Olá! Tudo bem?

Na videoaula passada, o estudante foi instruído a respeito da instalação do MySQL; estudou sobre persistência de dados e como funciona os frameworks de persistência JPA e Hibernate; viu também como esses frameworks interagem com o banco de dados MySQL; e conheceu e aplicou o EntityManager, que foi o responsável por persistir o objeto filme no banco de dados. Neste vídeo, você continuará o estudo sobre persistência, aprofundando os conhecimentos acerca do EntityManager. Além de conhecer os estados de persistência de uma entidade que são: Managed, Detached, Transient e Deleted. Com eles, você poderá entender como funciona o ciclo de persistência de dados.

Toda operação da aplicação Java no banco de dados para manuseio das persistências deve ser realizada através de uma API (ApplicationProgramming Interface ou Interface de Programação de Aplicações). Essa API deve prover todos os métodos necessários para o manuseio correto das operações, como abrir uma transação com o banco, persistir entidades, realizar o resgate de uma entidade do banco e realizar alterações nos dados dessa entidade. No seu projeto, quem realiza o papel da API é o EntityManager.

No processo de persistência das entidades na aplicação, o Hibernate define alguns estados de persistência dessas entidades. Bem, um desses estados é o Managed. Ele ocorre nesse estado quando o objeto da aplicação está em sincronismo com a entidade do banco de dados, e qualquer alteração realizada nesse objeto será refletida no banco de dados. Assim, um dos momentos que você pode se deparar com essa situação é quando realizar o resgate de uma entidade do banco de dados através da chamada ao método find de EntityManager, e atribuir o retorno desse método a um objeto. Por exemplo: você cria um objeto de nome emf do tipo EntityManagerFactory e atribui a esse objeto o comando Persistence.createEntityManagerFactory, passando, entre parênteses, catalogodefilmes, entre aspas. Na linha abaixo, deve-se criar um objeto nomeado do tipo EntityManager, e atribuir a ele o valor retornado do comando emf.createEntityManager, assim você abre e fecha parênteses. Na linha abaixo novamente, utilize o método begin de em.getTransaction para abrir uma transação com o banco de dados. Mais uma vez, na linha abaixo, realize o comando em.find, passando, entre parênteses, Filme.class virgula 1.Esse comando irá retornar um objeto do tipo Filme, de identificador igual a 1.

```
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("catalogodefilmes");
        EntityManagerem = emf.createEntityManager();
        em.getTransaction().begin();

        Filme filme = em.find(Filme.class, 1);
                ...
```

O objeto filme que recebeu o retorno do método find está em sincronismo com a entidade, e, portanto, qualquer alteração dos seus dados será espelhada no banco de dados MySQL.

Tudo bem até aqui? Senão, calma! Através de uma atividade prática você compreenderá como isso acontece, ok?

Em primeiro lugar, você deve isolar, em seu projeto, a criação da EntityManager, através da fábrica EntityManagerFactory, pois você irá reutilizar o mesmo trecho de código em outros pontos do projeto. Feito isso, você irá criar uma classe chamada JPAUtil e adicioná-la ao pacote br.com.lead.util. Para isso, você deve utilizar o Eclipse novamente e seguir o passo de criação de uma nova classe. Depois, navegue até o seu projeto, utilizando a tecla Aplicação do seu teclado; em seguida, navegue no menu exibido; acesse New; e, por último, Class.

#Audiodescrição: A imagem mostra a caixa de diálogo

“New Java Class”. Nela, há o texto “Java Class. Create a new Java class”. O campo “Source folder”, está preenchido por “CatalogoDeFiles/src”; abaixo dela, temos o campo: “Package”, preenchido por “br.com.lead.servlet”. Em seguida, o campo “Name” está preenchido por “JPAUtilcapt”; a caixa de seleção “Modifiers” está marcada em “public”. Em seguida, o campo “Superclass” está preenchido por: “java.lang.Object” e a caixa de seleção que aparece em seguida “Which method stubs would you like to create?”, tem a opção “Inherited abstract methods” marcada. Abaixo, há as opções “Back”, “Finish”, que está selecionada e “Cancel”.

