Programmation Web Avancée
Hibernate

Thierry Hamon

Bureau H202 - Institut Galilée
Tél. : 33 1.48.38.35.53
Bureau 150 – LIM&BIO – EA 3969
Université Paris 13 - UFR Léonard de Vinci
74, rue Marcel Cachin, F-93017 Bobigny cedex
Tél. : 33 1.48.38.73.07, Fax. : 33 1.48.38.73.55
thierry.hamon@univ-paris13.fr
http://www-limbio.smbh.univ-paris13.fr/membres/hamon/PWA-20122013

Introduction

Hibernate :

- Framework Java
- Gestion de persistance des objets dans les BD relationnelles
- Solution pour faire le lien entre la programmation objet et les SGBD :
  - les accès à la base de données sont gérés par des objets et des méthodes Java
  - les JavaBean masquent la structure et les spécificités de la BD
- Hibernate (3) : Représentation XML du résultat de la requête

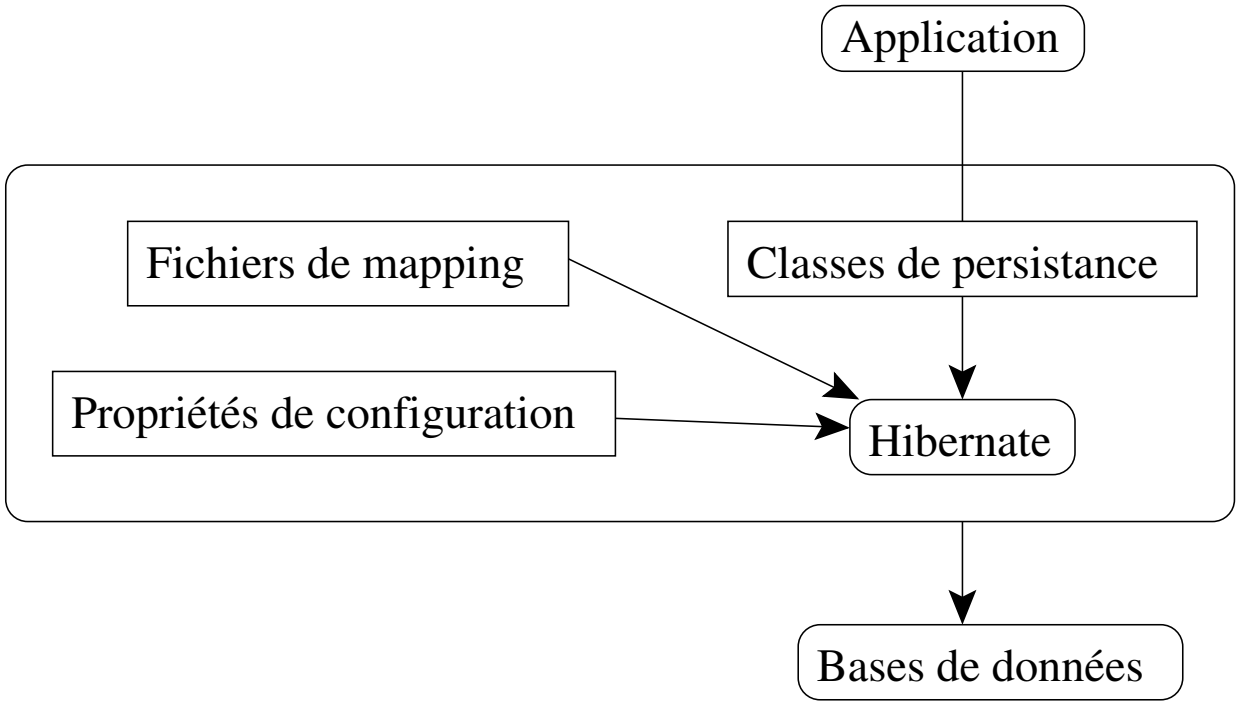
Autre représentation des objets : Plain Old Java Object (POJO)

Architecture

Eléments nécessaires :

- Classe de persistance : une classe de JavaBean pour l'encapsulation des données d'une table
- Correspondance entre la classe et la table : fichier de *mapping*
- Propriétés de configuration : Informations permettant la connexion à la BD

Architecture



Persistance

- Non-persistance : Perte des objets manipulés par une application lors la fin d'une session
- Persistance : Sauvegarde des objets manipulés par une application pour pouvoir les recréer lors d'une autre session
- Moyen : utilisation d'une BD

Rendre une application persistante :

