





37%

ATIVIDADES

9 de 10

FÓRUM DO

CURSO

VOLTAR

PARA

DASHBOARD

MODO

NOTURNO

ABRIR

CADERNO

F

115.9k xp

a

## Vamos migrar agora todo o sistema para usar o MySQL?

1) Abra a classe `LivroDao` , remova a linha que declara o Banco e injete o `EntityManager`:

```
// private Banco banco = new Banco();
```

```
@PersistenceContext
private EntityManager manager;
```

COPIAR CÓDIGO

2) Ainda em `LivroDao` , substitua a utilização do banco *fake* nos métodos pelo `EntityManager`:

```
public void salva(Livro livro) {
    // banco.save(livro);
    manager.persist(livro);
}
```

```
public List<Livro> todosLivros() {
    // return banco.listaLivros();
    return manager.createQuery("select
l from Livro l", Livro.class)
        .getResultList();
}
```

COPIAR CÓDIGO

3) Faça o mesmo na classe `UsuarioDao` , porém, tenha atenção ao método `buscaPeloLogin()` onde a busca pelo usuário no banco de dados deve ser feita pelo *String* login. No nosso exemplo temos o JPA usando JPQL para a pesquisa. DICA: Se quiser saber mais sobre JPQL, procure no Alura o treinamentos específico de JPA.

```
public Usuario buscaPeloLogin(String
login) {
    // return
    this.banco.buscaPeloNome(login);

    Usuario usuario = (Usuario)
    this.manager
```



37%

ATIVIDADES

9 de 10

FÓRUM DO CURSO

VOLTAR PARA DASHBOARD



MODONOTURNO



ABRIR CADERNO

```
        .createQuery("select u from
Usuario u where u.login=:pLogin")
        .setParameter("pLogin",
login).getSingleResult();
        return usuario;
    }
}
```

COPIAR CÓDIGO

4) Caso não tenha feito ainda anote a classe LoginBean com @Named e @RequestScoped (do pacote javax.enterprise.context ) e injete o DAO:

```
@Named
@RequestScoped
public class LoginBean {

    private Usuario usuario = new Usuario();

    @Inject
    private UsuarioDao dao;
}
```

COPIAR CÓDIGO

Observação: As anotações @Named e @RequestScoped vem de uma outra especificação do Java EE, vem do CDI (Context and Dependency Injection). As especificações CDI e EJB são bem parecidas no sentido que ambos fazem Inversão de controle e Injeção de dependências. A diferença é que o CDI não sabe, por padrão pelo menos, gerenciar o JPA e a transação e não oferece vários outros serviços do mundo EJB (como remotabilidade ou timer service). EJB e CDI são de certa forma concorrentes dentro do JavaEE, sendo EJB um pouco mais completo e o CDI um pouco mais flexível/simples. Na nossa aplicação o CDI tem o papel de juntar a interface JSF com EJB. O CDI é o intermediário entre os dois.

No Alura temos alguns treinamentos que abordam o uso do CDI dentro de uma aplicação web, por exemplo:

Vraptor: <https://cursos.alura.com.br/course/desenvolvimento-web-com-vraptor-4> (https://cursos.alura.com.br/course/desenvolvimento-web-com-vraptor-4) JSF III: <https://cursos.alura.com.br/course/jsf-cdi> (https://cursos.alura.com.br/course/jsf-cdi)

5) Voltando para o nosso objeto, ao migrarmos para o MySQL não teremos mais disponível o usuário admin , cadastrado no Banco fake (em memória) que usávamos para os testes.



37%

ATIVIDADES  
9 de 10

FÓRUM DO  
CURSO

VOLTAR  
PARA  
DASHBOARD



MODO  
NOTURNO



ABRIR  
CADERNO


Precisaremos cadastrar algum usuário na tabela Usuario do banco de dados para conseguir acessar o sistema. Para isso, acesse o terminal de comando do MySQL e execute o comando de INSERT :

```
$ mysql -u root

mysql> use livraria;
Database changed
mysql> INSERT INTO Usuario (login,senha)
VALUES ('admin','pass');
Query OK, 1 row affected (0,00 sec)
mysql> quit
```

COPIAR CÓDIGO

6) Agora você já pode reiniciar seu servidor no eclipse, recarregar sua aplicação no navegador e acessar o sistema com as informações de login e senha que acabamos de inserir no MySQL.



115.9k xp