Nessa classe, você irá criar um atributo de EntityManagerFactory que realizará a leitura do arquivo de persistência. Aproveite para criar também um método chamado getEntityManager, que deverá criar o EntityManager e retorná-lo. Feito isso, defina tanto o atributo quanto o método, como estático para que não seja necessário instanciar a classe para realizar a chamada ao método. Dessa forma, essa classe irá centralizar a criação dos objetos de EntityManager, e, sempre que precisar, no seu projeto, de um objeto de EntityManager, você deverá utilizar esse método dessa classe.

```
package br.com.lead.util;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class JPAUtil {

	static EntityManagerFactory emf =
Persistence.createEntityManagerFactory("catalogodefilmes");

	public static EntityManagergetEntityManager() {
		return emf.createEntityManager();
	}
}
```

Feito esse procedimento, vá até a classe PersisteFilmeServlet criada na aula passada,para realizar uma edição. Dentro do método service de PersisteFilmeServlet,remova o uso da EntityManagerFactory. Perceba que você irá utilizar, agora, a classeJPAUtil que acabou de criar. Como dito anteriormente, a classe JPAUtil irá centralizar a criação dos objetos de EntityManager. Então, vamos chamar o método JPAUtil.getEntityManager e atribuir o retorno ao objeto de EntityManager do método service.Fique atento, pois o método service deverá estar da seguinte forma:nas primeiras três linhas, devemos capturar os parâmetros provenientes da requisição; em seguida temos a criação do objeto filme;depois temos a criação do objeto EntityManager, utilizando, agora,JPAUtil.getEntityManager; e, após isso, temos as operações de abertura de transação com o banco e persistência do objeto filme. Por último,deve-se fechar a EntityManager.

```
@Override
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String nome = req.getParameter("nome");
        Stringgenero = req.getParameter("genero");
        Integer ano = Integer.valueOf(req.getParameter("ano"));

        Filme filme = new Filme(nome, genero, ano);

        EntityManagerem = JPAUtil.getEntityManager();

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

        em.close();
}
```

Agora, você deve alterar a classe FilmeServlet, aquela criada na videoaula 3.1 com o intuito de exibir os dados para o usuário. Essa ação é necessária para exibir a informação do filme que foi adicionado. Na classe FilmeServlet, você deve alterar o método service para que ele realize uma busca no banco de dados, a fim de resgatar a informação do objeto adicionado. Além disso, você deve também remover o trecho de código em que objetos de Filme eram criados com dados fictícios, para serem exibidos para o usuário posteriormente. Nesse ponto, você deve, agora, utilizar a comunicação com o banco de dados para capturar o objeto de filme desejado. Portanto, prossiga da seguinte maneira:

UtilizeJPAUtil.getEntityManager para que você crie uma EntityManager. Após criar uma EntityManager, vocêdeve abrir uma transação com o banco de dados; e depois da abertura da transação com o banco de dados, você deverá utilizar o método find, citado anteriormente. Nesse método, você deve passar por parâmetro o tipo da classe que está buscando, ou seja, Filme.class. O identificador da entidade será o identificador 1. Realizado o resgate da entidade, você deve fechar a EntityManager e adicionar esse objeto

na lista que deve ser exibida para o usuário. Seguindo essa descrição, o método service deverá ficar da seguinte forma:

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

        EntityManagerem = JPAUtil.getEntityManager();
        em.getTransaction().begin();

        Filme filme = em.find(Filme.class, 1);

        em.close();

        ArrayList<Filme>listaFiltrada = new ArrayList<Filme>();
        listaFiltrada.add(filme);

        req.setAttribute("listaFiltrada", listaFiltrada);