- insertion et mise à jour de données dans une BD
- à partir de données XML de manière similaire à la manipulation d'objets Java

Gestion de la persistance

Plusieurs aspects :

- Connexion à la BD
- Création de tous les objets ou uniquement de ceux nécessaires
- Transformation (*mapping*) de la structure de la BD pour l'adapter aux objets
- Chargement en mémoire des données en une seule fois ou au fur et à mesure

Hibernate : Framework en charge de la persistance des données dans une application Web

Mise en œuvre simple

- Exemple tiré du tutoriel `http://docs.jboss.org/hibernate/orm/3.5/reference/en/html/tutorial.html` (VF : `http://www.dil.univ-mrs.fr/~massat/docs/hibernate-3.1/reference/fr/html/tutorial.html`)
- Persistance grâce à la BD interne d'Hibernate
- Définition
  - Configuration (propriétés et mapping)
  - Classe de persistance
  - Gestion de la session / interface Hibernate

Configuration

- Fichier de mapping (nom du fichier : nom de la classe, extension `.hbm.xml`)
- Configuration de Hibernate (`hibernate.cfg.xml`)

Ingénieurs sup Galilée

Fichier de mapping

- Chargement et stockage des données persistantes
- Indication des tables et de la BD à utiliser
- Définition de la structure des tables
- Définition des clés primaires
- Correspondance entre les noms et les types des objets et les noms des colonnes

Exemple de fichier de mapping

fichier `Event.hbm.xml`

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="events.Event" table="EVENTS">
        <id name="id" column="EVENT_ID">
            <generator class="native"/>
        </id>
        <property name="date" type="timestamp" column="EVENT_DATE"/>
        <property name="title"/>

        <set name="participants" table="PERSON_EVENT" inverse="true">
            <key column="EVENT_ID"/>
            <many-to-many column="PERSON_ID" class="events.Person"/>
        </set>
    </class>
</hibernate-mapping>
```

Explications

- Balise `hibernate-mapping` : définition des mappings
- Balise `class` : définition de la classe et de la table concernées par le mapping
  - classe : `events.Event`
  - table : `EVENTS`

Explications

- Balise `id` : déclaration de la propriété de l'identifiant
  - `name="id"` : identifiant Java conduisant Hibernate à utiliser les *getter/setter* standard (*getId/setId*)
  - `generator` : stratégie de génération de l'identifiant
    - `native` : choix entre les valeurs `identity` (colonne du SGBD), `sequence` (séquence fournie par le SGBD) ou `hilo` (génération spécifique à Hibernate)
    - `increment` : identifiants entiers (à utiliser lorsque qu'aucun autre processus accède à la table
    - Possibilité d'écrire sa propre stratégie de génération d'identifiant

Explications

- Propriété `title` :
  - le nom de la propriété est le nom de la colonne
  - détermination du type de mapping et de la conversion par Hibernate
- Propriété `date` :
  - attribut `column` : correspondance avec la colonne `EVENT_DATE` (`date` pouvant être réservé)
  - attribut `type` : spécification du type de mapping et de la conversion
- Balise `set` : définition d'une association (table `PERSON_EVENT`)
  NB : le mapping pour la table `PERSON` est à écrire

Ingénieurs sup Galilée

Configuration de Hibernate

- Fichier `hibernate.properties` ou `hibernate.cfg.xml` (prioritaire)
- Balise `hibernate-configuration` : configuration de hibernate pour l'application courante
- Balise `session-factory` : Fabrique de session
- Définition des paramètres de connexion
  - Pilote du SGBD (propriété `connection.driver_class`)
    Autres valeurs possibles :
    - `com.mysql.jdbc.Driver` (MySQL)
    - `org.postgresql.Driver` (Postgresql)
    - `oracle.jdbc.driver.OracleDriver` (Oracle)

Configuration de Hibernate

- Définition des paramètres de connexion (suite)
  - URL du serveur (propriété `connection.url`)
    Autres valeurs possibles :
    - `jdbc:mysql://localhost/hibernateAppli` (MySQL)
    - `jdbc:postgresql://localhost/hibernateAppli` (Psotgresql)
    - `jdbc:oracle:thin:@localhost:1521:oracle` (Oracle - à vérifier)
  - Nom d'utilisateur et mot de passe (proprétés `connection.username` et `connection.password`)
  - Nombre de connexion simultanée (propriété `connection.pool_size`)

Configuration de Hibernate

- Définition des paramètres de connexion (suite)
  - Dialecte SQL à utiliser (propriété `dialect`)
    Autres valeurs :
    - `org.hibernate.dialect.MySQLDialect` (MySQL)
    - `org.hibernate.dialect.PostgreSQLDialect` (Postgresql)
    - `org.hibernate.dialect.OracleDialect`, `org.hibernate.dialect.Oracle10gDialect` (Oracle)
- Fichiers de mapping : balise `mapping`
- Propriété `hbm2ddl.auto`
  - comportement au démarrage du serveur de BD Hibernate (validation, mise à jour de la BD existante, suppression et création de la BD)
  - valeurs possibles : `validate`, `update`, `create`, `create-drop`)
  - cette propriété est à supprimer pour garder les informations entre deux arrêt du serveur BD Hibernate

Exemple de fichier de configuration Hibernate

hibernate.cfg.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="connection.driver_class">org.hsqldb.jdbcDriver</property>
        <property name="connection.url">jdbc:hsqldb:hsql://localhost</property>
        <property name="connection.username">sa</property>
        <property name="connection.password"></property>

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.HSQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache  -->
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider
        </property>
```

