Vídeo 5.2: Persistindo dados no banco

Olá, nessa videoaula você irá realizar a persistência de informações no banco de dados fazendo-se uso dos frameworks JPA e Hibernate, que foram configurados em seu projeto na videoaula 5.1. Para você ter sucesso na persistência dos dados o primeiro passo que você deve realizar é mapear a entidade que você irá persistir no banco de dados.

A entidade que você irá mapear nesse momento é a entidade Filme, que foi criada previamente através do Eclipse na videoaula 3.1. Agora, acesse o Eclipse e navegue até a classe Filme. É importante lembrar que toda classe para ser mapeada deve conter um identificador que seja único para aquela entidade. Esse identificador deve funcionar como uma chave primária. Nesse sentido, você iráadicionar a sua classe um atributo chamado Id, que será do tipo Integer. Feito isso, você irá criar os métodos get e set desse atributo. Anteriormente, na videoaula 3.1, você criou um construtor para a classe Filme que recebia alguns parâmetros como nome, gênero e ano, a fim de facilitar a criação dos seus objetos. Agora, você deve criar um construtor padrão, ou seja, um construtor que não receba nenhum parâmetro.

Executado os passos anteriores, agora, você pode iniciar o mapeamento da classe Filme. Inicialmente, adicione a anotação @Entity, logo acima da declaração da classe dentro do Eclipse. Essa anotação permite que os objetos dessa classe possam ser persistidos. Em seguida, você deve informar também qual o atributo da sua classe servirá como chave, nesse caso será o id e isso deve ser informado através da anotação @Id acima do atributo id. Nesse momento, você não deve se preocupar com a geração desse identificador, deixe essa responsabilidade para o banco de dados. A próxima ação que você deve executar, é no que diz respeito a adição da anotação @GeneratedValues também acima do atributo.

#Audiodescrição: segue Código.

```
package br.com.lead.modelo;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Filme {

        @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
        private Integer id;
        private String nome;
        private String genero;
        private Integer ano;

        public Filme() {}

        public Filme(String nome, String genero, Integer ano) {
                this.nome = nome;
                this.genero = genero;
                this.ano = ano;
        }

        public Integer getId() {
    return id;
  }

  public void setId(Integer id) {
    this.id = id;
  }

        public String getNome() {
                return nome;
        }

        public String getGenero() {
                return genero;
        }

        public Integer getAno() {
                return ano;
        }
}
```

Com esse mapeamento, ao se realizar a persistência dos seus objetos, o Hibernate irá se encarregar de criar uma tabela dentro do schema catalogodefilmes, com as colunas id, nome, gênero e ano, sem a necessidade de você escrever uma linha de código SQL.

O Hibernate te fornece diversas possibilidades. Caso você deseje que algum atributo tenha um nome diferente da coluna do banco de dados, você pode adicionar, acima do atributo desejado, a anotação @column(name=”nome_desejado”) (@column e entre parênteses name igual nome desejado entre aspas). Caso você não deseje que determinada coluna não receba valores nulos, você pode adicionar a seguinte anotação acima do atributo @Column(nullable = false) (@column e entre parênteses nullable igual false). Com intuito de praticar, você deve fazer isso no atributo nome. Ele não pode receber nulo, e a coluna do banco de dados deve ser nm_filme. O atributo nome deverá ficar assim:

```
@Column(name = "nm_filme", nullable = false)
private String nome;
```

Perfeito! você realizou o mapeamento da entidade Filme, agora, você irá realizar a persistência dos dados. Na videoaula da aula 2, você aprendeu a criar uma Servlet, que auxiliou na listagem dos seus filmes. Logo em seguida, você irá criar uma Servlet chamada PersisteFilmeServlet, que se encarregará de persistir os filmes no banco de dados, a partir de uma requisição HTML. Você deve seguir o mesmo processo estudado anteriormente para criação dessa classe.

