
escuela técnica superior
de ingeniería informática
Tiempo: ?h
Hibernate
Versión original: Autor (mes 200?)
Última revisión: Autor (mes 2007): descripción cambios
Departamento de
Lenguajes y Sistemas Informáticos
Grupo de Ingeniería del Software
Febrero 2007
UNIVERSIDAD D SEVILLA
[Ángel US V7] Diseño: Amador Durán Toro (2003-2006)


[Ángel US V7] Diseño: Amador Durán Toro (2003-2006)
Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos
1. Introducción
2. POJOs
3.Mapeo de clases
4. Primitivas de Hibernate
5. Patrón DAO
6. Alternativa
7. Bibliografía
Hibernate
• Capa de Persistencia (Hibernate)
– Trabaja con POJO's
– Framework ligero → Adiós servidor de aplicaciones
– No es una especificación pero es el más usado
– ORM (Object/Relation Mapping)
– Mapeo en ficheros independientes(.hbm.xml)
– Hibernate.cfg.xml
– Clases java para trabajar con BD's
– ¿Herramienta para generar esto automáticamente?
– HibernateTools
HIBERNATE
Sevilla, abril de 2007
Grupo de Ingeniería del Software
UNIVERSIDAD D SEVILLA

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos





- POJO's
  - Representan el Modelo del dominio
  - Muchas veces, también los datos a almacenar
  - Implementación
    - Constructor sin parámetros
    - Atributos privados con sus get/set
    - Relaciones como referencias y Collections (1 y * respectivamente)
    - Enumeraciones y componentes como referencias
    - Métodos de negocio (los dejaremos fuera)
    - Serializable, HashCode y Equals



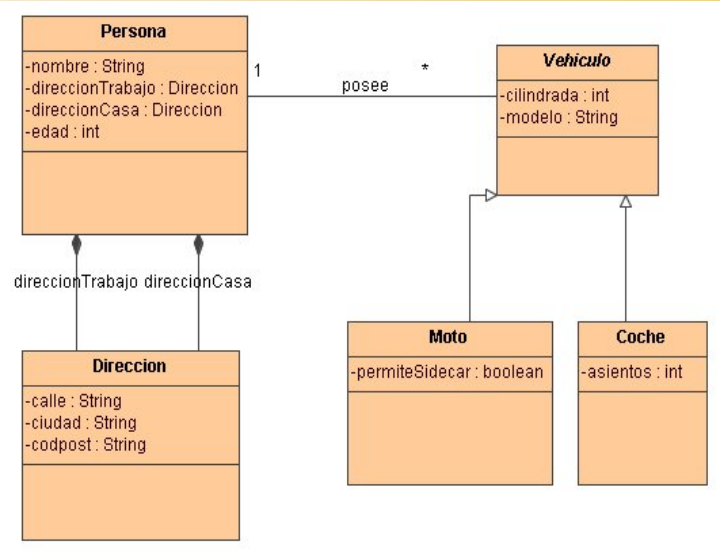


Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos





- Ejemplo



Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



```
public class Persona implements Serializable
{
  private Long id;
  private String nombre;
  private int edad;
  private Direccion direccionTrabajo;
  private Direccion direccionCasa;

  private Collection<Vehiculo> vehiculos = new HashSet<Vehiculo>();
  public Long getId(){ return this.id; }
  public void setId(Long id){ this.id = id; }
  public int getEdad(){ return this.edad; }
  public void setEdad(int edad){ this.edad = edad; }
  public String getNombre(){ return this.nombre; }
  public void setNombre(String nom){ this.nombre = nom; }
  public Direccion getDireccionTrabajo()  { return this.direccionTrabajo; }
  public void setDireccionTrabajo (Direccion d) { this.direccionTrabajo = d; }
  public Direccion getDireccionCasa() { return this.direccionCasa; }
  public void setDireccionCasa (Direccion d) { this.direccionCasa = d; }
  public Collection<Vehiculo> getVehiculos() {return this.vehiculos;}
  public void setVehiculos(Collection<Vehiculo> vehiculos){
       this.vehiculos = vehiculos;
  }
  public void addVehiculo(Vehiculo vehiculo) {
      this.vehiculos.add(vehiculo);
       /*vehiculo.setPersona(this);*/
  }
  public boolean equals(Object object){…}
  public int hashCode(){…}
}
```





