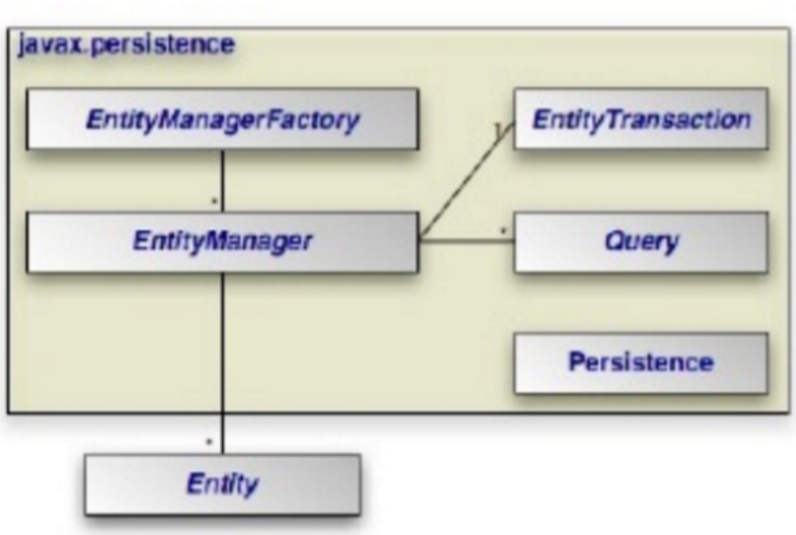
JPA API

quinta-feira, 8 de julho de 2021 19:22

O HIBERNATE
O Hibernate (servidor JPA)abstrai o código SQL, toda a camada JDBS e o SQL será gerado em tempo de execução, além de gerar um SQL serva para um determinador banco de dados, já que a cada banco fala um "dialeto" diferente dessa linguagem. Possibilita ainda trocar de banco de dados sem ter que alterar o código JAVA.

Implementação da JPA



Para a JPA saber qual classe ela terá de tratar, isto é, realizar o mapeamento objeto relacional, é utilizado anotações.
Ao criarmos uma classe VO temos de implementar a interface Serializable e utilizar a anotação @Entity para especificar a classe que será persistida. É utilizado uma anotação opcional @Table(name="nomedatabela") que define o nome da tabela do banco de dados, caso não seja utilizada a JPA ira criar a tabela com o nome da classe.
Para definir qual será a chave primaria da entidade, basta utilizar a anotação @id e para especificar que o valor da chave primaria será gerado automaticamente usa-se a anotação @GeneratedValue (strategy = GenerationType.SEQUENCE), existe dois tipos de valores para gerar a chave primaria, o SEQUENCE(como o nome induz, irá gerar a chave em sequencia) e o outro é a utilização de uma tabela para todas as chaves primarias geradas automaticamente.

CLASSES IMPORTANTES DA JPA

```
EntityManagerFactory emf = null;
EntityManager em = null;
```

PERSISTENCE: usada para criar fábricas de gerenciadores de entidades e contexto de persistência**;** (Gera fábrica de gerenciador de entidade)

```
emf = Persistence.createEntityManagerFactory("UnidadeBDPostgre");
```

EntityManagerFactory: Representa os objetos de fábrica de gerenciadores - usada para criar os gerenciadores de entidades. (A fabrica gera o gerenciador de entidade)

```
em = emf.createEntityManager();
```

EntityManager: representa os objetos manipulados de entidades e transações. Objeto que realiza as operações de CRUD, ou seja realmente persiste o objeto.(entidade)

Persistência em Objeto

INCLUSAO

A persistência de um objeto é a inserção de dados no banco de dados para atributos do objeto.
Para a realização é necessário instanciar o objeto e através da solicitação de informações ao usuário, setar esses dados no objeto.

```
grupoVO.setNome(nome);
grupoVO.setMargemLucro(margem);
grupoVO.setPromocao(promocao);
```