Com a Servlet criada, você deve informar o seguinte mapeamento @WebServlet("/persistir-filme") (@WebServlet e entre parênteses e aspas duplas barra persistir-filme) para o diferenciar da outra Servlet. No método service, você irá capturar os parâmetros que passarão na requisição, ou seja, nome, gênero e ano do filme, utilizando o getParameter do objeto da requisição. Com esses dados capturados, você criará o objeto filme, passando as informações por parâmetro. Feito isso, você deve, nesse momento, realizar a persistência desse objeto no banco de dados. Então, para que o JPA consiga realizar a persistência dos dados no banco, você deve realizar a leitura do arquivo persistence.xml, criado com intuito de que as configurações possam ser carregadas. Para a realização desse processo você deve adicionar a instrução EntityManagerFactory emf = Persistence.createEntityManagerFactory("catalogodefilmes") (EntityManagerFactory espaço emf recebe Persistence.createEntityManagerFactory, e, entre parênteses e aspas duplas, catalogodefilmes), logo abaixo da criação do objeto de Filme. Com esse passo concluído, você terá um objeto do tipo EntityManagerFactory. Esse objeto será utilizado para que você crie uma instância do objeto EntityManager, que será a responsável por realizar a comunicação com o JPA. Com esse objeto criado, você irá abrir uma transação com o banco de dados, e, então, realizar a persistência do objeto. Após isso, deve-se fechar a transação; fechar o objeto de EntityManager; e fechar o objeto EntityManagerFactory. Feito isso, o seu método deverá ficar assim:

#Audiodescrição: segue código.

```
@Override
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        String nome = req.getParameter("nome");
        String genero = req.getParameter("genero");
        Integer ano = Integer.valueOf(req.getParameter("ano"));
        Filme filme = new Filme(nome, genero, ano);
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("catalogodefilmes");
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();
        em.persist(filme);
        em.getTransaction().commit();
        em.close();
        emf.close();
}
```

Para validar se o que fez está correto, vamos até a aba Servers do Eclipse e vamos executar novamente a aplicação através do Tomcat, para, assim, enviar uma requisição para a Servlet e validar o resultado, como será demonstrado a seguir. Com a aplicação rodando, acesse a seguinte URL através do seu navegador web http://localhost:8080/CatalogoDeFilmes/persistir-filme?nome=deadpool&genero=comedia&ano=2016 a página não irá nos apresentar uma resposta, pois não implementamos um retorno para essa requisição, mas, para conferir se os dados foram persistidos como desejado, vamos acessar novamente o banco de dados, utilizando o programa MySQL 8.0 Command Line Client, como foi visto anteriormente, e vamos acessar a base de dados catalogodefilmes, através do comando use catalogodefilmes, após esse comando, vamos dar um select na nossa tabela filme para vermos se foi criado alguma informação, select * from filme;. Após o comando, percebemos que o registro foi salvo no banco, pois foi retornado a informação referente ao filme inserido, id 1, ano 2016, gênero comedia e nm_filme deadpool.

#A imagem mostra a tela de início do “MySQL 8.0 Command Line Client”. Sobre fundo preto, há uma saudação e informações do banco de dados. Entre as informações do banco de dados, há o comando “mysql> use catalagodefilmes
Database changed

mysql> select * from filme

- > ;

Id/ ano/ genero/ nm_filme

1/ 2016/ comedia/ deadpool

Em seguida, está escrito: “1 row in set (0.00 sec)

mysql>”

Até aqui, você estudou sobre persistência de dados, utilizando o Hibernate e JPA, ou seja, frameworks ORM. Esses frameworks são os mais utilizados no universo Java. Além disso, você com certeza entendeu suas configurações e aplicabilidade, mas existe uma gama de outras possibilidades que dar para fazer com elas, você sabia? No mais, você conheceu os conceitos básicos acerca de mapeamento de entidades; viu sobre o banco MySQL; aprendeu a preparar o arquivo persistence.xml; e como persistir seus objetos. Com isso, é importante que você continue explorando as ferramentas vistas e com o intuito de aprofundar seus estudos.

Na próxima videoaula, você irá aprofundar seus conhecimentos acerca dessas ferramentas, para, assim, melhorar mais ainda sua aplicação. Até mais!

Referências

AFONSO, Alexandre. **Tutorial definitivo:** *tudo o que você precisa para começar bem com JPA.* Disponível em: <https://blog.algaworks.com/tutorial-jpa/>. Acesso em: 7 maio 2020.

ALVES, Gustavo Furtado de Oliveira. **Como instalar o MySQL nos indos (Passo a Passo!).** Disponível em: <https://dicasdeprogramacao.com.br/como-instalar-o-mysql-no-windows/>. Acesso em: 6 maio 2020.

Java para desenvolvimento web. Disponível em: <https://www.caelum.com.br/apostila-java-web/>. Acesso em: 6 maio 2020.

MySQL Community Downloads. Disponível em: <https://dev.mysql.com/downloads/installer/>. Acesso em: 22 maio 2020.