



## 07 Cadeia de interceptadores

PRÓXIMA ATIVIDADE



64%

ATIVIDADES  
7 de 11FÓRUM DO  
CURSOVOLTAR  
PARA  
DASHBOARD

Dada a implementação seguinte:

```
//imports omitidos
public class LogInterceptador {

    @AroundInvoke
    public Object loga(InvocationContext ctx)
    throws Exception {

        System.out.print("1");

        Object retorno = ctx.proceed();

        System.out.print("2");

        return retorno;
    }
}
```

COPIAR CÓDIGO

```
public class AuditoriaInterceptador {

    @AroundInvoke
    public Object audita(InvocationContext ctx)
    throws Exception {

        System.out.print("3");

        System.out.print("4");

        return null;
    }
}
```

COPIAR CÓDIGO

```
@Interceptors({LogInterceptador.class,
AuditoriaInterceptador.class})
@Stateless
public class ContaDao{

    public void salva(Conta conta) {
        System.out.print("5");
    }
}
```

COPIAR CÓDIGO



117.8k xp



Assumindo que o código compila e foi publicado corretamente,



64%

ATIVIDADES  
7 de 11FÓRUM DO  
CURSOVOLTAR  
PARA  
DASHBOARD

qual seria a saída ao chamar o método `salva(...)` da classe `ContaDao` ?

**A**

13542



1342

**C**

12345

**D**

12543

**E**

134

Nesse exemplo, a execução começa com o `LogInterceptor` e vai imprimir **1**. Depois de chamar `proceed()` a execução vai para o próximo interceptador e entra no método `audita(...)` da classe `AuditoriaInterceptor`. Nesse método imprimimos **3** e **4**. Repare que não há a chamada do método `proceed()`. Ou seja, nunca vamos chamar o próximo na cadeia. A execução não vai chegar a classe `ContaDao`, por isso retornamos para a classe `LogInterceptor` e imprimimos **2**. O resultado final é **1342**.

Como vimos, interceptadores podem ser chamados em uma cadeia. É bom pensar que um interceptador é como uma *casca* do Session Bean. Faz parte dele e podemos adicionar quantas cascas quisermos.

PRÓXIMA ATIVIDADE



117.8k xp