Exemple de fichier de configuration Hibernate

hibernate.cfg.xml

```
        <!-- Echo all executed SQL to stdout -->
          <property name="show_sql">true</property>

          <!-- Drop and re-create the database schema on startup -->
          <property name="hbm2ddl.auto">create</property>

          <mapping resource="events/Event.hbm.xml" />
          <mapping resource="events/Person.hbm.xml" />
      </session-factory>
  </hibernate-configuration>
```

Classe de persistance

- Classe JavaBean
- Utilisation de convention de nommage JavaBean pour les méthodes `getter` et `setter`
- Visibilité des champs : privée
- Visibilité de la classe : `public` recommandée
- Contraintes :
  - Propriété obligatoire : `id`
  - Constructeur sans argument obligatoire

Exemple de classe de persistance

```
package events;

import java.util.*;

public class Event {
    private Long id;

    private String title;
    private Date date;

    public Event() {}
    public Long getId() {
        return id;
    }
    private void setId(Long id) {
        this.id = id;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
```

Ingénieurs sup Galilée

Exemple de classe de persistance

```
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    private Set participants = new HashSet();
    public Set getParticipants() {
        return participants;
    }
    public void setParticipants(Set participants) {
        this.participants = participants;
    }
}
```

Gestion de la session

- Fichier `HibernateUtil.java`
- Démarrage de Hibernate
  - Création de la fabrique de session `SessionFactory`
  - puis création de la session (un *thread*)

Ingénieurs sup Galilée

Fichier HibernateUtil.java

```
package util;

import org.hibernate.*;
import org.hibernate.cfg.*;

public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Mise en œuvre et déploiement

- Ecriture d'une servlet
- Utilisation de Ant pour générer l'archive war
- Tests supplémentaires

Exemple de servlet

Eléments :

- Initiation de la session et création de la transaction dans la méthode `doGet`
  - Récupération de la fabrique de session `getSessionFactory()`
  - Récupération de la session courante dans la fabrique `getCurrentSession()`
  - Démarrage de la transaction `beginTransaction()`
- Méthodes d'accès (ajout d'un enregistrement, récupération des enregistrements)
  - Sauvegarde d'un enregistrement : méthode `save(objet)`
    Etape préliminaire : création de l'objet correspondant à l'enregistrement
  - récupération des enregistrements : méthode `list()`

Exemple de Servlet

Initialisation de la transaction

```
package events;

import util.HibernateUtil;
import javax.servlet.http.*;
import javax.servlet.ServletException;
import java.io.*;
import java.util.*;
import java.text.SimpleDateFormat;

public class EventManagerServlet extends HttpServlet {
...
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            // Begin unit of work
            HibernateUtil.getSessionFactory().getCurrentSession().beginTransaction();

            // Write HTML header
            PrintWriter out = response.getWriter();
            out.println("<html><head><title>Event Manager</title></head><body>");
...
                String eventTitle = request.getParameter("eventTitle");
                String eventDate = request.getParameter("eventDate");
...
                    createAndStoreEvent(eventTitle, dateFormatter.parse(eventDate));
...
            listEvents(out);
```

Exemple de Servlet

Sauvegarde

```
protected void createAndStoreEvent(String title, Date theDate) {
    Event theEvent = new Event();
    theEvent.setTitle(title);
    theEvent.setDate(theDate);

    HibernateUtil.getSessionFactory()
                 .getCurrentSession().save(theEvent);
}
```

Ingénieurs sup Galilée

Exemple de Servlet

Interrogation