[Ángel-US-V7] Diseño. Amador Durán Toro (2003-2006)

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos

1. Introducción
2. POJOs
3.Mapeo de clases
4. Primitivas de Hibernate
5. Patrón DAO
6. Alternativa
7. Bibliografía

## *Hibernate*

- Mapeo de clases

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate
Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
 <class name="com.garage.Persona" table="PERSONA" …>
     <id name="id" type="java.lang.Long" unsaved-value="null">
          <column name="ID" sql-type="BIGINT"/>
          <generator class="native">
          </generator>
     </id>
     <property name="nombre" type="java.lang.String">
       <column name="NOMBRE" not-null="true" unique="false"
sql-type="VARCHAR(255)"/>
     </property>
     <property name="edad" type="int">
       <column name="EDAD" not-null="true" unique="false" sql-
type="INTEGER"/>
      </property>
   </class>
</hibernate-mapping>
```

Sevilla, abril de 2007
Grupo de Ingeniería del Software

UNIVERSIDAD D SEVILLA

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



## Hibernate

- Mapeo de Relaciones
  - En el fichero Persona.hbm.xml :

```
<set name="vehiculos" order-by="PERSONA_FK">
  <key foreign-key="VEHICULO_PERSONA_FKC">
    <column name="PERSONA_FK" sql-type="BIGINT"/>
  </key>
  <one-to-many class="com.garage.Vehiculo"/>
</set>
```

  - En Vehiculo.hbm.xml :

```
<many-to-one name="persona"
class="com.garage.Persona"
foreign-key="VEHICULO_PERSONA_FKC">
<column name="PERSONA_FK" not-null="true" sql-
type="BIGINT"/>
</many-to-one>
```



[Arge\ IS V7] Diseño. Amador Durán Toro (2003-2006)

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos

1. Introducción
2. POJOs
3.Mapeo de clases
4. Primitivas de Hibernate
5. Patrón DAO
6. Alternativa
7. Bibliografía

## Hibernate

- Mapeo de Componentes

```
<component name="direccionCasa" class=" com.garage.
Direccion">
    <property name="calle" type="java.lang.String">
      <column name="DIRECCION_CASA_CALLE" not-
null="true" unique="false" sql-type="VARCHAR(255)"/>
    </property>
    <property name="ciudad" type="java.lang.String">
      <column name="DIRECCION_CASA_CIUDAD" not-
null="true" unique="false" sql-type="VARCHAR(255)"/>
    </property>
    <property name="codpost" type="java.lang.String">
      <column name="DIRECCION_CASA_CODPOST" not-
null="true" unique="false" sql-type="VARCHAR(255)"/>
    </property>
  </component>
```

Sevilla, abril de 2007
Grupo de Ingeniería del Software
UNIVERSIDAD Đ SEVILLA

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



## Hibernate

- Mapeo de Herencia (Una tabla por clase concreta)

```
<class name="com.garage.Moto" table="MOTO" dynamic-
insert="false" dynamic-update="false">
      <id name="id" type="java.lang.Long" unsaved-value="null">
         <column name="ID" sql-type="BIGINT"/>
         <generator class="native">
         </generator>
      </id>
      <property name="modelo" type="int">
         <column name="MODELO" not-null="true" unique="false"
sql-type="INTEGER"/>
      </property>
      <many-to-one name="persona" class="com.garage.Persona"
foreign-key="VEHICULO_PERSONA_FKC" >
         <column name="PERSONA_FK" not-null="true" sql-
type="BIGINT"/>
      </many-to-one>
      <property name="permiteSidecar" type="boolean">
         <column name="PERMITE_SIDECAR" not-null="true"
unique="false" sql-type="VARCHAR(255)"/>
      </property>
</class>
```



[Angel-US-V7] Diseño. Amador Durán Toro (2003-2006)

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



## Hibernate

- Mapeo de Herencia (Una tabla por jerarquía de clases)

