

VÍDEO 7.2: Persistindo dados referente a Autor

Olá, seja bem-vindo!

Nesta videoaula, você aprenderá a fazer alterações na classe que realiza a persistência dos objetos de Filme no banco, isto é, a PersisteFilmeServlet. Como visto na aula 5 sobre persistência, essa classe recebe, como parâmetro da requisição, os dados de filme, nome, gênero e ano.

A partir de agora, ela também receberá os dados de autor, por parâmetro da requisição. Para isso, utilize o método `req.getParameter` para capturar os parâmetros `nomeAutor`, referente ao nome do autor do filme e o parâmetro `dataNascimentoAutor`, referente à data de nascimento do autor. Esse parâmetro `dataNascimentoAutor` deve ser convertido para um objeto de data, no caso, `LocalDate`, pois o retorno da requisição `req.getParameter` é uma `String`. Diante Disso, você deve utilizar o método `LocalDate.parse` para realizar essa transformação.

```
@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"));

    String nomeAutor = req.getParameter("nomeAutor");
    LocalDate dataNascimentoAutor = LocalDate.parse(req.getParameter("dataNascimentoAutor"));
}
```

#Audiodescricao: a imagem mostra o seguinte 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"));
}
```

Centro de Pesquisa, Desenvolvimento e Inovação Dell

Telefone:(85)3492-1062 | www.leadfortaleza.com.br
Av. Santos Dumont, 2456 - 1906 | 60150162 - Fortaleza. CE

```
String nomeAutor = req.getParameter("nomeAutor");  
LocalDate dataNascimentoAutor =  
    LocalDate.parse(req.getParameter("dataNascimentoAutor"));
```

Feito isso, você deve criar agora um objeto do tipo Autor, nomeado autor e atribuir a ele as informações de nomeAutor e dataNascimentoAutor, utilizando os métodos setters de cada atributo.

```
Autor autor = new Autor();  
autor.setDataNascimento(dataNascimentoAutor);  
autor.setNome(nomeAutor);
```

#Audiodescricao: a imagem mostra o seguinte código:

```
Autor autor = new Autor();  
autor.setDataNascimento(dataNascimentoAutor);  
autor.setNome(nomeAutor);
```

Nesse método, você tem um objeto filme criado. Esse objeto foi utilizado na aula de persistência para persistir os dados de filme. Assim, você deve utilizá-lo e atribuir, a esse objeto, a informação de autor, utilizando o método setter de Filme.

```
Filme filme = new Filme(nome, genero, ano);  
filme.setAutor(autor);
```

#Audiodescricao: a imagem mostra o seguinte código:

```
Filme filme = new Filme(nome, gênero, ano);  
filme.setAutor(autor);
```

Pronto, agora você tem os objetos já montados, mas antes prossiga com a persistência no banco. No método service, que você está editando, já existe a

Centro de Pesquisa, Desenvolvimento e Inovação Dell

Telefone: (85) 3492-1062 | www.leadfortaleza.com.br
Av. Santos Dumont, 2456 - 1906 | 60150162 - Fortaleza. CE

persistência do objeto filme. Assim, você deve agora realizar a persistência do objeto Autor. Como o objeto de Filme possui a informação de Autor, persista primeiro Autor, para só depois persistir Filme. O método todo deve ficar da seguinte maneira:

```
@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"));

    String nomeAutor = req.getParameter("nomeAutor");
    LocalDate dataNascimentoAutor = LocalDate.parse(req.getParameter("dataNascimentoAutor"));

    Autor autor = new Autor();
    autor.setDataNascimento(dataNascimentoAutor);
    autor.setNome(nomeAutor);

    Filme filme = new Filme(nome, genero, ano);
    filme.setAutor(autor);

    EntityManager em = JPAUtil.getEntityManager();

    em.getTransaction().begin();
    em.persist(autor);
    em.persist(filme);
    em.getTransaction().commit();

    em.close();
}
```

#Audiodescricao: a imagem mostra o seguinte 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"));
```

```
    String nomeAutor = req.getParameter("nomeAutor");
```

```
    LocalDate dataNascimentoAutor =
```

```
LocalDate.parse(req.getParameter("dataNascimentoAutor"));
```

```
    Autor autor = new Autor();
```

```
    autor.setDataNascimento(dataNascimentoAutor);
```

Centro de Pesquisa, Desenvolvimento e Inovação Dell

Telefone:(85)3492-1062 | www.leadfortaleza.com.br

Av. Santos Dumont, 2456 - 1906 | 60150162 - Fortaleza. CE

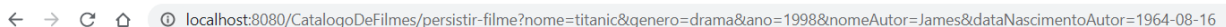
```
autor.setNome(nomeAutor);
```

```
Filme filme = new Filme(nome, genero, ano);  
filme.setAutor(autor);
```

```
EntityManager em = JPAUtil.getEntityManager();
```

```
em.getTransaction().begin();  
em.persist(autor);  
em.persist(filme);  
em.getTransaction().commit();  
  
em.close();  
}
```

O próximo passo é a execução da aplicação no Tomcat a fim de ser possível realizar a persistência de um novo filme, agora, passando os dados do autor do filme. Para isso, vá até a aba Server do Eclipse e navegue no menu de opções até a opção Start. Com o Tomcat executando, utilize a URL de persistência de Filme <http://localhost:8080/CatalogoDeFilmes/persistir-filme?nome=titanic&genero=drama&ano=1998&nomeAutor=James&dataNascimentoAutor=1964-08-16> para realizar uma requisição pelo navegador.

