos mais de um tipo de lock
14.8. Outros tipos de lock
14.9. Lock com JPQL e Criteria API

**15. Multitenancy**
15.1. O que é Multitenancy (ou Multitenant) e os tipos de abordagem

15.2. Alteração na classe EntityManagerTest para melhorar a organização dos testes
15.3. Criando um novo schema no banco de dados
15.4. Implementando multitenancy com abordagem por schema
15.5. Implementando multitenancy com abordagem por máquina
15.6. Analisando uma aplicação web com Multitenant
15.7. Implementando multitenancy por coluna em uma aplicação web

**16. PostgreSQL e EclipseLink**
16.1. Instalando o PostgreSQL
16.2. Alterando as configurações para usar JPA com PostgreSQL
16.3. Alterando as configurações para usar EclipseLink como implementação do JPA

**17. JPA em aplicações web**
17.1. Reconhecendo o que aprendemos de JPA dentro de uma aplicação web
17.2. Configurando um projeto web com Spring MVC
17.3. Entendendo o JPA em um projeto com Spring MVC
17.4. Entendendo a camada de persistência
17.5. Configurando um projeto web com KumuluzEE
17.6. Entendendo o JPA em um projeto com KumuluzEE
17.7. Transações com RESOURCE_LOCAL vs JTA
17.8. Baixando e configurando o JBoss Wildfly
17.9. Publicando o projeto no JBoss Wildfly
17.10. Conclusão do curso e próximos passos