```
<class name="com.garage.Vehiculo" table="VEHICULO" >
      <id name="id" type="java.lang.Long" unsaved-value="null">
         <column name="ID" sql-type="BIGINT"/>
         <generator class="native">
         </generator>
      </id>
      <discriminator column="class" type="string"/>
      <property name="modelo" type="int">
         <column name="MODELO" not-null="true" unique="false"
sql-type="INTEGER"/>
      </property>
      ...
      <subclass name="com.garage.Moto" discriminator-
value="Moto" abstract="false">
        <property name="permiteSidecar" type="boolean">
           <column name="PERMITE_SIDECAR" not-null="true"
           unique="false" sql-type="VARCHAR(255)"/>
        </property>
      </subclass>
</class>
```

Sevilla, abril de 2007
Grupo de Ingeniería del Software
UNIVERSIDAD D SEVILLA

- Mapeo de Herencia (Una tabla por subclase)

```
<class name="Vehiculo" table="VEHICULO">
      <id name="id" type="java.lang.Long" unsaved-value="null">
         <column name="ID" sql-type="BIGINT"/>
         <generator class="native">
         </generator>
      </id>
      <property name="modelo" type="int">…</property>
         …
     <joined-subclass name="com.garage.Moto" table="MOTO"
abstract="false">
         <key foreign-key="MOTOIFKC">
            <column name="ID" sql-type="BIGINT"/>
         </key>
        <property name="permiteSidecar" type="boolean">
            <column name="PERMITE_SIDECAR" not-null="true"
            unique="false" sql-type="VARCHAR(255)"/>
         </property>
      </joined-subclass>
         …
    </class>
</hibernate-mapping>
```







- Hibernate.cfg.xml

```
<?xml version="1.0" …>
<hibernate-configuration>
   <session-factory>
      <!-- properties -->
      <property name="connection.driver_class">
         com.mysql.jdbc.Driver
      </property>
      <property name="connection.username">root</property>
      <property name="connection.password">admin</property>
      <property name="connection.url">
         jdbc:mysql://localhost/basededatos
      </property>
      <property name="show_sql">true</property>
      <property name="dialect">
         org.hibernate.dialect.MySQLDialect
      </property>
      <!-- mapping files -->
      <mapping resource="com/garage/Persona.hbm.xml"/>
      <mapping resource="com/garage/Vehiculo.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



- Manejo de Sesiones y transacciones

```
SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
Session ses = sessionFactory.openSession();
Transaction tx = ses.beginTransaction();
tx.commit();
tx.rollback();
ses.close();
```

- HibernateException
- Funciones de hibernate (Create, Delete y Update)

```
Persona p = new Persona();
session.Load(p,pkId); // o Get()
Session.save(p);
Session.saveOrUpdate(p);
session.Delete(p);
p.setNombre("Juan");
session.update(p);
```





Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



- Funciones de Hibernate (Read)

```
List personas = session.createCriteria(Persona.class)
    .Add(Restrictions.like("nombre", "Ramon%") )
    // el nombre debe ser Ramon y lo que siga
    .Add(Restrictions.between("edad", minAge, maxAge) )
    //la edad está entre min y max
    .Add(Restrictions.or(       //o la edad es cero o null
        Restrictions.eq( "edad", new Integer(0) ),
        Restrictions.isNull("edad")
    ))
    .Add(Restrictions.in( "nombre", new String[] { "Ramon",
"Paco", "Fran" } ) )
    // el nombre solo puede ser Ramón, Paco o Fran
    .Add(Restrictions.disjunction()
    .Add(Restrictions.isNull("edad") )
    .Add(Restrictions.eq("edad", new Integer(0) ) )
    .Add(Restrictions.eq("edad", new Integer(1) ) )
    .Add(Restrictions.eq("edad", new Integer(2) ) )
    // la edad solo puede ser null, 0, 1 o 2
    ))
    .list();
```

