02. Configurações e EntityManager

Date	@15/12/2022
Categoria	Java e persistência
Curso	Persistência com JPA: Hibernate

Tópicos

- Arquivo persistence.xml
- Persistence Unit
- Mapeando uma entidade
- Entidades da JPA
- Persistindo uma entidade
- Transação
- Faça como eu fiz
- O que aprendemos?

Persistence Unit

Qual objetivo da tag `<persistence-unit>` no arquivo `persistence.xml`?

- Agrupar as configurações de uma *unidade de persistência*, que representa um banco de dados utilizado pela aplicação.

Mapeando uma entidade

Algumas anotações que foram utilizada nessa parte da aula:

- `@Entity`
    - Então, em cima da classe, podemos colocar uma anotação da JPA que é o `@Entity`. Assim, é como se disséssemos: JPA, está vendo essa classe `Produto`? Ela é uma entidade, ou seja, existe uma tabela no banco de dados que está mapeando, e que é o espelho dessa classe. Então, é para isso que serve essa anotação `@Entity`.
- `@Table`
    - Eventualmente, se o nome da tabela não for o mesmo da entidade, teremos que ensinar isso para a JPA, porque, por padrão, ela considera que o nome da tabela é o mesmo nome da entidade (no nosso caso, não é). Para fazer essa configuração, adicionaremos mais uma anotação em cima da classe que é o `@Table`. Apertaremos "Ctrl + Shift + O" para importar e, de novo, selecionaremos `javax.persistence.Table`.

      Na anotação `@Table`, abriremos parênteses, selecionaremos o atributo `name:String - Table`com a qual passaremos o nome da tabela que é `name = "produtos"`.
- `@Column`
    - Uma curiosidade é que o nome dos atributos é exatamente igual ao nome das colunas no banco de dados. Logo, isso é algo que não precisaremos ensinar para a JPA, ela já assume que o nome da coluna é o mesmo do atributo dentro da entidade. Se fosse diferente, isto é, se o nome da coluna "descricao" fosse "desc", por exemplo, como ensinaríamos para a JPA caso não quiséssemos chamar o atributo de desc e, sim, de "descricao"?

      Neste caso, nós colocaríamos, em cima do atributo, uma anotação chamada `@Column`(e apertaríamos "Ctrl + Shift + O" para importar). Da mesma maneira, existe uma atributo chamado `name`, seguido dele, passaríamos o nome da coluna no banco de dados `"desc"`. Ou seja, `"Column(name = "desc")`. É como se disséssemos para a JPA: o nome do atributo é `descricao`, mas o nome da coluna, `@Column`, é `desc`.
- `@Id`
    - Só temos mais um detalhe importante para a JPA. No banco de dados, a coluna "id" é a chave primária. Nós precisamos informar qual é a "*primary key*", a chave primária da tabela no mundo relacional. Também precisamos informar para a JPA que, dos quatro atributos, o primeiro, que se chama `id`, é a chave primária, já que ele não associa automaticamente.

Em cima do atributo `id`, colocaremos uma notação chamada `@Id` e apertamos "Ctrl + Shift + O" para importar. No nosso caso, ele importou diretamente do `javax.persistence.Id`. Como, geralmente, quem cuida do `id`, da chave primária é o banco de dados e não a aplicação, também precisamos ensinar para a JPA que quem gerará o identificador não é a aplicação e, sim, o banco de dados.

- `@GeneratedValue`
  - Quando formos salvar um produto, o `id` estará nulo. Não tem problema, porque é o banco de dados que vai gerar o próximo `id`. Podemos configurar isso com outra notação, que colocamos em cima do atributo `id`, que é o `@GeneratedValue`, isto é, para dizer como o valor da chave primária é gerado.

Entidades da JPA

Qual a melhore definição de uma entidade JPA?

- É uma classe que faz o mapeamento de uma tabela do banco de dados.
  - Uma entidade JPA funciona como um *espelho* de uma tabela no banco de dados

---

Transação

Quando devemos iniciar e *comitar* uma transação ao persistir uma entidade?

- Ao realizar operações de *escrita* no banco de dados, como `insert`, `update` e `delete`

O que aprendemos?

Nessa aula, você aprendeu:

- Como configurar a JPA via arquivo `persistence.xml`;
- Como mapear entidades JPA;

- Como utilizar o `EntityManager` para persistir entidades no banco de dados.
