Roteiro da Aula - JPA

Objetivo geral:

※ Conhecer os principais recursos fundamentais do JPA (Java Persistence API) na teoria e prática;

Visão geral sobre mapeamento objeto-relacional

Por anos, umas das principais dificuldade na utlização da abordagem orientada a objetos era a comunicação com bancos de dados relacionais.

```
@Override
public List<Seller> findByDepartment(Department dep) {


    PreparedStatement statement = null;
    ResultSet result = null;
    List<Seller> sellers = new ArrayList<Seller>();


    try {

        String sql = "SELECT seller.*,department.Name as DepName "
                + "FROM seller INNER "
                + "JOIN department "
                + "ON seller.DepartmentId = department.Id "
                + "WHERE DepartmentId = ? ORDER BY Name";

        statement = connection.prepareStatement(sql);

        statement.setInt(1, dep.getId());

        result = statement.executeQuery();

        Map<Integer, Department> map = new HashMap<Integer, Department>();

        while(result.next()) {

            Department department = map.get(result.getInt("DepartmentId"));

            if (department == null) {
                department = createDepartment(result);
                map.put(result.getInt("DepartmentId"), department);
            }

            Seller seller = createSeller(result, department);
```

Outros problemas que devem ser tratados:

- ※ Contexto de persistência (objetos que estão ou não atrelados a uma conexão em um dado momento)
- ※ Mapa de identidade (cache de objetos já carregados)
- ※ Carregamento tardio (lazy loading)
- ※ Outros

Java Persistence API (JPA)

Java Persistence API (JPA) é a especificação padrão da plataforma Java EE (pacote javax.persistence) para mapeamento objeto-relacional e persistência de dados.

JPA é apenas uma especificação (JSR 338): http://download.oracle.com/otn-pub/jcp/persistence-2_1-fr-eval-spec/JavaPersistence.pdf

Para trabalhar com JPA é preciso incluir no projeto uma implementação da API (ex: Hibernate).

Arquitetura de uma aplicação que utiliza JPA:

****

Principais classes:

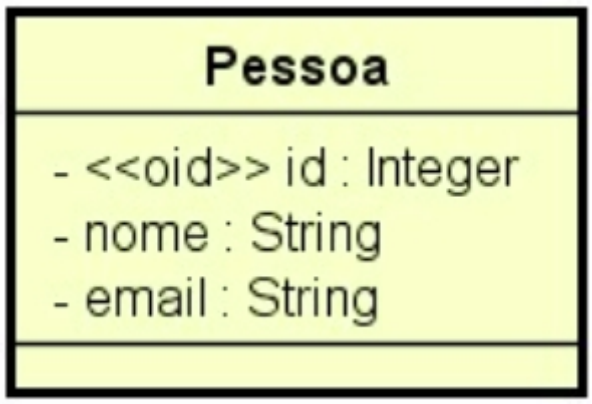
EntityManager
https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html

Um objeto EntityManager encapsula uma conexão com a base de dados e serve para efetuar operações de **acesso a dados** (inserção, remoção, deleção, atualização) em **entidades** (clientes, produtos, pedidos, etc.) por ele **monitoradas** em um mesmo **contexto de persistência**.

Escopo: tipicamente mantem-se uma instância única de EntityManager para cada thread do sistema (no caso de aplicações web, para cada requisição ao sistema).

EntityManagerFactory
https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManagerFactory.html
Um objeto EntityManagerFactory é utilizado para instanciar objetos EntityManager.
Escopo: tipicamente mantem-se uma instância única de EntityManagerFactory para toda aplicação.

Criando uma aplicação simples



Vamos instanciar três pessoas e mostrar seus dados na tela.

Passos:

1. Crie a classe "Pessoa" no pacote "dominio":

```
package dominio;

import (...)

public class Pessoa implements Serializable {
        private static final long serialVersionUID = 1L;

        private Integer id;
        private String nome;
        private String email;

        (...)
```

2. Crie a classe "Programa" no pacote "aplicacao"

```
Pessoa p1 = new Pessoa(null,"Jefferson Chaves","jefferson.chaves@email.com");
Pessoa p2 = new Pessoa(null, "Maria Ines","maria.ines@email.com");
Pessoa p3 = new Pessoa(null, "Leonardo Chaves","leonardo.chaves@email.com");

System.out.println(p1);
System.out.println(p2);
System.out.println(p3);
```

3. Incluindo JPA para persistir os objetos em banco de dados

a) Atualize o Maven do projeto para Java 11

i. Edite o arquivo pom.xml

ii. Inclua o conteúdo abaixo

iii. Salve o projeto

iv. Botão direito no projeto -> Maven -> Update Project

```
<properties>
	<maven.compiler.source>11</maven.compiler.source>
	<maven.compiler.target>11</maven.compiler.target>
</properties>
```

código

4. Inclua as dependências Maven a serem baixadas:

```
<dependencies>
	<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>5.4.12.Final</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager -->
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-entitymanager</artifactId>
		<version>5.4.12.Final</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.19</version>
	</dependency>
</dependencies>
```

código

5. **Configure o JPA no seu projeto por meio do arquivo persistence.xml**

    i. Crie uma pasta "META-INF" a partir da pasta "resources"

    ii. Dentro da pasta META-INF crie um arquivo "persistence.xml"

    iii. Conteúdo do arquivo persistence.xml

6. **Inclua os MAPEAMENTOS na classe de domínio:**

```
@Entity
public class Pessoa implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Integer id;
	(...)
```

7. **Teste sua aplicação:**

```
EntityManagerFactory entityFactory = Persistence.createEntityManagerFactory("loja-jpa");

EntityManager entityManager = entityFactory.createEntityManager();

entityManager.getTransaction().begin();

entityManager.persist(p1);
entityManager.persist(p2);
entityManager.persist(p3);

Pessoa p = entityManager.find(Pessoa.class, 2l);

System.out.println(p);

entityManager.getTransaction().commit();
entityManager.close();
```

Estados e ciclo de vida e Contexto de Persistência

Objetos de entidades são instâncias de classes mapeadas usando JPA, que ficam na memória e representam registros do banco de dados. Essas instâncias possuem um ciclo de vida, que é gerenciado pelo JPA. Os estados do ciclo de vida das entidades são: transient (ou new), managed, detached e removed.