Escuela Técnica Superior de Ingeniería Informática
Departamento de Lenguajes y Sistemas Informáticos



- Patron DAO (java 5)
  - Recomendado por hibernate

```
public interface GenericDAO<T, ID extends Serializable>
{
   T findById(ID id, boolean lock);
   List<T> findAll();
   List<T> findByExample(T exampleInstance, String...
excludeProps);
   T makePersistent(T entity);
   void makeTransient(T entity);
}
```

  - Extender para cada pojo

```
public interface PersonaDAO extends GenericDAO<Persona,
Long>
{}
```









Escuela Técnica Superior de Ingeniería Informática
Departamento de Lenguajes y Sistemas Informáticos



- Forma de trabajo

```
Persona p = new Persona();
// Atributo Normal
p.setNombre("nombre");
// Componentes
Direccion d = new Direccion("ciuCas"," calCas"," codCas");
//constructor con parametros
p.setDireccionCasa(d);
d = new Direccion("calDirTra"," ciuDirTra"," codDirTra");
p.setDireccionTrabajo(d);
// Enumeration - constructor desde string
Sexo sx = Sexo.fromString("HOMBRE");
p.setSexo(sx);
//Guardo
DAOFactory dfact;
dfact = DAOFactory.getInstance(); ← Implementar
PersonaDAO pdao = dfact.getPersonaDAO();
HibernateUtils.beginTransaction(); ← Implementar
pdao.makePersistent(p);
HibernateUtils.commitTransaction(); ← Implementar
```

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos



- La alternativa – JPA
  - EJB3 → Especificación de SUN
  - Intervinieron gente de Hibernate y Oracle
  - Trabaja con POJOs
  - Xml mapping → Anotaciones
  - HQL → EJB QL
  - Session y transaction → Por contenedor (no Home)
  - Anoaciones opcionales










Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos

1. Introducción
2. POJOs
3.Mapeo de clases
4. Primitivas de Hibernate
5. Patrón DAO
6. Alternativa
7. Bibliografía

- Ejemplo

```
@Entity
public class Persona {
  private Long id;
  private String nombre;

  private Collection<Vehiculo> vehiculos = new HashSet<Vehiculo>();

  @Id(generate=AUTO)
  public Long getId(){ return this.id; }
  public void setId(Long id){ this.id = id; }

  @Column(length=500)
  public String getNombre(){ return this.nombre; }
  public void setNombre(String nom){ this.nombre = nom; }

  @OneToMany(cascade=ALL)
  public Collection<Vehiculo> getVehiculos() {return this.vehiculos;}
  public void setVehiculos(Collection<Vehiculo> vehiculos){
       this.vehiculos = vehiculos;
  }
  public void addVehiculo(Vehiculo vehiculo) {
      this.vehiculos.add(vehiculo);
       /*vehiculo.setPersona(this);*/
  }
}
```

Sevilla, abril de 2007
Grupo de Ingeniería del Software

UNIVERSIDAD Đ SEVILLA

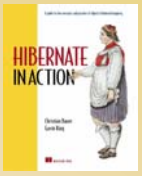
[Ángel IJS V7] Diseño: Amador Durán Toro (2003-2006)

Escuela Técnica Superior
de Ingeniería Informática
Departamento de Lenguajes
y Sistemas Informáticos

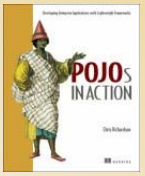




- Bibliografía
  - **Hibernate in Action**
    **Manning**




  - **POJOs in Action**
    **Manning**




  - **EJB3 in Action**
    **Manning**