# Document d'analyse

*Architecture d'une application Java Hibernate*

| | |
|---|---|
| **Date création du document** | 31/10/2007 |
| **Date de dernière mise à jour** | 22/09/2010 |
| **Version** | V1.1 |

# Sommaire

# Présentation du document.

## Le contexte.

En 2007, l'unité commence à développer des applications de type batch en Java.

Les problématiques étant différentes des applications de type web (gestion des versions, mises en production, ..), il convient dès lors de définir une architecture spécifique pour les applications de type batch.

## Objet – champ d'application.

L'objet de ce document est :

- d'apporter une aide technique sur l'integration d'Hibernate sur un environnement de type Eclipse.

- d'apporter une aide théorique sur les aspects de mapping O/R mis en œuvre dans Hibernate.

## Documents de référence.

Integration d'Hibernate dans Eclipse : http://www.hibernate.org/hib_docs/tools/reference/en/html/

Aspets théoriques sur Hibernate : http://www.hibernate.org/hib_docs/v3/reference/fr/html/

## Définitions et abréviations.

- **Mapping O/R ou ORM** : Mapping Objet Relationnel, technique mettant en oeuvre une correspondance entre une base de données relationnelle et des objets créés par un langage de programmation

# Organisation de l'environnement de travail avec Hibernate

## Elements de l'environnement de travail

### Ce qu'il faut

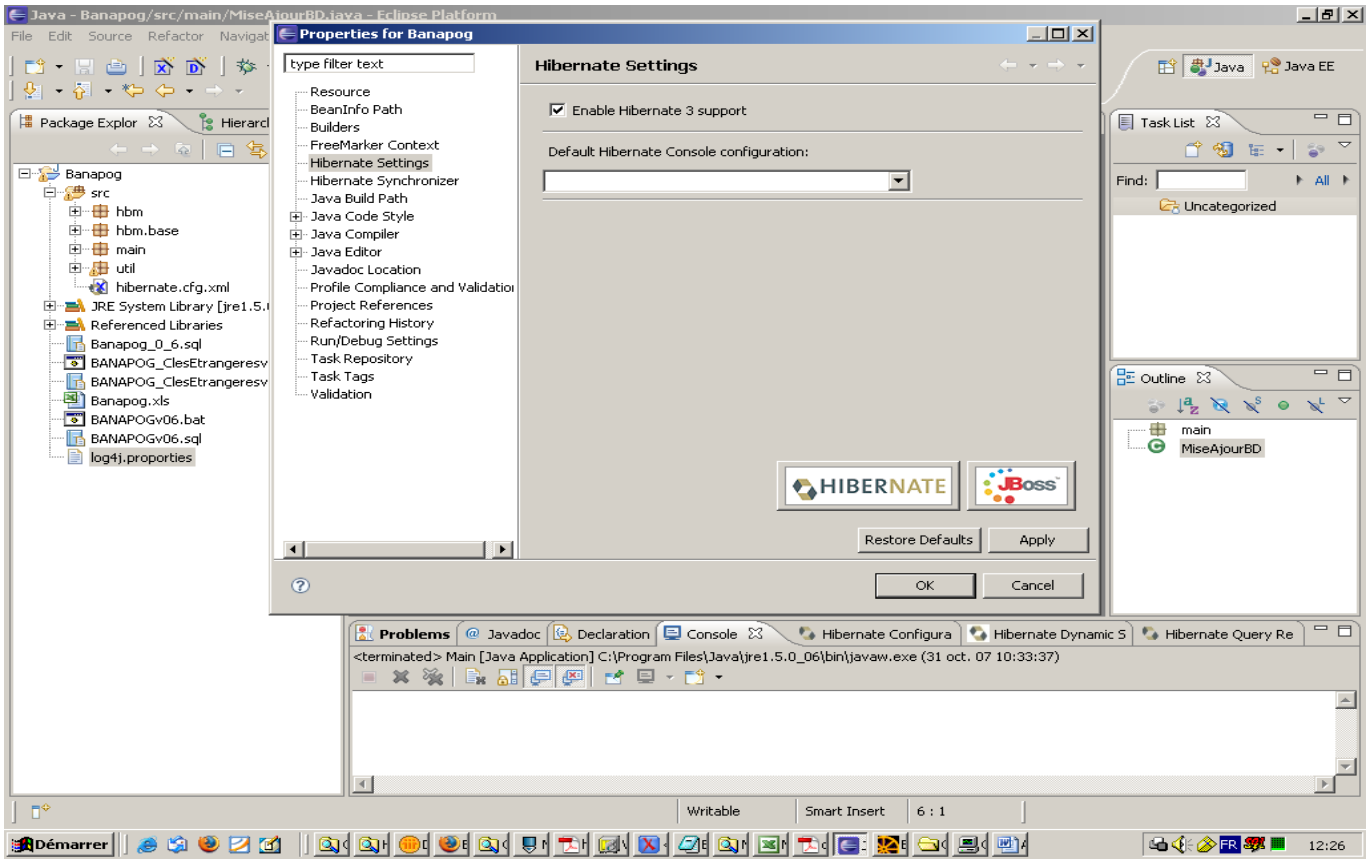
| Nom | Version | Commentaires |
|---|---|---|
| IDE Eclipse | Europa | Prendre la J2E Developpers |
| Hibernate core | 3.2.0 beta 11 | JDK 1.4 au moins |
| Hibernate synchronizer ou Hibernate tools | 3.1.9 pour Synchronizer et 3.2 beta 11 pour Hibernate tools | Hibernate tools plus performant mais c'est une beta, il y a encore des soucis sur les mappings complexes |
| BDD Sybase SQL Anywhere | 7 au moins | |

