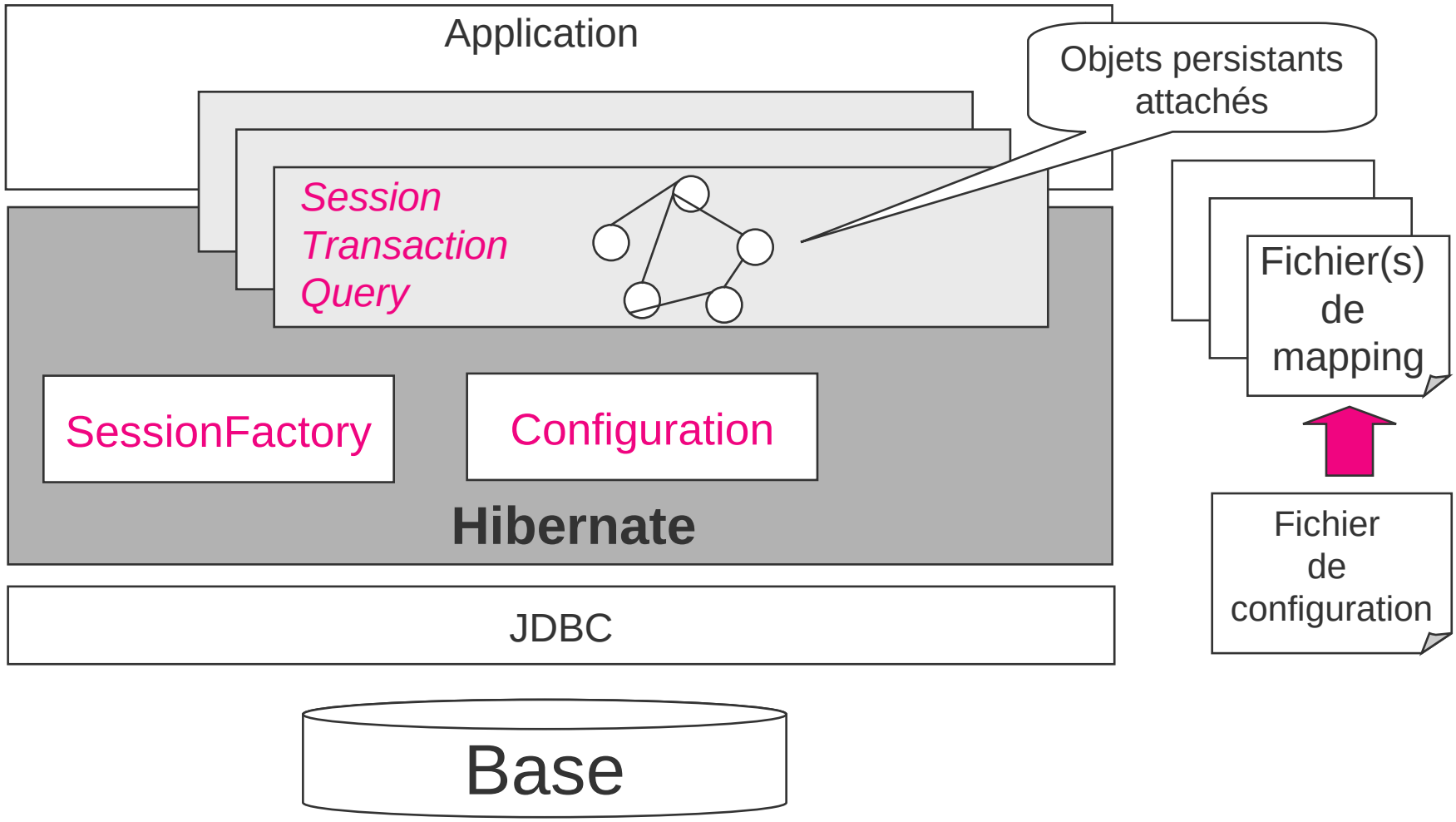
Démarrez avec Hibernate

Hibernate : un cache de la BD

API coeur

Exemple

Architecture Hibernate
Présentation



Cache de la base de données

- La session Hibernate peut être vue comme un cache de la base de données
- Tout au long de la durée de vie de la session, des objets entités sont chargés dans le cache. On dit qu'il sont **attachés** à la session
- Les mises à jour (création, modification, suppression) sont effectués sur les objets en mémoire
- A certains moments, le cache est synchronisé avec la base de données et Hibernate génère les ordres SQL nécessaires
- Le comportement par défaut est de synchroniser le cache uniquement à la fermeture de la session

