

ATIVIDADES  
9 de 13

FÓRUM DO  
CURSO

VOLTAR  
PARA  
DASHBOARD

MODOS  
NOTURNOS

ABRIR  
CADERNO

## Vamos transformar a classe Banco em um *Session Bean*?

1) Abra a classe `Banco` que ainda não é um EJB. Como não faz sentido termos mais de um objeto dessa classe no *pool* de EJBs, e como o JBoss Wildfly AS está configurado para manter 20 instâncias dos *Session Beans* em memória, vamos transformá-lo em um EJB especial para não modificarmos em nada a configuração do servidor. Anote a classe `Banco` com `@Singleton` do pacote `javax.ejb.*`.

```
import javax.ejb.Singleton;

@Singleton
public class Banco {
    ...
}
```

COPIAR CÓDIGO

2) Na classe `AutorDao`, vamos injetar o objeto `banco` ao invés de instanciá-lo. Anote a declaração do atributo com `@Inject`:

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

COPIAR CÓDIGO

3) Para termos certeza que apenas uma instância do *Singleton* `Banco` será criada, usaremos os *callbacks* para realizar um teste. Crie um método chamado `aposCriacao()` que não retorne nada e imprima no console uma mensagem. Anote-o com `@PostConstruct`:

```
@PostConstruct
void aposCriacao() {
    System.out.println("[INFO] O Banco acabou de ser criado.");
}
```

COPIAR CÓDIGO

4) Reinicie o servidor JBoss AS e verifique que o *EJB Container* já carregou a classe `Banco`:

ATIVIDADES

9 de 13

FÓRUM DO

CURSO

VOLTAR

PARA

DASHBOARD

MODO

NOTURNO

ABRIR

CADERNO

```
java:global/livraria
/Banco!br.com.caelum.livraria.dao.Banco
java:app/livraria
/Banco!br.com.caelum.livraria.dao.Banco
java:module
/Banco!br.com.caelum.livraria.dao.Banco
java:global/livraria/Banco
java:app/livraria/Banco
java:module/Banco
```

COPIAR CÓDIGO

5) Agora, limpe o console do eclipse e acesse a aplicação pelo navegador. Observe que a mensagem da criação do banco apareceu no console. Então adicione um novo autor para ter certeza que nenhuma nova instância do banco será criada.

6) Aproveite agora e modifique as classes `LivroDao` e `UsuarioDao` para não mais instanciar a classe `Banco`, mas sim recebê-la por injeção como na classe `AutorDao`.

7) Vamos também garantir que o *Singleton* `Banco` seja carregado já na inicialização do JBoss AS, e não sob demanda da aplicação. Para isso, acrescente a anotação `@Startup` à classe `Banco` e faça um *Full Publish* no servidor:

```
@Singleton
@Startup
public class Banco {
    ...
}
```

COPIAR CÓDIGO

8) Qual a vantagem de carregar um *Session Beans* do tipo *Singleton* desde a inicialização do servidor? Como essa inicialização se chama?

*ATENÇÃO:* Se você alterou a classe `UsuarioDao` para receber a classe `Banco` por injeção você deve modificar a classe `LoginBean` para receber o EJB `UsuarioDao` também por injeção. Você também pode alterar a classe `AutorBean` para receber o EJB `AutorDAO` por injeção ao invés de instanciá-lo. Em ambos os casos basta remover o operador `new` e anotar a declaração do atributo com `@Inject`.

```
@Inject
```

private UsuarioDao dao; // = new UsuarioDao();

COPIAR CÓDIGO

ATIVIDADES  
9 de 13

FÓRUM DO CURSO

VOLTAR PARA DASHBOARD

Opinião do instrutor

MODO NOTURNO

ABRIR CADERNO

Eles são úteis principalmente para inicializar alguma configuração ou agendar algum serviço, coisas que só fazem sentido no início da aplicação, ou seja, quando o JBoss AS carrega a aplicação já queremos que o *Session Bean* seja criado para carregar todas as configurações.

Essa inicialização também é chamada de *Eager Initialization*.

F

114.8k xp

a