

Curso Java COMPLETO

educandoweb.com

Prof. Dr. Nelio Alves

Nivelamento: mapeamento objeto-relacional com JPA / Hibernate

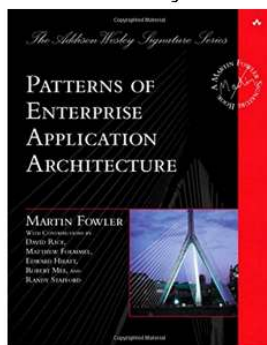
1) Visão geral sobre mapeamento objeto-relacional

O mapeamento objeto relacional veio pra melhorar a comunicação do sistema orientado a objetos com o sistema relacional

PROBLEMA:

Por vários anos, a maior dificuldade de se usar a abordagem orientada a objetos é a comunicação com o banco de dados relacional.

Martin Fowler: ~30% do esforço de se fazer um sistema



```
public Filme buscaPorCodigo(int cod) {  
  
    Filme f = null;  
    Connection c = FabricaDeConexao.getConexao();  
    try {  
        PreparedStatement  
        stmt = c.prepareStatement("select * from  
        stmt.setInt(1, cod);  
        ResultSet resultado = stmt.executeQuery(  
        if (resultado.next()) {  
            f = new Filme();  
            f.setCod_filme(resultado.getInt("co  
            f.setDescricao(resultado.getString(  
            f.setAno(resultado.getInt("ano"));  
            CategoriaDao catDao = new Categoria  
            Categoria aux = catDao.buscaPorCod  
            f.setCategoria(aux);  
        }  
        resultado.close();  
        c.close();  
    } catch (SQLException e) {  
        System.out.println("Erro ao tentar Lista  
        e.printStackTrace();  
    }  
  
    return f;  
}
```

Toda hora tem que ficar transportando de tabela para objeto e vice-versa

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

2) JPA

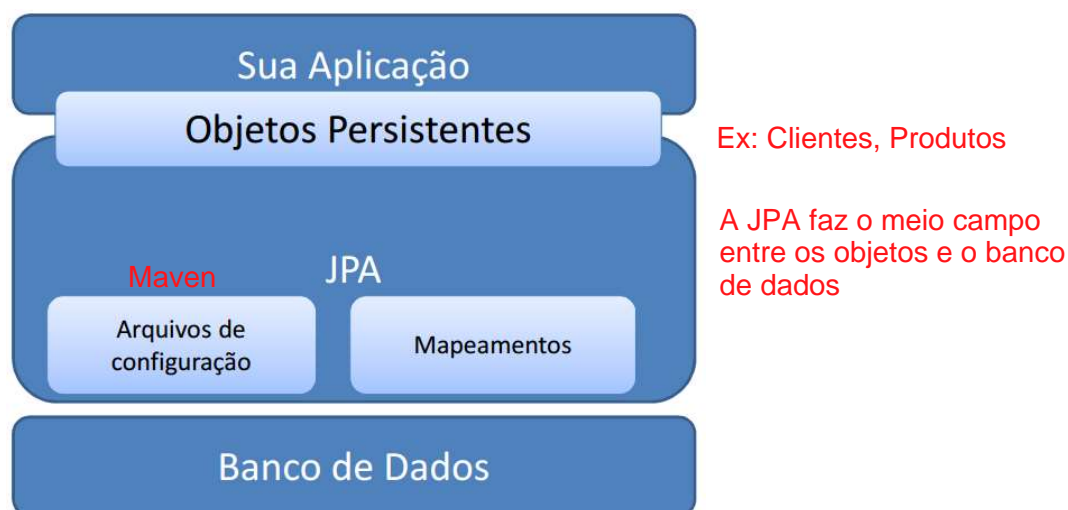
Java Persistence API (JPA) é a especificação padrão da plataforma Java EE (pacote **javax.persistence**) para mapeamento objeto-relacional e persistência de dados.

