

Curso: Spring Boot com Ionic - Estudo de Caso Completo

<https://www.udemy.com/user/nelio-alves>

Prof. Dr. Nelio Alves

Capítulo: Implementação de modelo conceitual

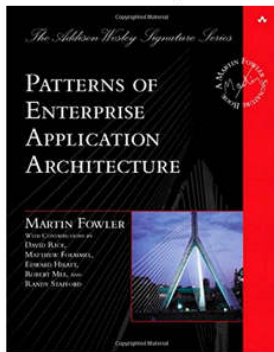
Aula extra: Nivelamento sobre JPA

1) Visão geral sobre mapeamento objeto-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.buscaPorCodi  
            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

<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

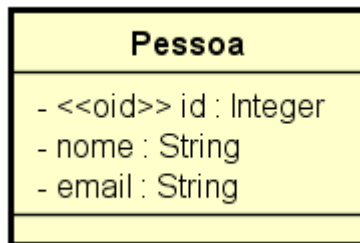
<https://docs.oracle.com/javaee/7/api/javax/persistence/EntityManagerFactory.html>

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 (se preciso veja a aula "Instalando o MySQL" do Capítulo 4)
- Inicie o Apache e o MySQL
- No PhpMyAdmin, crie uma base de dados chamada "aulajpa"

2) Acrescente as bibliotecas Hibernate e MySQL Connector ao projeto

- Crie uma pasta "lib" no projeto: Botão direito no projeto -> New -> Folder
- Copie os arquivos .jar anexos à aula
- Botão direito no projeto -> Buildpath -> Configure build path
- Aba Libraries -> Add JARs (selecione os arquivos)
- Apply and Close

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

- Crie uma pasta "META-INF" a partir da pasta "src"
- 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.driver" value="com.mysql.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/aulajpa" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="" />

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

4) Inclua os MAPEAMENTOS na classe de domínio:

```
package dominio;

import (...)

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

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

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