
Prog. Orientada a Objetos e Mapeamento Objeto-Relacional – IMD0104

Interceptors

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

Interceptors

- ❑ A interface **Interceptor** fornecer informações sobre um objeto persistente antes de ser **salvo**, **atualizado**, **excluído** ou **carregado**.
 - ❑ Um possível uso para essa interface é gerar **informações de auditoria**.
 - ❑ O **Interceptor** fornece automaticamente a **createTimestamp** quando um objeto é criado e atualiza a função **lastUpdateTimestamp** quando um objeto é atualizado.
-

Logging

- A aplicação pode logar as informações em qualquer formato:
 - ❖ **Arquivo:** método mais rápido, porém, de difícil recuperação de dados.
 - ❖ **Banco de Dados:** um pouco mais lento, porém mais eficiente e estruturado para busca.

Métodos Interceptores

- ❑ @PrePersist/@PostPersist - A entidade **antes** e **depois** de ser persistido.
- ❑ @PreLoad - Antes da entidade ser carregada.
- ❑ @PreUpdate/@PostUpdate - A entidade **antes** e **depois** de ser alterada.
- ❑ @PreRemove/@PostRemove - A entidade **antes** e **depois** de ser removido.

Utilizando o Hibernate

- ❑ O hibernate oferece um recurso de interceptar mudanças, através da extensão da classe:
 - ❖ **EmptyInterceptor**
- ❑ Definição do interceptor é feita no arquivo ***persistence.xml***.

Configurando o Interceptor

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.0"
3     xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence.xml"
5     <persistence-unit name="ConexaoDB" transaction-type="RESOURCE_LOCAL">
6         <provider>org.hibernate.ejb.HibernatePersistence</provider>
7         <!-- <class>model.Contato</class> -->
8         <shared-cache-mode>ENABLE_SELECTIVE</shared-cache-mode>
9         <properties>
10
11         <property name="hibernate.ejb.interceptor" value="interceptor.InterceptorPadrao" />
12
13         <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5432/projetojpa" />
14         <property name="javax.persistence.jdbc.password" value="jcx1" />
15         <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
16         <property name="javax.persistence.jdbc.user" value="jcx1" />
17         <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider" />
18         <property name="hibernate.hbm2ddl.auto" value="update" />
19         <property name="hibernate.show_sql" value="true" />
20
21     </properties>
22 </persistence-unit>
23 </persistence>
```

Métodos utilizados

```
public class InteceptorPadrao extends EmptyInterceptor{

    private static final long serialVersionUID = 1L;

    @Override
    public boolean onSave(Object entity, Serializable id, Object[] state,
        String[] propertyNames, Type[] types) {
        return super.onSave(entity, id, state, propertyNames, types);
    }

    @Override
    public void onDelete(Object entity, Serializable id, Object[] state,
        String[] propertyNames, Type[] types) {
        super.onDelete(entity, id, state, propertyNames, types);
    }

    @Override
    public boolean onLoad(Object entity, Serializable id, Object[] state,
        String[] propertyNames, Type[] types) {
        return super.onLoad(entity, id, state, propertyNames, types);
    }
}
```

Implementação do Interceptor

```
17 public class InterceptorPadrao extends EmptyInterceptor{
18
19     private static final long serialVersionUID = 1L;
20     private String tipoOperacao;
21     private Set<Contato> contatosSalvos = new HashSet<Contato>();
22     private Set<Contato> contatosAtualizados = new HashSet<Contato>();
23     private Set<Contato> contatosExcluidos = new HashSet<Contato>();
24
25     GenericDAO gdao = new GenericDAO();
26
27     public boolean onSave(Object entity, Serializable id, Object[] state,
28         String[] propertyNames, Type[] types){
29
30         if (entity instanceof Contato) {
31             Contato ent = (Contato) entity;
32             contatosSalvos.add(ent);
33         }
34         tipoOperacao = "Save";
35
36         return super.onSave(entity, id, state, propertyNames, types);
37     }
```


Implementação do Interceptor

```
public boolean onFlushDirty(Object entity, Serializable id,
    Object[] currentState, Object[] previousState, String[] propertyNames,
    Type[] types) {
    if (entity instanceof Contato) {
        Contato ent = (Contato) entity;
        contatosAtualizados.add(ent);
        tipoOperacao = "Update";
        return true;
    }
    return false;
}
```

```
public void onDelete(Object entity, Serializable id, Object[] state,
    String[] propertyNames, Type[] types){

    if (entity instanceof Contato) {
        Contato ent = (Contato) entity;
        contatosExcluidos.add(ent);
    }

    tipoOperacao = "Delete";
    super.onDelete(entity, id, state, propertyNames, types);
}
```

Implementação do Interceptor

```
@Override  
public void postFlush(@SuppressWarnings("rawtypes") Iterator entities) {  
    super.postFlush(entities);
```

```
    // Para ação de inserção  
    for (Contato cont : contatosSalvos) {  
        LogDatabase log = new LogDatabase();  
        log.setEntidade(cont.getClass().getCanonicalName());  
        log.setTipoOperacao(tipoOperacao);  
        log.setValorAlterado(cont.getId().toString());  
        log.setDataAlteracao(new Date());  
        gdao.inserirLog(log);  
    }
```

```
    // Para ação de alteração  
    for (Contato cont : contatosAtualizados) {  
        LogDatabase log = new LogDatabase();  
        log.setEntidade(cont.getClass().getCanonicalName());  
        log.setTipoOperacao(tipoOperacao);  
        log.setValorAlterado(cont.getId().toString());  
        log.setDataAlteracao(new Date());  
        gdao.inserirLog(log);  
    }
```

Implementação do Interceptor