Identité d’objet

- Identité objet (Opérateur ==)
  - Même espace mémoire dans la JVM
- Égalité d’objets (Méthodes equals() et hashcode())
  - Même valeur mais pas forcément même espace mémoire
- Identité relationnelle (base de données)
  - Sont représentés dans la base par la même ligne et ont la même clé primaire. obj1.getId() == obj2.getId()

=> Hibernate garantit qu'il n'y a jamais 2 instances de la même entité dans une session

=> En général, les méthodes *equals()* et *hashcode()* s'appuie sur l'identité relationnelle

Cycle de vie des objets métier
Définitions

- **Transient**
  - ne possède pas une image de son état stockée en BD.
  - quand il est récupéré (garbage), ses données sont perdues
- **Persistent / Attaché**
  - possède une image de son état stockée en BD (un ID)
  - durée de vie potentiellement infinie.
  - Attaché à une Session qui assure la synchronisation état dans JVM vs état en BD
- **Détaché**
  - Possède une image de son état stockée en BD (un ID)
  - N'est pas attaché à une session (pas de synchronisation)

Démarrez avec Hibernate

Hibernate : un cache de la BD

API coeur

Exemple

Architecture Hibernate
Interfaces Configuration et SessionFactory

- *Configuration*
  - Objet à utiliser à l'initialisation seulement
  - *configure*() à partir du fichier « *hibernate.cfg.xml* »
  - ou plus rarement programmatiquement
- *SessionFactory*
  - *openSession*() : retourne une session connectée à la base en fonction des paramètres fournis en configuration.

Exemple
Initialiser Hibernate

```
package com.tsystems.etechno.j12.exemple.hibernate;

import org.hibernate.HibernateException;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class DBHelper {
  private static Configuration cfg = null;
  private static SessionFactory factory = null;

  static {
      try {
        cfg = new Configuration();
        factory = cfg.configure().buildSessionFactory();
      } catch (HibernateException e) {
        System.err.println("problème d'initialisation d'Hibernate");
        throw e;
      }
  }

    public static SessionFactory getFactory(){return factory;}
}
```

Exemple
hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
     <property name="connection.driver_class">
       org.gjt.mm.mysql.Driver</property>
     <property name="hibernate.connection.url">
       jdbc:mysql://172.16.102.42/Mediatheque</property>
     <property name="hibernate.connection.username">
       hve</property>
     <property name="hibernate.connection.password">
       pwd</property>
     <property name="show_sql">
       true</property>

    <mapping resource="org/formation/exemple/metier/Theme.hbm.xml"/>
    <mapping resource="org/formation/exemple/metier/MotClef.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

Propriétés de la connection et du moteur Hibernate

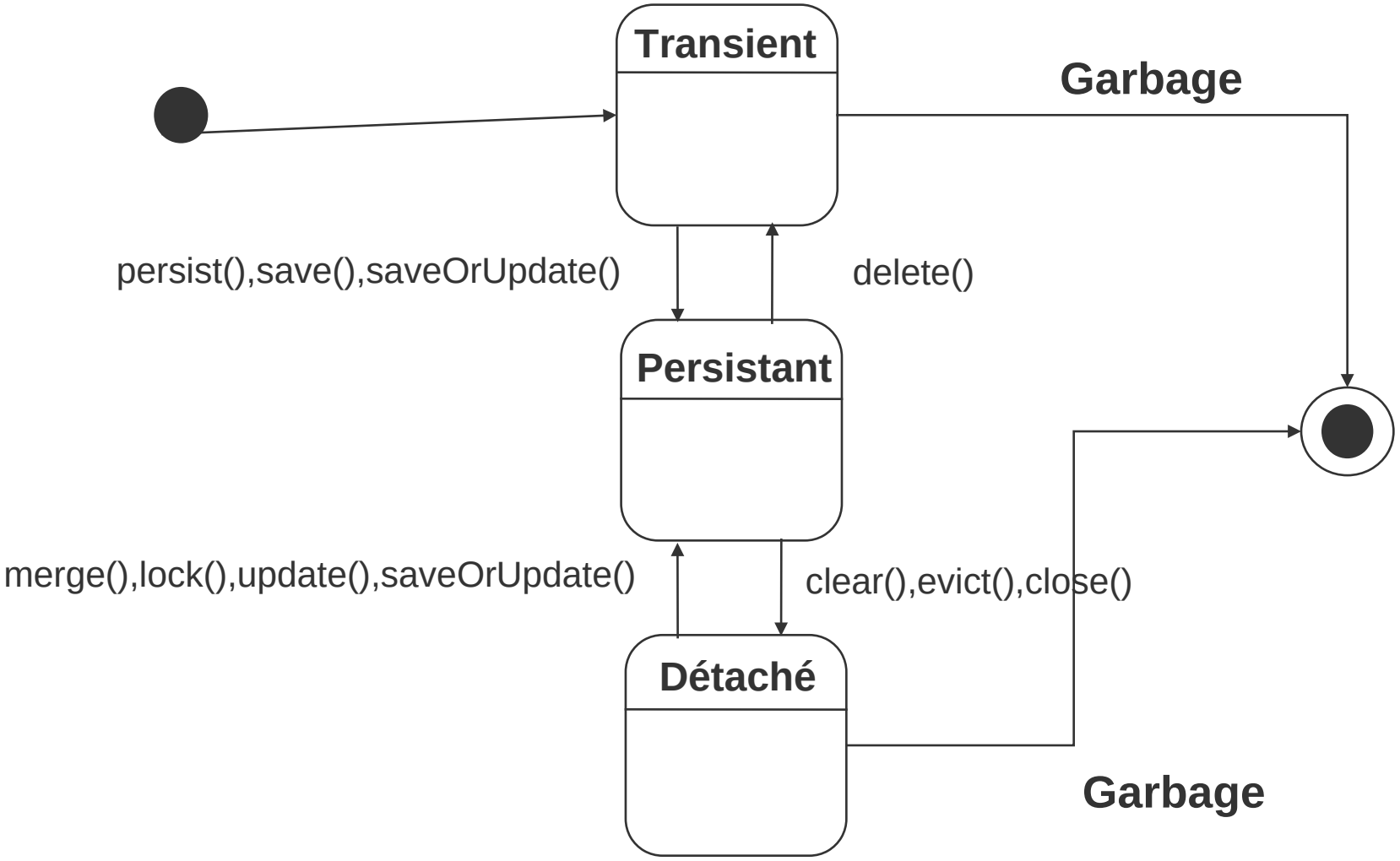
Propriétés du moteur Hibernate pour activer la trace des requêtes SQL

