03

Evitando conflitos com Lock otimista

Transcrição

Uma solução otimista consiste em permitir que os conflitos ocorram, detectando-os e tomando medidas para corrigi-los. A ideia é que a aplicação/banco nunca trave o acesso ao registro, sendo otimista, assumindo que isso é uma situação não tão frequente. Mas o que faremos quando acontece uma atualização concorrente? Temos que prevenir a inconsistência dos dados, lançando uma *exception* ao cliente.

Usando @Version

A JPA oferece suporte a *lock otimista* através da anotação `@Version`. Para fazer uso do *lock otimista* devemos criar um campo de versionamento que use essa anotação:

```
@Entity
public class Produto {
    // ...

    @Version
    private Integer versao; // não esqueça do getter e setter

    // ...
}
```

Para guardar a versão, podemos usar um campo numérico ou um timestamp (Calendar ou Date). Se a entidade possuir `@Version` em todo update feito, o hibernate irá verificar automaticamente o valor desse campo. Caso o registro no banco possua um valor menor do que o está sendo enviado para o campo `versao`, ele aceita a atualização e incrementa seu valor. Caso possua um valor maior, será disparado uma exceção do tipo `StaleObjectStateException` dentro de uma `javax.persistence.OptimisticLockException`.

Ou seja, se duas pessoas tentarem atualizar o mesmo registro, a primeira pessoa conseguirá atualizar incrementando seu valor. A última será impedida de atualizar já que tentará enviar um valor inferior do que está no banco.

Testando o Lock Otimista

Vamos ver isso na prática? Como estamos trabalhando na web, vamos precisar enviar na página de edição o valor do campo `versao`. Portanto, na página `produto/form.jsp` precisaremos adicionar um campo `hidden` para enviar esse valor:

```
<input type="hidden" name="versao" value="${produto.versao}">
```

- Para testarmos escolha um produto e clique no campo editar;
- Abra uma nova aba e vá até a página de edição do mesmo produto;
- Faça uma alteração no segundo produto e salve;
- Volte à primeira aba e tente atualizar.

Ao testar esse exemplo recebemos uma exceção do tipo `StaleObjectStateException` como já conversamos anteriormente. Repare que o problema de concorrência ocorreu, porém o próprio Hibernate já tomou uma medida para evitar a inconsistência no nosso banco! Nesse caso, poderíamos redirecionar o usuário para uma tela amigável de erro.