### Installation d'Eclipse

- Récupérer le fichier zip à l'adresse http://www.eclipse.org/downloads/moreinfo/jee.php,
- Le dézipper à un endroit quelconque du micro. Aucune autre installation n'est nécessaire.

### Installation d'Hibernate

Eclipse Europa intègre le noyau d'Hibernate 3, il faut juste activer son utilisation dans les settings du projet :
(Pour toute autre version, il faut télécharger le plugin Hibernate, le placer dans le dossier plugin d'Eclipse)




### Installation d'Hibernate tools ou Hibernate Synchronizer

Ces deux outils automatisent la génération des fichiers de mapping, des classes persistantes et dans certains cas des classes DAO pour l'accès à la base de données.

Ils s'installent sous forme de plugins Eclipse :

- Récupérer et dézipper l'outil à utiliser,
- Placer le contenu du dossier « plugin » obtenu dans le dossier « plugin » d'Eclipse,
- Placer le contenu du dossier « features » obtenu dans le dossier « features » d'Eclipse

**Remarque importante1** : Eclipse Europa utilisant la version 3 d'Hibernate, si on veut utiliser Synchronizer, il faut installer sa version 3.1.9 car les versions précédentes de Synchronizer utilisent Hibernate 2.

**Remarque importante 2**: Hibernate tools (Outil officiel d'Hibernate) parait plus complet et plus performant que Hibernate Synchronizer mais des soucis sont apparus à la génération des fichiers de mapping sur des tables avec plusieurs colonnes comme clé primaire.

**Remarque importante 3** : La génération des fichiers de mapping, des classes persistantes, des classes DAO marche complètement avec Hibernate Synchronizer mais les associations relationnelles entre tables ne sont pas correctement mappées.

