## **Auditoria Avançada com Hibernate Envers (Easy Entity Versioning)**

Desde pequenas a aplicações de grande porte, a necessidade de monitorar as ações realizadas por um usuário frente as diversas funcionalidades existentes são reais. Em um sistema de uma instituição financeira, por exemplo, existe inclusive uma motivação legal para que seja registrada todas as alterações realizadas nos saldos das contas para uma eventual análise posterior.

O Hibernate Envers oferece a organização do histórico das versões dos dados trabalhados pela aplicação, através das entidades mapeadas para a persistência JPA para auditar as modificações ocorridas em um dado registro. Dessa forma, com sua utilização, uma aplicação é capaz de gerir todas as modificações realizadas no seu database de forma fácil e não intrusiva.

O Hibernate Envers é uma biblioteca que permite criar facilmente, auditoria de classes persistentes através do controle de versões de persistência em mapeamentos objetos relacionais feitos através do Hibernate.

Como vantagens da utilização desse framework podemos citar: permite auditar todas as entidades mapeadas pelo Hibernate; baseado em revisões; independência do fabricante para o banco de dados; agrega valor ao produto; redução no custo de manutenção e maior produtividade.

**Configuração**

Para ter acesso às funcionalidades do **Envers** é necessário adicionar a dependência no **pom.xml**

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-envers</artifactId>

<version>5.2.10.Final</version>

<scope>provided</scope>

</dependency>

**Criando, mapeando e auditando**

Através de anotações no código, a partir de agora, vamos criar e definir o mapeamento objeto/relacional de nossa entidade, além de sinalizar ao Envers qual entidade sofrerá auditoria.

@Entity

@EntityListeners(ProdutoListener.**class**)

@Audited(targetAuditMode=RelationTargetAuditMode.***NOT\_AUDITED***)

@AuditTable(value = "aud\_Produto")

**public** **class** Produto{

**Auditando informações extras**

Para isso, vamos utilizar-se do recurso de escuta de revisão do *framework* e criar uma classe que representará a entidade de revisão do projeto, sendo responsável por mapear todos os dados que queremos armazenar no momento da criação de uma nova revisão.

Sendo assim, dentro do pacote br.ufpi.carrinhoCompras.model, crie a classe **AuditEntity**.

@Entity(name = "revinfo\_cust")

@RevisionEntity(AuditListener.**class**)

**public** **class** AuditEntity **extends** DefaultRevisionEntity {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@GeneratedValue(strategy=GenerationType.***IDENTITY***)

**private** **int** id;

**private** String usuario;

**private** String ip;

}

Na linha 13 a classe foi mapeada como uma entidade JPA, através da *anotation* **@Entity**. Além disso, é feita a alteração do nome da tabela central de auditoria do Envers, que, por padrão, leva o nome de **"revinfo"**. Através do atributo "**name**" foi definido o novo nome como sendo "**revinfo\_cust**".

A anotação **@RevisionEntity**, presente na linha 14, indica que essa é uma entidade de revisão e que será usada para armazenar o histórico das revisões. Nessa mesma linha é feito menção a classe **AuditListener,** que vai ser um “interceptor” da classe **RevisionListener** e implementará o comportamento customizado que será utilizado no momento da criação da revisão.

Os novos campos que serão inseridos na tabela de revisão aparecem definidos nas linhas 22 e 23. Os demais campos, que são *default* do framework, são incorporados através da classe **DefaultRevisionEntity**, estendida na linha 15.

Outra classe que deve ser criada é a **AuditListener** que estende **AuditEventListener**, como mostra abaixo:

**public** **class** AuditListener **implements** RevisionListener {

@Override

**public** **void** newRevision(Object revisionEntity) {

FacesContext context = FacesContext.*getCurrentInstance*();

HttpSession session = (HttpSession) context.getExternalContext().getSession(**false**);

HttpServletRequest request = (HttpServletRequest) context.getExternalContext().getRequest();

Cliente cliente = (Cliente) session.getAttribute("usuarioLogado");

AuditEntity revEntity = (AuditEntity) revisionEntity;

revEntity.setUsuario(cliente.getNome());

revEntity.setIp(request.getRemoteAddr());

}

}

Nesta classe**,**o método **newRevision** – linha 15 – foi sobrescrito. O mesmo recebe como parâmetro a entidade que está sendo auditida, que no nosso caso é o **Produto**. Esse método é chamado toda vez que o Envers vai criar uma nova revisão, assim, podemos instanciar nossa classe **AuditEntity** e definir os atributos que queremos nela.

Nas linhas 24 e 25 adicionei, o usuário que está realizando a alteração, e o IP de sua máquina. Os atributos que vêm da **DefaultRevisionEntity** (id e timestamp), são preenchidos automaticamente pelo Envers.

**Referências**

Site oficial do Hibernate Envers: <http://hibernate.org/orm/envers/>

**Tópicos**: <http://www.oracle.com/technetwork/pt/articles/java/auditoria-avancada-hibernate-envers-3399218-ptb.html>