Unidade 15: Java Persistence API

Prof. Daniel Caetano

INTRODUÇÃO

Na maior parte deste curso foi usada a persistência implementando o padrão DAO (Data Access Objects), um padrão criado pela Microsoft. O Java, entretanto, tem uma API padronizada para o uso de persistência independente de implementação, chamada Java Persistence API ou JPA.

Para o uso dessa API, é necessário escolhermos uma implementação; neste curso iremos utilizar a implementação TopLink, que já está presente no NetBeans. Com base na JPA e no TopLink, iremos implementar um pequeno cadastro, introduzindo as funcionalidades mais básicas do JPA.

1. CRIANDO UM BANCO DE DADOS DE LOCADORA COM JPA

Primeiramente precisamos criar um banco de dados.

PASSO 1. Antes de mais nada, vamos iniciar o servidor de Banco de Dados. Para isso, vá na aba **Serviços > Banco de Dados > JavaDB** e clique com o botão direito, selecionando a opção "**Inicializar Servidor**".

PASSO 2. Vamos criar agora o banco. Para isso, vá na aba **Serviços > Banco de Dados > JavaDB** e clique com o botão direito, selecionando a opção "**Criar Banco de Dados**". Na janela que vai aparecer, use os seguintes dados para criar o banco:

Nome: **locadora**
User: **nbuser**
Senha: **nbuser**

PASSO 3. A fim de facilitar nossa vida nos próximos passos, vamos dar um nome mais amigável para o banco de dados. Assim, mude o nome do link do banco para "**Locadora DB**" (isso pode ser feito pela opção "propriedades" ou clicando duas vezes, lentamente, no nome do link).

Com o banco de dados criado, agora vamos criar um projeto. Não se preocupe com as tabelas, o NetBeans as criará para nós, usando o JPA.

PASSO 4. Vamos criar o projeto. Para isso, clique no ícone novo projeto. **Selecione Java > Aplicativo Java**. Dê o nome de **Projeto5** para o projeto.

PASSO 5. Vamos criar o pacote para nossos arquivos de dados. Clicando na pastinha de código fonte, selecione a opção **Novo > Pacote...** E dê o nome de "**entidades**" para o pacote.

Vamos agora criar a classe de entidade que representa o filme.

PASSO 6. Vamos criar agora o nosso arquivo de entidade. Clicando com o botão direito no pacote "entidades", selecione a opção **Novo > Classe da Entidade...** E a configure com o nome **Filme**. Clique no botão **Próximo**. Escolha como Biblioteca o "**Toplink Essentials (JPA 1.0)**" e como conexão o "**Locadora DB**".

Observe que na pasta META-INF surgiu um arquivo chamado **persistence.xml**. Neste arquivo é configurada a persistência, indicando a "tradução" dos dados.

PASSO 7. Selecione o arquivo **Filme.java** para edição. Vamos adicionar alguns atributos à classe. Abaixo do **private Long id**, insira:

```
private String nome;
private int tempo;
```

PASSO 8. Como inserimos atributos, precisaremos criar os getters e setters. Clique com o botão direito e selecione a opção **Inserir código... > Getter e Setter**. Marque todos os atributos e clique no botão **Gerar**.

PASSO 9. Podemos ajustar os métodos **hashCode**, **equals** e **toString** manualmente, mas é mais fácil deixar que o NetBeans os ajuste por nós. Sendo assim, **apague** os métodos citados acima.

PASSO 10. Vamos recriar o equals e o hashCode. Clique com o botão direito e selecione a opção **Inserir código... > equals() e hashCode()**.Marque todos os atributos e clique no botão **Gerar**.

PASSO 11. Vamos recriar o toString. Clique com o botão direito e selecione a opção **Inserir código... > toString()**.Marque todos os atributos e clique no botão **Gerar**.

A entidade está pronta. Para podermos usar o JPA, entretanto, precisamos criar um elemento chamado JPA Controller, algo similar a um DAO.