```
@Override  
public void postFlush(@SuppressWarnings("rawtypes") Iterator entities) {  
    super.postFlush(entities);
```

```
// Para ação de exclusão  
for (Contato cont : contatosExcluidos) {  
    LogDatabase log = new LogDatabase();  
    log.setEntidade(cont.getClass().getCanonicalName());  
    log.setTipoOperacao(tipoOperacao);  
    log.setValorAlterado(cont.getId().toString());  
    log.setDataAlteracao(new Date());  
    gdao.inserirLog(log);  
}
```

```
contatosSalvos.clear();  
contatosAtualizados.clear();  
contatosExcluidos.clear();  
}
```

Classes de Domínio

```
12 @Entity
13 public class Contato {
14
15     @Id
16     @GeneratedValue(strategy=GenerationType.SEQUENCE)
17     private Long id;
18
19     private String nome;
20     private String telefone;
21
22     @Temporal(TemporalType.DATE)
23     private Date dataNascimento;
```

Classes de Domínio

```
10 @Entity
11 public class LogDatabase {
12     @Id
13     @GeneratedValue(strategy=GenerationType.SEQUENCE)
14     private Long id;
15     private String entidade;
16     private String valorAlterado;
17     private String tipoOperacao;
18     private Date dataAlteracao;
19
20     // Getters and Setters
```

Classe GenericDAO

```
7 public class GenericDAO {
8
9     public void inserir (Object entidade) {
10         EntityManager em = getEntityManager();
11         em.getTransaction().begin();
12         em.persist(entidade);
13         em.getTransaction().commit();
14     }
15
16     public void alterar (Object entidade) {
17         EntityManager em = getEntityManager();
18         em.getTransaction().begin();
19         em.merge(entidade);
20         em.getTransaction().commit();
21     }
22
23     public void deletar (Object entidade) {
24         EntityManager em = getEntityManager();
25         em.getTransaction().begin();
26         em.remove(entidade);
27         em.getTransaction().commit();
28     }
```

Classe GenericDAO

```
public class GenericDAO {  
    public void inserir (Object entidade){  
    }  
    public void alterar (Object entidade){  
    }  
    public void deletar (Object entidade){  
    }  
  
    private EntityManager getEntityManager(){  
        return Banco.getInstance().getEntityManager();  
    }  
  
    public void inserirLog(Object log){  
        EntityManagerFactory emf =  
            Persistence.createEntityManagerFactory("ConexaoDB");  
        EntityManager em = emf.createEntityManager();  
        try {  
            em.getTransaction().begin();  
            em.persist(log);  
            em.getTransaction().commit();  
        }  
        finally {  
            em.close();  
        }  
    }  
}
```

Dúvidas...



Testando a Aplicação

```
11 public class Principal {
12
13     private static EntityManager em;
14
15     public static void main(String[] args) {
16
17         Calendar data = Calendar.getInstance();
18         GenericDAO gdao = new GenericDAO();
19
20         Contato c1 = new Contato();
21         c1.setNome("Eduardo Cunha");
22         c1.setTelefone("8888-3333");
23         c1.setDataNascimento(data.getTime());
24
25         Contato c2 = new Contato();
26         c2.setNome("Waldir Maranhão");
27         c2.setTelefone("9999-4444");
28         c2.setDataNascimento(data.getTime());
29
30         gdao.inserir(c1);
31         gdao.inserir(c2);
```

Testando a Aplicação

Edit Data - PostgreSQL 9.3 (localhost:5432) - projetojpa - contato					
File Edit View Tools Help					
100 rows					
	id [PK] bigint	datanascimento date	nome character varying(255)	telefone character varying(255)	
1	10	2016-05-11	Eduardo Cunha	8888-3333	
2	11	2016-05-11	Waldir Maranhão	9999-4444	
*					

Edit Data - PostgreSQL 9.3 (localhost:5432) - projetojpa - logdatabase					
File Edit View Tools Help					
100 rows					
	id [PK] bigint	dataalteracao timestamp with time zone	entidade character varying(255)	tipooperacao character varying(255)	valoralterado character varying(255)
1	20	2016-05-11	model.Contato	Save	10
2	30	2016-05-11	model.Contato	Save	11
*					

Testando a Aplicação

```
11 public class Principal {
12
13     private static EntityManager em;
14
15     public static void main(String[] args) {
16
17         Calendar data = Calendar.getInstance();
18         GenericDAO gdao = new GenericDAO();
19
20         em = Banco.getInstance().getEntityManager();
21         Contato c1 = new Contato();
22         c1 = em.find(Contato.class, (long)10);
23         c1.setNome("Francisco Cunha");
24         gdao.alterar(c1);
25
26         c1 = em.find(Contato.class, (long)11);
27         gdao.deletar(c1);
28     }
29 }
```

Testando a Aplicação

Edit Data - PostgreSQL 9.3 (localhost:5432) - projetojpa - contato					
File Edit View Tools Help					
100 rows					
	id [PK] bigint	datanascimento date	nome character varying(255)	telefone character varying(255)	
1	10	2016-05-11	Francisco Cunha	8888-3333	
*					

Edit Data - PostgreSQL 9.3 (localhost:5432) - projetojpa - logdatabase					
File Edit View Tools Help					
100 rows					
	id [PK] bigint	dataalteracao timestamp without time zone	entidade character varying(255)	tipooperacao character varying(255)	valoralterado character varying(255)
1	20	2016-05-11 11:37:54.819	model.Contato	Save	10
2	30	2016-05-11 11:37:55.439	model.Contato	Save	11
3	40	2016-05-11 11:44:34.91	model.Contato	Update	10
4	50	2016-05-11 11:44:35.43	model.Contato	Delete	11
*					