        RequestDispatcherdispatcher = req.getRequestDispatcher("lista-filmes.jsp");
        dispatcher.forward(req, resp);
}
```

Com o Tomcat inicializado, acesse agora a aplicação através da URL http://localhost:8080/CatalogoDeFilmes/filme (http, dois pontos, barra barra, localhost, dois pontos, oitenta oitenta, barra, catalogo de filmes, barra, filme),utilizando o seu navegador. Na página exibida, deve ser apresentada a informação do filme que você inseriu anteriormente no banco.

#Audiodescrição: 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/filme”. Logo abaixo, temos o texto “Lista de Filmes:
Nome: deadpol
Gênero:comedia
Ano: 2016”.

No exemplo mencionado anteriormente, quando você realizou a chamada ao método find de EntityManager, você estava trabalhando com uma persistência no estado Managed eo objeto filme estava sincronizado com a persistência do banco. Caso você alterasse algum atributo dele, como, por exemplo, nome, gênero ou ano, o valor seria refletido na base de dados, devido ao sincronismo. Lembre-se de que o estado Managed dura até o momento que se fecha o EntityManager.

Agora, reflita um pouco: se por algum motivo a conexão com o banco caiu, ou se você fechou o EntityManager, e, mesmo assim, você ainda continuou trabalhando com esse objeto filme? Nesse caso, você irá verificar o estado Detached. Esse estado ocorre quando um determinado objeto possui os mesmos dados que identificam uma persistência no banco de dados. Porém, esse objeto não possui mais a conexão com o banco. Ressalta-se que o importante para identificar a persistência na base de dados é sua chave primária, ou seja, no seu caso, o ID do objeto. Nesse momento, se você ainda está tentando persistir um objeto que esteja no estado Detached, saiba que o Eclipse irá lançar um erro na aplicação que está sendo desenvolvida.

“

Caused by: org.hibernate.PersistentObjectException: detached entity passed to persist...

”

Um outro estado de persistência é o Transient, que se caracteriza por um novo objeto que não tem nenhuma referência na base dados. Por exemplo, no momento de realizar a persistência do objeto no banco de dados, o desenvolvedor deve criar o objeto previamente. Até o momento anterior, a chamada ao método persist do EntityManager, o objeto não possui nenhuma referência no banco. Então, ele encontra-se no estado Transient. Após a chamada ao método persist, o objeto entra no estado Managed.Assim, no método service, da classe PersisteFilmeServlet, é possível verificar isso:

```
@Override
protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        String nome = req.getParameter("nome");
        Stringgenero = req.getParameter("genero");
        Integer ano = Integer.valueOf(req.getParameter("ano"));
```

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

        EntityManagerem = JPAUtil.getEntityManager();

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

        em.close();
}
```

Por último, tem-se o estado Deleted ou Removed. Esse estado ocorre a partir do momento que você realiza a remoção, isto é, a exclusão da entidade. Pode-se realizar esse procedimento chamando o método “remove” de EntityManager, passando como parâmetro a entidade que deseja remover. Ah, vale salientar que, para chamar o método “remove”, se faz necessário ter uma transação aberta com o banco de dados através do EntityManager, e o objeto deve estar sincronizado com a entidade do banco antes de chamar o método “remove”. Esse sincronismo pode ocorrer chamando o método find, como citado anteriormente. Caso seja realizado a chamada ao método remove, sem antes ter realizado o sincronismo com a entidade a ser deletada, não será possível deletar o objeto, e, como consequência, o Eclipse irá enviar um erro. Agora, para fixar o que foi explicado, segue um exemplo da utilização do método remove:

```
EntityManagerem = JPAUtil.getEntityManager();
em.getTransaction().begin();

Filme filme = em.find(Filme.class, 1);

em.remove(filme);

em.close();
```

Até aqui, você aprendeu sobre o EntityManager e sobre os estados de Persistência do Hibernate, que são: Managed, Detached, Transient e Deleted. Além disso, você conheceu como detectar o momento em que uma persistência se encontra em cada estado. Com isso, você pôde avançar um pouco com o projeto CataloDeFilmes; separou a criação da EntityManager; e realizou a apresentação do objeto persistido no banco.

Por fim, vale ressaltar que as ferramentas utilizadas na aula possuem outras aplicações e utilidades, assim é importante que você se aprofunde nos estudos das tecnologias apresentadas para um melhor desempenho, certo?
No mais, bons estudos!

Referências

Definindo Entity Manager na Java Persistence API. Disponível em: <https://www.devmedia.com.br/definindo-entity-manager-na-java-persistence-api/28271>.Acesso em: 10maio 2020.

HibernateEntityLifecycle. Disponível em: <https://www.baeldung.com/hibernate-entity-lifecycle>. Acesso em: 10maio 2020.