

20%

ATIVIDADES

7 de 13


FÓRUM DO

CURSO

VOLTAR


PARA

DASHBOARD




MODO

NOTURNO




ABRIR

CADERNO



114.6k xp



Vamos entender na prática o ciclo de vida dos EJBs?

1) Abra a classe `AutorDao` e crie um método chamado `aposCriacao` que retorne `void` e escreva no console `"[INFO] AutorDao foi criado."`. Anote o novo método com `@PostConstruct`:

```
@PostConstruct
void aposCriacao() {
    System.out.println("[INFO] AutorDao foi criado.");
}
```

COPIAR CÓDIGO

2) Dentro do método `salva()`, coloque uma saída no console antes e depois do comando para salvar o `Autor`. Depois faça com que a *thread* de execução aguarde 20s. Não esqueça de fazer o tratamento de erros:

```
public void salva(Autor autor) {
    System.out.println("[INFO] Salvando o Autor " + autor.getNome());

    try {
        Thread.sleep(20000); // 20 segundos
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    banco.save(autor);
    System.out.println("[INFO] Salvou o Autor " + autor.getNome());
}
```

COPIAR CÓDIGO

3) Faça o *Full Publish*, depois abra a aplicação em duas abas diferentes do navegador e adicione um novo autor em cada uma das abas dentro do tempo de 20 segundos de espera da *thread*.

4) Observando as mensagens no console da aplicação, o que

você percebeu do comportamento do EJB nessa situação?



20%

ATIVIDADES
7 de 13

FÓRUM DO
CURSO

VOLTAR
PARA
DASHBOARD



Opinião do instrutor



Ao salvar pela segunda vez o Autor, antes da primeira *thread* ter terminado sua execução, o *EJB Container* criou mais um `AutorDao` para atender a execução do segundo *thread*. Ou seja, como o primeiro *EJB Session Bean* `AutorDao` estava em uso, o *EJB Container* decidiu criar mais um para atender a outra chamada de execução.

Esse exercício mostra que um *EJB Session Bean* não é compartilhado entre *threads*, por isso dizemos que ele é *Thread Safe*. *Thread Safety* é mais um serviço que ganhamos ao usar os EJBs.



114.6k xp