PASSO 12. Vamos criar agora o nosso arquivo Controller. Clicando com o botão direito no pacote "entidades", selecione a opção **Novo > Outro.** Agora selecione **Persistence > Classes do Controlador do JPA de classes de entidade** e clique em **Próximo**. Clique no botão **Adcionar tudo >>** e depois clique em **Próximo**. Clique em **Finalizar**.

Como vamos usar o JavaDB (Derby), precisamos adicionar o driver para o mesmo em nosso projeto. Isso só é necessário porque estamos trabalhando com um aplicativo Java tradicional (desktop). Caos estivéssemos criando uma Web Application com o GlassFish, o Derby estaria incluído automaticamente, já que faz parte do pacote do GlassFish.

PASSO 13. Vamos então inserir o driver do JavaDB. Na área de projeto, selecione a pasta "Bibliotecas" e clique com o botão direito do mouse. No menu, selecione a opção "**Adicionar Jar/Pasta**". Adicione estes dois arquivos:

```
Program Files>GlassFish>javadb>lib>derby.jar
Program Files>GlassFish>javadb>lib>derbyclient.jar
```

Com tudo isso pronto, vamos agora usar nossa classe de entidade.

PASSO 14. Vá até o arquivo **Projeto5.java**. Dentro do método **main**, insira o seguinte código:

```
int i;
Filme f;
List<Filme> lista;

EntityManagerFactory emf;
emf = Persistence.createEntityManagerFactory("Projeto5PU");
FilmeJpaController jpa = new FilmeJpaController(emf);

f = new Filme();
f.setNome("Harry Potter");
f.setTempo(150);
jpa.create(f);

f = new Filme();
f.setNome("Senhor dos Anéis");
f.setTempo(170);
jpa.create(f);

lista = jpa.findFilmeEntities();
for (i=0; i<lista.size(); i++) {
        f = lista.get(i);
        System.out.println(f);
}

f.setTempo(180);
jpa.edit(f);                  // Acrescentar try/catch com ALT+ENTER

lista = jpa.findFilmeEntities();
for (i=0; i<lista.size(); i++) {
        f = lista.get(i);
        System.out.println(f);
}
```

PASSO 15. Esse código não compilará, visto que a linha **jpa.edit** pode causar erros de processamento. Sendo assim, envolva-a em um try~catch, usando o **ALT+ENTER**.

Tudo pronto? Vejamos se funcionou!

PASSO 16. Vá até a aba Serviços. Selecione "**Desconectar DB**" e depois "**Conectar DB**". No esquema **JavaDB**, clique com o botão direito em **Filme** e escolha **Visualizar Dados**.

2. COMO REALIZAR BUSCAS COM "WHERE"?

O JPAController criado pelo NetBeans contém alguns comandos padrão:

create(filme)	– cria elemento
edit(filme)	– atualiza elemento
destroy(id)	– remove elemento
findFilmeEntities()	– realiza buscas
findFilmeEntities(max,first)	– realiza buscas
findFilmeEntities(all, max,first)	– realiza buscas
findFilme(id)	– realiza buscas

Essas buscas são genéricas, isto é, não "filtram" por atributos específicos, como usualmente fazemos com o SQL. Para buscas com filtros, temos que criar um método no JPAController, incluindo um SELECT diferente. Observe o SELECT do findFilmeEntities(all, max,first)

```
Query q = em.createQuery("select object(o) from Filme as o");
```

Observe como isso se parece com um SQL comum, mas trata com objetos. Para poder filtrar, por exemplo, pelo nome "Harry Potter" podemos usar uma linha como a indicada abaixo:

```
Query q = em.createQuery("select object(o) from Filme as o where
o.nome LIKE 'Harry Potter'");
```

3. COMO EVITAR INSERIR ELEMENTOS REPETIDOS

Para evitar inserir elementos repetidos, deve-se sempre realizar uma busca específica antes, ou, supondo-se que a chave primária ID não é automática, tentar fazer um **edit** antes de um **create**.

Para facilitar esta tarefa, o método edit já retorna a exceção **NonexistentEntityException**, que indica exatamente quando uma entidade não existe e, portanto, precisa ser criada.