Chegou a hora de trabalharmos com interceptadores! Vamos começar monitorando o tempo de acesso ao Banco de Dados?

1) Crie uma nova classe chamada `LogInterceptador` no pacote `br.com.caelum.livraria.interceptador`.

2) Ela terá apenas um método que chamaremos `intercepta()` e retornará um `Object`.

```
package br.com.caelum.livraria.interceptador;

public class LogInterceptador {

    public Object intercepta() {

    }

}
```

COPIAR CÓDIGO

3) O método `intercepta()` fará o monitoramento do tempo de acesso ao banco de dados pegando o tempo quando o método é invocado e subtraindo do tempo quando o método é finalizado. Além disso, entre essas duas ações de monitoramento, um método do DAO deverá ser chamado para de fato termos a ida ao banco de dados. Usaremos o objeto especial que está sempre disponível dentro do interceptador chamado `InvocationContext`. Através dele podemos continuar a execução da aplicação, ou seja, chamar o método no DAO. Para isso, temos o método `proceed()` que prossegue com o método interceptado. **ATENÇÃO:** Ao chamar o método `proceed()` é necessário tratar uma exceção que faremos através de `throws`.

```
    public Object intercepta(InvocationContext
contex) throws Exception {

        long millis =
System.currentTimeMillis();

        Object o = contex.proceed();

        System.out.println("[INFO] Tempo gasto
no acesso ao BD: "
                + (System.currentTimeMillis()
```

```
- millis) + "ms");

        return o;

    }
```

COPIAR CÓDIGO

4) Também é necessário anotar o método com `@AroundInvoke` para deixar claro ao *EJB Container* que o método realmente intercepta o fluxo.

```
    @AroundInvoke
    public Object intercepta(InvocationContext
contex) throws Exception {

        ...
    }
```

COPIAR CÓDIGO

5) Agora, anotaremos com `@Interceptors` as classes que devam ser interceptadas e como parâmetro da anotação o nome da classe interceptadora (`LogInterceptador.class`). Comece interceptando a classe `AutorDao` :

```
@Interceptors({ LogInterceptador.class })
public class AutorDao {
    ...
}
```

COPIAR CÓDIGO

6) Faça o *Full Publish* da aplicação, limpe o console do Eclipse e teste a aplicação.

7) Nesse exercício é possível identificar com segurança qual método foi chamado para gerar os tempos de acesso ao banco de dados?

Opinião do instrutor

Não! Para ter segurança na identificação seria preciso imprimir também o nome do método antes da saída com o tempo de acesso.