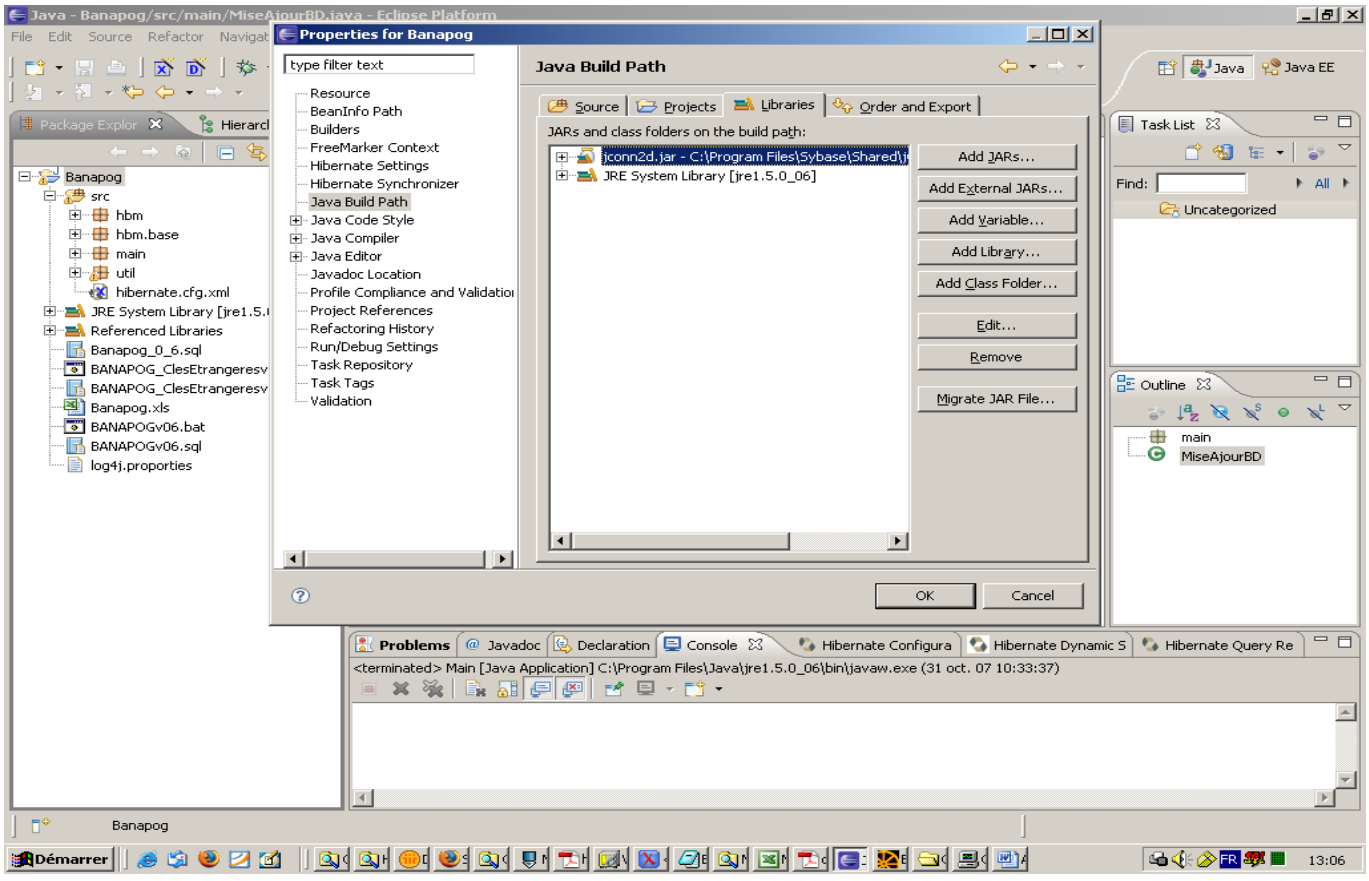
**Conseil** : Utiliser Hibernate Synchronizer pour la génération des mappings et retoucher les fichiers générés.