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 Faz a conexão e o acesso aos dados

<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 mantém-se uma instância única de EntityManager para cada thread do sistema (no caso de aplicações web, para cada requisição ao sistema).

EntityManagerFactory cria o entity manager

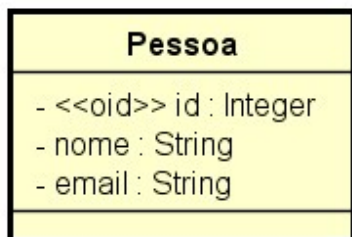
<https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManagerFactory.html>

Um objeto EntityManagerFactory é utilizado para instanciar objetos EntityManager.

Escopo: tipicamente mantém-se uma instância única de EntityManagerFactory para toda aplicação.

3) Criando uma aplicação simples

Vamos instanciar três pessoas e mostrar seus dados na tela.



Passos:

1) Mude a perspectiva do STS para Java

Window -> Perspective -> Open Perspective -> Java

2) Crie o projeto

File -> New -> Java Project

3) Crie a classe "Pessoa" no pacote "dominio" e faça os mapeamentos:

```
package dominio;

import (...);

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

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

    (...)
```

4) Crie a classe "Programa" no pacote "aplicacao"

```
Pessoa p1 = new Pessoa(1, "Carlos da Silva", "carlos@gmail.com");
Pessoa p2 = new Pessoa(2, "Joaquim Torres", "joaquim@gmail.com");
Pessoa p3 = new Pessoa(3, "Ana Maria", "ana@gmail.com");
System.out.println(p1);
System.out.println(p2);
System.out.println(p3);
```

4) Incluindo JPA para persistir os objetos em banco de dados

Passos:

1) Crie uma base de dados MySQL vazia

- Instale o Xampp no seu computador
- Inicie o Apache e o MySQL
- No PhpMyAdmin, crie uma base de dados chamada "aulajpa"

2) Crie um novo projeto Maven **Gerenciador de dependências e building do java**

- File -> New -> Other -> Maven Project
- Create Simple Project -> Next
- Group Id: com.educandoweb
- Artifact Id: aulajpamaven
- Finish

3) Copie as classes Programa e Pessoa para o novo projeto

4) Atualize o Maven do projeto para versão atual LTS Java

- Edite o arquivo pom.xml **Nesse arquivo é ficam as configurações do maven**
- Inclua o conteúdo abaixo
- Salve o projeto
- Botão direito no projeto -> Maven -> Update Project (Force update)

```
<properties>
  <maven.compiler.source>17</maven.compiler.source>
  <maven.compiler.target>17</maven.compiler.target>
</properties>
```

5) 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>
```

6) Configure o JPA no seu projeto por meio do arquivo persistence.xml

- Crie uma pasta "META-INF" a partir da pasta "resources"
- Dentro da pasta META-INF crie um arquivo "persistence.xml"
- Conteúdo do arquivo persistence.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd" version="2.1">

    <persistence-unit name="exemplo-jpa" transaction-type="RESOURCE_LOCAL">
        <properties>
            <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/aulajpa?useSSL=FALSE&serverTimezone=UTC" />

            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />

            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="" />

            <property name="hibernate.hbm2ddl.auto" value="update" />
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL8Dialect" />
        </properties>
    </persistence-unit>
</persistence>
```

Gerenciar as transações manualmente

Gera automaticamente o banco de dados.
update: Sempre que roda a aplicação o banco de dados vai ser atualizado.

7) Inclua os MAPEAMENTOS na classe de domínio: Qual classe corresponde a qual tabela e os relacionamentos

```
package dominio;

import (...);

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

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

Como o JPA trabalha? Por padrão ele vai criar uma tabela com o mesmo nome da classe e dentro dessa tabela ele vai criar os atributos com os mesmos nomes dos atributos aqui.

Se você quiser alterar um campo, vai ter que ser dentro da classe.

ex.

8) Na classe "Programa" faça os testes (veja vídeo-aula).