```
    private void listEvents(PrintWriter out) {
        List result = HibernateUtil.getSessionFactory()
                        .getCurrentSession().createCriteria(Event.class).list();
        if (result.size() > 0) {
...
            for (Iterator it = result.iterator(); it.hasNext();) {
                Event event = (Event) it.next();
...
            }
...
        }
    }
```

Ingénieurs sup Galilée

Ant

- Outil de construction de projet similaire à `make` mais en Java
- Génération de classes ou d'archives à partir d'un fichier de construction `build.xml`
- Définition de cibles (balise `target`, attribut `name`) et de dépendances (attribut `depends`)
- Réalisation d'opération sur une architecture définie dans le fichier de construction

Ingénieurs sup Galilée

Fichier build.xml

```
<project name="hibernate-tutorial" default="compile">

    <property name="sourcedir" value="${basedir}/src" />
    <property name="targetdir" value="${basedir}/bin" />
    <property name="librarydir" value="${basedir}/lib" />

    <path id="libraries">
        <fileset dir="${librarydir}">
            <include name="*.jar" />
        </fileset>
    </path>

    <target name="clean">
        <delete dir="${targetdir}" />
        <mkdir dir="${targetdir}" />
    </target>

    <target name="compile" depends="clean, copy-resources">
      <javac srcdir="${sourcedir}"
             destdir="${targetdir}"
             classpathref="libraries" />
    </target>
```

Ingénieurs sup Galilée

Fichier build.xml

```
    <target name="copy-resources">
        <copy todir="${targetdir}">
            <fileset dir="${sourcedir}">
                <exclude name="**/*.java" />
            </fileset>
        </copy>
    </target>

    <target name="run" depends="compile">
        <java fork="true" classname="events.EventManager" classpathref="libraries">
            <classpath path="${targetdir}" />
            <arg value="${action}" />
        </java>
    </target>

    <target name="war" depends="compile">
        <war destfile="hibernate-tutorial.war" webxml="web.xml">
            <lib dir="${librarydir}">
              <exclude name="jsdk*.jar" />
            </lib>

            <classes dir="${targetdir}" />
        </war>
    </target>
</project>
```

Ant - Utilisation

- commande : `ant`
- sans cible : cible par défaut

```
<project name="hibernate-tutorial" default="compile">
```

- avec une cible :
  - `ant war` : génération de l'archive WAR (pour le déploiement sur tomcat – nécessite un fichier `web.xml`)
  - `ant run -Daction=list` : exécution de la classe `events.EventManager` avec le paramètre `action=list`

Architecture de l'application Web

Répertoire racine : `hibernateTutorial`
Sous-répertoires :

- `lib` : contient les .jar nécessaires à Hibernate
- `ls` :

```
antlr.jar      commons-collections3.jar hibernate3.jar
asm3-all.jar   commons-collections.jar hibernate-commons-annotations.jar
asm3.jar       commons-logging.jar hibernate-core.jar
cglib.jar      dom4j-1.6.1.jar hibernate-ehcache.jar
hibernate-entitymanager.jar javassist.jar  slf4j-api.jar
hibernate-jdbc4-testing.jar jta.jar        slf4j-log4j12.jar
hibernate-jpa-2.0-api.jar log4j-1.2.jar
hsqldb.jar readme.txt
```

- `src` : sources de l'application

Architecture de l'application Web

- `src` : sources de l'application
  - Fichier `hibernate.cfg.xml`
  - Fichier `log4j.properties` (gestion de logs)
  - Répertoire `util` : chemin vers le code Java `HibernateUtil.java`
  - Répertoire `events` : chemin vers le code Java de l'application et les fichiers de mapping
  - `ls` :

```
Event.hbm.xml   Event.java  EventManager.java  EventManagerServlet.java
Person.hbm.xml Person.java
```

Procédure de test de l'exemple

on se situe dans le répertoire `hibernateTutorial`

- Compilation : `ant`
- Génération de l'archive : `ant war`
- Lancement du serveur BD Hibernate :
  `java -classpath ../lib/hsqldb.jar org.hsqldb.Server`
  NB : Utiliser le script `runCleanDatabase.sh` (présent dans l'archive)

Procédure de test de l'exemple

- Déploiement sur le serveur Tomcat
  URL d'accès :
  http ://marseille2 :8080/hibernate-tutorial/eventmanager
  NB : modifier `eventmanager` en incluant votre nom (voir le fichier `web.xml`

(un extrait des logs du serveur tomcat est présent dans le fichier `log.txt`)