**Paramétrage du driver jdbc de la base de données**

Il y'a deux drivers jdbc pouvant sonctionner avec une base de données de type Sybase :

- Jconnect (C:\Program Files\Sybase\Shared\jConnect-5_2\classes\jconn2.jar)
- Jodbc (C:\Program Files\Sybase\SQL Anywhere 7\java\jdbcdr.jar)

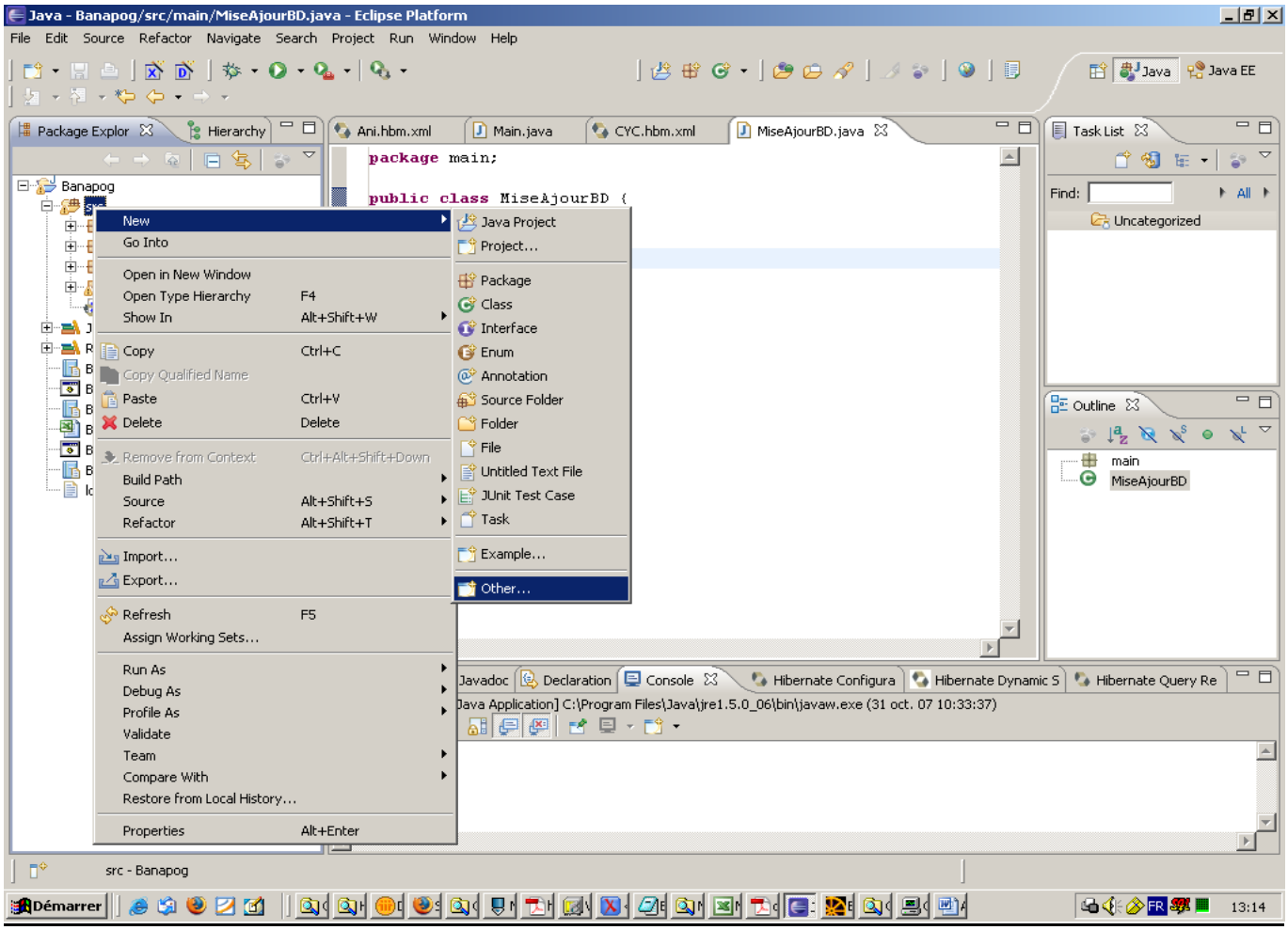
Il faut rajouter le path du driver utilisé dans les settings d'Eclipse :