Após setar os dados, é necessário que se crie um EntityManagerFactory usando as configurações da unidade de persistencia especificada,ou seja, solicita-se que crie uma fabrica de gerenciadores de entidades usando a unidade de pesquistencia.

```
emf = Persistence.createEntityManagerFactory("UnidadeBDPostgre");
```

A partir do emf, cria-se um objeto gerenciar de entidade para realizar as transações coma s entidades.

```
em = emf.createEntityManager();
```

Tendo agora um em (manipulador/gerenciar de entidade) podemos realizar a persistencia ao objeto.
Para dar inicio nas transações do objeto é necessário solicitar o inicio

```
em.getTransaction().begin();
```

E então para persistir os dados basta mandar os dados em **persist**, neste momento já temos a entidade criada com suas especificações (quem é a chave primaria, tipo de dados, etc) e então no momento da persistencia o persist desmonta a entidade em um script sql e envia os dados (funcionando a JDBC por baixo)

```
em.persist(grupoVO);
```

E por fim commitar as alterações/transações feitas, confirmando as alterações.

```
em.getTransaction().commit();
```

ALTERAÇÃO

Repete-se o mesmo processo de criação de gerenciar de entidade e criação de um objeto de manipuação de entidade. Neste momento estamos solicitando um dado ao usuario para alterar no bd, logo solicitamos o dado

```
String pNome = JOptionPane.showInputDialog("Forneca o nome do grupo de produto a ser localizado");
```

Criamos um query para realizar a consulta no banco de dados (em vez do select padrão), dentro desse query utilizamos um scrip especifico do JPA, o que em scripsql usamos nome de tabelas, aqui usamos nome de atributos do objeto.

```
Query consulta = em.createQuery("SELECT gp FROM GrupoProdutoVO gp WHERE UPPER(gp.nome) = :pNome");
```

Depois temos de setar o parâmetro com o seu valor

```
consulta.setParameter("pNome", pNome.toUpperCase());
```

Após setar, solicita-se uma consukta ao bd que retorne um resultado em forma de lista, entretando em forma de lista do objeto. Anteriormente em JDBC quando recebiamos varios dados do resultado tinhamos de percorrer todo o resultado e ir montando o objeto. Agora com o JPA esse processo já é feito automaticamente quando solicitado no gerResultList.

```
List<GrupoProdutoVO> lista = consulta.getResultList();
```

Observe que criamos uma lista do tipo objeto e essa lista vai receber da consulta (objeto query) uma lista de resultados. Não temos acesso a esse resultset, apenas a uma lista de objeto.

Agora com a lista de objeto feito, percorremos a listra, e adimitndo que o primeiro resultado é o mais preciso basta que adicionemos ele a um objeto. Em seguida solicitar os dados que deseja alterar, e setar as informações

```
if (lista.size() > 0) {
    grupoVO = lista.get(0);
    String nome = JOptionPane.showInputDialog("Forneca o nome do grupo de produto", grupoVO.getNome());
    float margem = Float.parseFloat(JOptionPane.showInputDialog("Forneca o percentual da margem de lucro do grupo
        de produto", grupoVO.getMargemLucro()));
float promocao = Float.parseFloat(JOptionPane.showInputDialog("Forneca o percentual de promocao do grupo de
    produto", grupoVO.getPromocao()));
grupoVO.setNome(nome);
grupoVO.setMargemLucro(margem);
grupoVO.setPromocao(promocao);
```

O merge é o responsável por realizar as alterações após setado

```
em.merge(grupoVO)
```

Em seguida basta commitar para confirmar as alterações

```
em.getTransaction().commit();
```

EXCLUSÃO

Repete-se o processo para alterar o objeto, diferindo apenas na que não vamos mais settar dados e sim pegar o primeiro resultado da lista e remove-lo.

```
if (lista.size() > 0) {
    grupoVO = lista.get(0);
    em.remove(grupoVO);
    em.getTransaction().commit();
```