08

Mãos a obra: Injetando as dependências

Vamos começar injetando as dependências na classe `LoginBean`.

Utilizando um dos três pontos de injeção, injete a dependência do `UsuarioDao` na classe `LoginBean`:

```
private UsuarioDao usuarioDao;

@Inject
public LoginBean(UsuarioDao usuarioDao){
    this.usuarioDao = usuarioDao;
}
```

Ou

```
@Inject
private UsuarioDao usuarioDao;
```

Ou

```
private UsuarioDao usuarioDao;

@Inject
public void inicializador(UsuarioDao usuarioDao){
    this.usuarioDao = usuarioDao;
}
```

Feito isso substitua todos os pontos em que estamos instanciando um objeto de `UsuarioDao`, para usar o atributo `this.usuarioDao`.

Vamos agora ensinar ao **CDI** como ele deve produzir nosso `DAO` genérico e o `EntityManager`.

Vamos começar ensinando ao **CDI** como criar `EntityManager`:

Na classe `JPAUtil` anote o método `getEntityManager` com `@Produces` (para que o **CDI** utilize esse método para criar um `EntiManager`), e além disso vamos anotar também com `@RequestScoped` (para que nossos `EntityManager` s estejam disponíveis somente durante uma requisição.).

Fora isso vamos ensinar para o **CDI** como descartar corretamente o `EntityManager` com a anotação `@Disposes`.

```
public class JPAUtil {

    private static EntityManagerFactory emf = Persistence.createEntityManagerFactory("livraria");

    @Produces
    @RequestScoped
```

```
    public EntityManager getEntityManager(){
        return emf.createEntityManager();
    }

    public void close(@Disposes EntityManager em){
        if( em.isOpen()) {
            em.close();
        }
    }

}
```

Agora vamos ensinar o **CDI** como produzir nosso `DAO`.

Crie a classe `DAOFactory` e nela crie um método que retorne um `DAO` genérico. Utilize o `InjectionPoint` para recuperar a classe que deve ser informada no construtor do `DAO`.

```
@SupressWarnings("unchecked")
public class DAOFactory {

    @Inject
    private EntityManager manager;

    @Produces
    public <T> DAO<T> factory(InjectionPoint point){
        ParameterizedType type = (ParameterizedType) point.getType();

        Class<T> classe = (Class<T>) type.getActualTypeArguments()[0];

        return new DAO<T>(classe, manager);

    }
}
```

Dentro da classe `DAO` remova todas as declarações que instanciam um `JPAUtil` para pegar o `EntityManager` e fechar a conexão do mesmo. Além disso receba o `EntityManager` pelo construtor.

Agora altere todas as classes `Beans` para receberem o `DAO` injetado.