Références sur les fichiers de mapping

Architecture Hibernate
Interface Session

- Interface principale du service de persistance Hibernate.
- Implémentation non Thread-safe
- Méthodes
  - Rendre un objet persistant (persist(), save() => INSERT)
  - Lire un objet (get() / load() => SELECT)
  - Mettre à jour (merge(), update() => UPDATE)
  - beginTransaction()
  - close()
  - createQuery()

Cycle de vie et méthodes de la session
Diagramme d'états



Architecture Hibernate
Interface Query et Transaction

- Query
  - Représentation d'une requête d'objets
  - Liée à la session qui l'a créée
    - *maSession.createQuery(« from Theme »)*
  - Permet les paramètres
  - *list()* , *uniqueResult()*
- Transaction
  - Représente une unité de travail
  - Liée à une session
  - Pour une session une seule transaction à la fois.
  - *commit()* / *rollback()*

Démarrez avec Hibernate

Hibernate : un cache de la BD

API cœur

Exemple

Exemple
Requête en lecture des thèmes

```
public List<Theme> getAllThemes(){
    List<Theme> ret = null;
    Session session = DBHelper.getFactory().openSession();
    Transaction tx = null;
    try {
       tx = session.beginTransaction();
       Query hqlQuery = session.createQuery("from Theme");
       ret = (List<Theme>)hqlQuery.list();
       tx.commit();
    }
                // Gestion des exceptions
        return ret;
    }
```

1 2 3 4 5

Trace Hibernate

```
Hibernate: select theme0_.id as id, theme0_.label as label0_ from TTheme theme0_
Hibernate: select motclefs0_.IDTheme as IDTheme1_, motclefs0_.ID as ID1_, motclefs0_.ID as
    ID0_, motclefs0_.mot_clef as mot2_1_0_ from tmotclef motclefs0_ where motclefs0_.IDTheme=?
...
```

Exemple
Requête en lecture d'un thème

```
public Theme getThemeComplet(String label){
    Session session = DBHelper.getFactory().openSession();      1
    Transaction tx = null;
    Theme ret = null;
    try {
    tx = session.beginTransaction();
    Query hqlQuery =                                                    2
           session.createQuery("from Theme t where t.label like :labelSearch ");
    hqlQuery.setParameter("labelSearch", "%"+label+"%");
    ret = (Theme)hqlQuery.uniqueResult();
    tx.commit();
    }…… // gestion des exception
        return ret;
}
```

Trace Hibernate

```
Hibernate: select theme0_.id as id, theme0_.label as label0_ from TTheme theme0_ where theme0_.label like ?
Hibernate: select motclefs0_.IDTheme as IDTheme1_, motclefs0_.ID as ID1_, motclefs0_.ID as ID0_,
        motclefs0_.mot_clef as mot2_1_0_ from tmotclef motclefs0_ where motclefs0_.IDTheme=?
```