A screenshot of a web browser's address bar showing the URL: localhost:8080/CatalogoDeFilmes/persistir-filme?nome=titanic&genero=drama&ano=1998&nomeAutor=James&dataNascimentoAutor=1964-08-16. The browser interface includes back, forward, and refresh buttons.

#Audiodescricao: a imagem mostra um recorte de um navegador de internet. Nele, temos a barra de ferramentas, com as opções “Voltar”, “Avançar”, “Recarregar” e “Página inicial”, além da barra de endereço, com o seguinte endereço URL: “localhost:8080/CatalogoDeFilmes/persistir-

filme?nome=titanic&genero=drama&ano=1998&nomeAutor=James&dataNascimentoAutor=1964-08-16”.

Após realizar a requisição pelo navegador, você deve acessar, novamente, o banco para verificar se a persistência deu certo. Iremos avaliar como ficou a estrutura do banco de dados. Para isso, você utilizará o MySQL 8.0 CommandLine Cliente de novo. Como visto em aulas passadas, ao entrar no MySQL 8.0, você deve informar a senha do banco de dados para ter acesso ao programa. Após acessar o MySQL, utilize o comando `use catalogodefилmes`, para acessar a base `catalogodefилmes`. Em seguida, utilize o comando `show tables`, para verificar as tabelas existentes na base de dados. Com isso, você perceberá que foi criada a tabela `autor` no banco de dados.

```
mysql> use catalogodefилme;
ERROR 1049 (42000): Unknown database 'catalogodefилme'
mysql> use catalogodefилmes;
Database changed
mysql> show tables
+-----+
| Tables_in_catalogodefилmes |
+-----+
| autor                       |
| filme                      |
+-----+
2 rows in set (0.00 sec)

mysql>
```

#Audiodescriçao: a imagem mostra uma tabela:

```
mysql> use catalogodefилmes;
Error 1049 (42000): Unknown database 'catalogodefилmes'
mysql> use catalogodefилmes;
Database changed
mysql> show tables
->;
+-----+
| Tables_ catalogodefилmes |
+-----+
| autor                       |
| filme                      |
+-----+
```

```
+-----+
2 rows in set (0.00 sec)
```

mysql>

Para verificar o conteúdo da tabela, você deve dar um select na tabela autor, select * from autor;.

```
mysql> select * from autor;
+-----+
| id | data_nascimento | nome |
+-----+
| 1 | 1964-08-16      | James |
+-----+
1 row in set (0.00 sec)

mysql>
```

#Audiodescriçao: a imagem mostra uma tabela:

```
mysql> select * from autor ;
+-----+
| id | data_nascimento | nome |
+-----+
| 1 | 1994-08-16 | James |
+-----+
1 row in set (0.00 sec)
```

mysql>

Feito isso, você perceberá que a informação referente a Autor foi adicionada com sucesso na tabela autor. Os dados do autor são: nome, James;data_nascimento, 16/08/1964; e id igual a 1. Agora, você irá verificar a tabela filme. Para isso, execute um select em filme.

```
mysql> select * from filme;
+-----+-----+-----+-----+-----+
| id | ano | genero | nm_filme | autor_id |
+-----+-----+-----+-----+-----+
| 1 | 2016 | comedia | deadpool | NULL |
| 2 | 1998 | drama | titanic | 1 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

#Audiodescriçao: a imagem mostra a tabela filme:

```
mysql> select * from filme;
```

```
+-----+-----+-----+-----+
| id | ano | nm_filme | autor_id |
+-----+-----+-----+-----+
| 1 | 2016 | comedia | deadpool | NULL |
| 2 | 1998 | drama | titanic | 1 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Perceba que, na tabela filme, foi adicionada uma nova coluna referente ao id do autor (autor_id). Agora, cada registro de filme ficará relacionado a um registro de autor. Por exemplo, o filme Titanic que realizamos a persistência, está relacionado ao registro 1 de autor, pela coluna autor_id.

