06

Mãos a obra: Alternativos

Em nossa biblioteca temos uma interface `Transacionado`, então vamos mudar nossa classe `MeuGerenciadorDeTransacao` para que ele implemente a interface ao invés de estender a classe `TransacionadoPadrao`.

```
public class MeuGerenciadorTransacao implements Transacionado {

    @Inject
    private EntityManager em;

    public Object executaComTransacao( InvocationContext context ) {
        System.out.println("Antes de abrir a transação");
        em.getTransaction().begin();

        try {
            System.out.println("Antes de executar o método interceptado");
            Object resultado = context.proceed();

            System.out.println("Antes de efetuar o commit da transação");
            em.getTransaction().commit();

        } catch( Exception e ) {
            System.out.println("Antes de efetuar o rollback da transação");
            em.getTransaction().rollback();
            throw new RuntimeException(e);
        }
    }

}
```

Porém se deixarmos nossa classe assim teremos uma ambiguidade, pois temos duas classes que respondem por `Transacionado` (`TransacionadoPadrao` e `MeuGerenciadorDeTransacao`).

Para evitar isso podemos usar uma outra funcionalidade do **CDI** que é os `Alternatives`. Um `Bean` alternativo tem uma prioridade maior sobre os `Bean` s não alternativos, ou seja, se tivermos 10 classes que respondem à um mesmo tipo e uma delas for um `Bean` alternativo, o **CDI** irá usar o `Bean` alternativo ao invés das outras 9 classes.

Para transformar nosso `bean` em um `bean` alternativo basta anotar nossa classe com `@Alternatives` e ativa-lo no arquivo `beans.xml` ou anotar a classe com `@Priority` (da mesma forma que fizemos com interceptor).

Para ativa-lo através do arquivo `beans.xml` utilize as seguintes `tag` s:

```
<alternatives>
    <class>br.com.alura.livraria.tx.MeuGerenciadorDeTransacao</class>
</alternatives>
```

O único problema com essa abordagem é que usando o arquivo `beans.xml` ele ativa o `bean` alternativo somente no projeto que estamos usando, ou seja, esse `bean` não é uma alternativa para as bibliotecas que temos no nosso projeto. (E nesse caso o **CDI** irá usar a classe `TransacionadoPadrao` para suprir a dependência da interface `Transacionado`.)

Quando usamos a anotação `@Priority` ativamos esse `bean` alternativo de uma forma global, ou seja, ele será uma alternativa tanto no nosso projeto quanto para as nossas bibliotecas.