Exemple
Requête en écriture

```
public void createTheme(Theme t) {
        Session session = DBHelper.getFactory().openSession();          1
        Transaction tx = null;
        Theme ret = null;
        try {
            tx = session.beginTransaction();          2

            session.persist(t);          3

            tx.commit();          4
        … // gestion des exceptions
}
```

Trace Hibernate

```
Hibernate: insert into TTheme (label) values (?)
Hibernate: insert into tmotclef (mot_clef) values (?)
Hibernate: insert into tmotclef (mot_clef) values (?)
Hibernate: insert into tmotclef (mot_clef) values (?)
Hibernate: update tmotclef set IDTheme=? where ID=?
Hibernate: update tmotclef set IDTheme=? where ID=?
Hibernate: update tmotclef set IDTheme=? where ID=?
```

Exemple
Requête de suppression

```
public void deleteTheme(Theme t) {
        Session session = DBHelper.getFactory().openSession();
        Transaction tx = null;
        Theme ret = null;
        try {
           tx = session.beginTransaction();
           session.delete(t);                    (1)
           tx.commit();
        }
        … // gestion des exceptions
    }
```

Trace Hibernate

```
Hibernate: select theme0_.id as id, theme0_.label as label0_ from TTheme theme0_ where theme0_.id = ?
Hibernate: select motclefs0_.IDTheme as IDTheme1_, motclefs0_.ID as ID1_, motclefs0_.ID as ID0_,
      motclefs0_.mot_clef as mot2_1_0_ from tmotclef motclefs0_ where motclefs0_.IDTheme=?
Hibernate: update tmotclef set IDTheme=null where IDTheme=?
Hibernate: delete from tmotclef where ID=?
Hibernate: delete from tmotclef where ID=?
Hibernate: delete from tmotclef where ID=?
Hibernate: delete from TTheme where id=?
```

Exemple
Requête de mise à jour

```
public void updateTheme(Theme t) {
    Session session = DBHelper.getFactory().openSession();
    Transaction tx = null;
    Theme ret = null;
    try {
        tx = session.beginTransaction();
        session.merge(t);                          (1)
       tx.commit();
    }
        …// gestion des exceptions

    }
```

Trace Hibernate

```
Hibernate: select theme0_.id as id0_, theme0_.label as label0_0_ from TTheme theme0_ where theme0_.id=?
Hibernate: select motclefs0_.IDTheme as IDTheme1_, motclefs0_.ID as ID1_, motclefs0_.ID as ID0_,
      motclefs0_.mot_clef as mot2_1_0_ from tmotclef motclefs0_ where motclefs0_.IDTheme=?
Hibernate: update TTheme set label=? where id=?
```

Exemple
Pattern d'usage Hibernate

```
public void doXXX(Param t) {
    Session session = DBHelper.getFactory().openSession();
    Transaction tx = null;
    Theme ret = null;
    try {
       tx = session.beginTransaction();
        // travail à effectuer
        tx.commit();
                  }
      catch(Exception e){
        if (tx != null){
            try { tx.rollback(); }
                catch(HibernateException he){ he.printStackTrace();}
        }
        e.printStackTrace();
    }
    finally{
            try { session.close(); }
         catch(HibernateException he){  he.printStackTrace(); }
        }
    }
```

Exemple
Fichier de mapping : Theme.hbm.xml (1/2)

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping SYSTEM
"C:\users\hve\Veille_technologique\cours_hibernate\
            hibernate-3.0\hibernate-mapping-3.0.dtd">

<hibernate-mapping  package="com.plb.etechno.j12.exemple.metier">
    <class name="Theme" table="TTheme">
           <id  name="id" column="id">
                <generator class="native"/>
           </id>

           <property name="label" column="label"/>

           <set name="motclefs"  lazy="false">
               <key column="IDTheme"/>
               <one-to-many class="MotClef"/>
           </set>
    </class>
</hibernate-mapping>
```

Exemple
Fichier de mapping : Theme.hbm.xml (2/2)

- DTD permettant de valider le fichier de mapping (obligatoire)
- Déclaration de la racine XML de mapping et attribut indiquant à quel package se réfèrent les classes mappées.
- Mapping entre la classe et la table
- Mapping de l'id (obligatoire)
- Mapping d'une propriété de la classe
- Mapping d'une relation 1 -> N

Exemple
Modifications sur la classe Theme

```
public class Theme {
    private Long id;
    private String label;
    private Set<MotClef> motclefs = new HashSet<MotClef>();

    public Set<MotClef> getMotclefs() {
        return motclefs;
    }
    public void setMotclefs(Set<MotClef> motclefs) {
        this.motclefs = motclefs;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getLabel() {
        return label;
    }
    public void setLabel(String label) {
        this.label = label;
    } … etc
```

1

2

3