Quando você realizou a persistência do primeiro registro de Filme em aulas passadas, referente ao filme Deadpool, ainda não existia a estrutura de autor criada. Logo, a coluna, referente ao autor_id de Deadpool, não foi preenchida, ou seja, está com o valor nulo. No registro que você adicionar agora, referente ao filme Titanic, você tem a informação do id do autor preenchida.

A partir disso, percebe-se que o JPA realizou a alteração na estrutura da base de dados sem a necessidade de você realizar alguma instrução SQL no banco. Nesse modelo de relacionamento, cada registro de Filme ficará vinculado a um registro de Autor. Isso ocorrerá através do id do filme.

Até aqui, você realizou o mapeamento da entidade Autor e criou um relacionamento entre as entidades Filme e Autor para que sua aplicação fique mais próxima do que existe no mundo real. Além disso, realizou-se uma persistência de um novo filme, só que, agora, informando também os dados referente ao autor do filme. Ademais, foi verificado como a sua estrutura de banco de dados foi alterada pelo JPA, ao realizar esses mapeamentos e adicionar um relacionamento.

Uma outra possibilidade de relacionamento entre entidades é o relacionamento um para um. Como exemplo, suponha a existência de duas entidades, Pessoa e InformacoesPessoais. No relacionamento um para um, cada registro no banco referente à entidade Pessoa, só faz sentido se tiver apenas um registro no banco da entidade InformacoesPessoais que se relacione com o registro de Pessoa. E, para cada registro no banco da entidade InformacoesPessoais, só faz sentido ter apenas um registro no banco da entidade Pessoa que se relacione com InformacoesPessoais. Caso necessite utilizar esse tipo de relacionamento, a anotação a ser utilizada em cima do seu atributo deve ser @OneToOne. Por isso, preste muita atenção: são dois relacionamentos distintos que são aplicados em situações distintas. Porém, o local onde é posicionado a anotação é o mesmo, logo acima do atributo que deseja mapear.

Por exemplo:

@OneToOne

private Pessoa pessoa;

Até esse ponto do conteúdo, você conheceu os relacionamentos @ManyToOne e @OneToOne, certo? Ótimo! Agora vamos conhecer como ocorre o relacionamento @ManyToMany. Portanto, para a demonstração desse relacionamento, suponha duas entidades, Aluno e Curso. Nesse caso, cada registro no banco de dados, referente a Aluno, pode estar vinculado a vários registros da entidade Curso. Assim, como cada registro de Curso pode estar vinculado a vários registros de Aluno. Nessa situação, é necessário que você utilize o mapeamento @ManyToMany.

A partir dos exemplos apresentados, foi percebido que decidir o tipo de relacionamento que irá existir entre duas entidades da sua aplicação, é uma decisão referente à lógica do seu negócio. Por exemplo, uma pessoa só faz sentido ter apenas um registro de informação pessoal e cada registro de informação pessoal, só faz sentido para uma pessoa. Da mesma forma que um aluno pode estar vinculado a vários cursos, assim, como um curso pode estar vinculado a vários alunos.

Quando o seu projeto está bem modelado, bem organizado referente à lógica do negócio, o mapeamento na aplicação das suas entidades torna-se mais claro. Uma boa prática para o desenvolvimento de qualquer aplicação, que envolva banco de dados, é quando o desenvolvedor começa, em primeiro lugar, pela modelagem da estrutura do banco, para só depois partir para a implementação das suas entidades.

Até aqui, você aprendeu sobre relacionamento entre entidades. Com esse conteúdo apresentado, você conseguirá manipular as entidades selecionadas por você, de acordo com uma modelagem de dados definida para sua aplicação. Vale salientar que existe outras possibilidades de mapeamentos entre entidades, além do que foi apresentado nesta videoaula. Então, é importante que você explore as possibilidades pelo Eclipse, e pesquise em documentações e fóruns acerca do assunto apresentado, afim de melhorar suas habilidades.

Até a próxima videoaula!

Referências

Hibernate Mapping: Mapeando Relacionamentos entre Entidades. Disponível em: <<https://www.devmedia.com.br/hibernate-mapping-mapeando-relacionamentos-entre-entidades/29445>>. Acesso em: 17 maio 2020.

HibernateOneToManyAnnotation Tutorial. Disponível em: <<https://www.baeldung.com/hibernate-one-to-many>>. Acesso em: 17 maio 2020.

Invista em você! Saiba como a DevMedia pode ajudar sua carreira.

Many-To-Many Relationship in JPA. Disponível em: <<https://www.baeldung.com/jpa-many-to-many>>. Acesso em: 17 maio 2020.

Centro de Pesquisa, Desenvolvimento e Inovação Dell

Telefone: (85) 3492-1062 | www.leadfortaleza.com.br
Av. Santos Dumont, 2456 - 1906 | 60150162 - Fortaleza. CE

One-to-OneRelationship in JPA. Disponível em: <<https://www.baeldung.com/jpa-one-to-one>>. Acesso em: 17 maio 2020.