## Utilisation d'Hibernate Synchronizer (plus ou moins valable pour Tools)

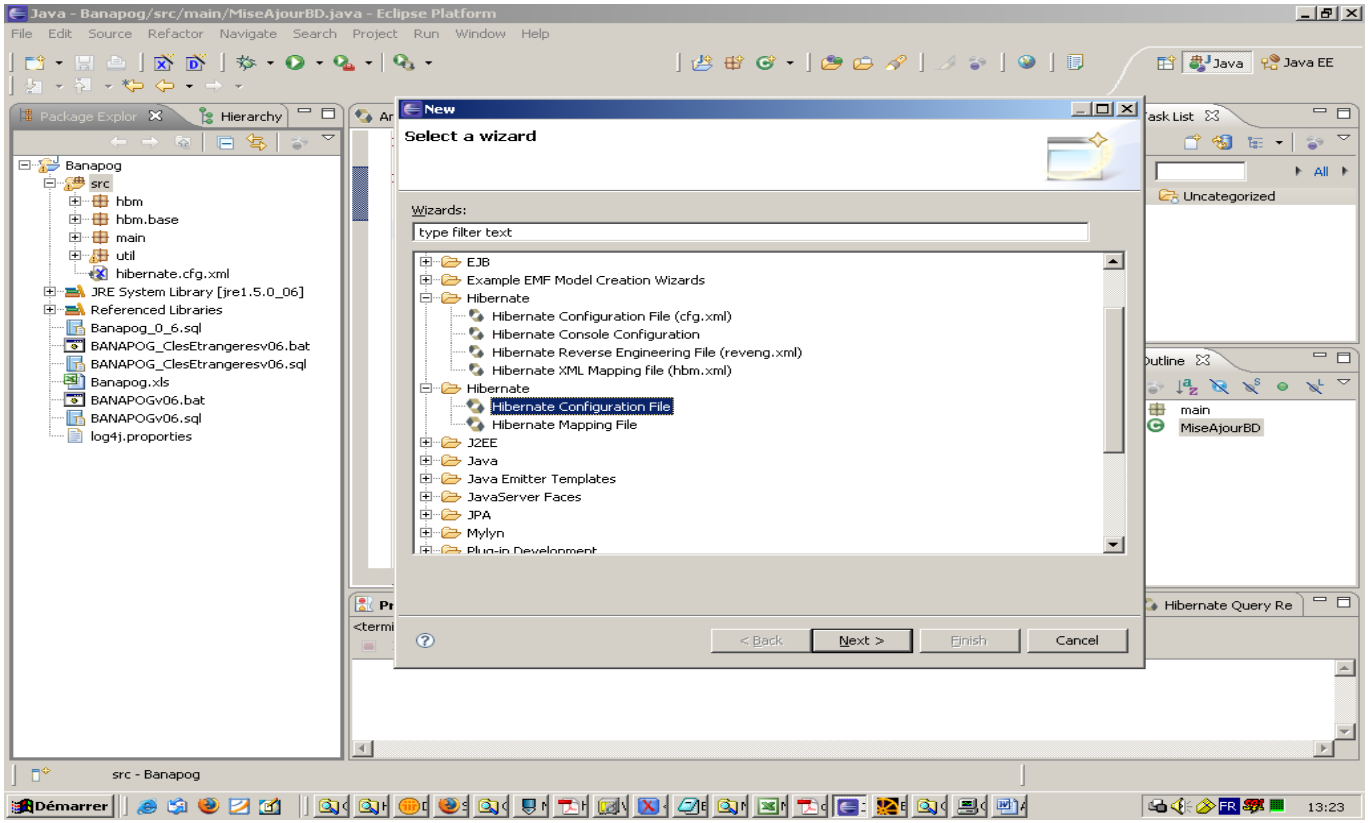
### Accès aux outils de Synchronizer ou de Tools

Synchronizer ou tools sont accéssibles via le menu New -> Other… d'Eclipse



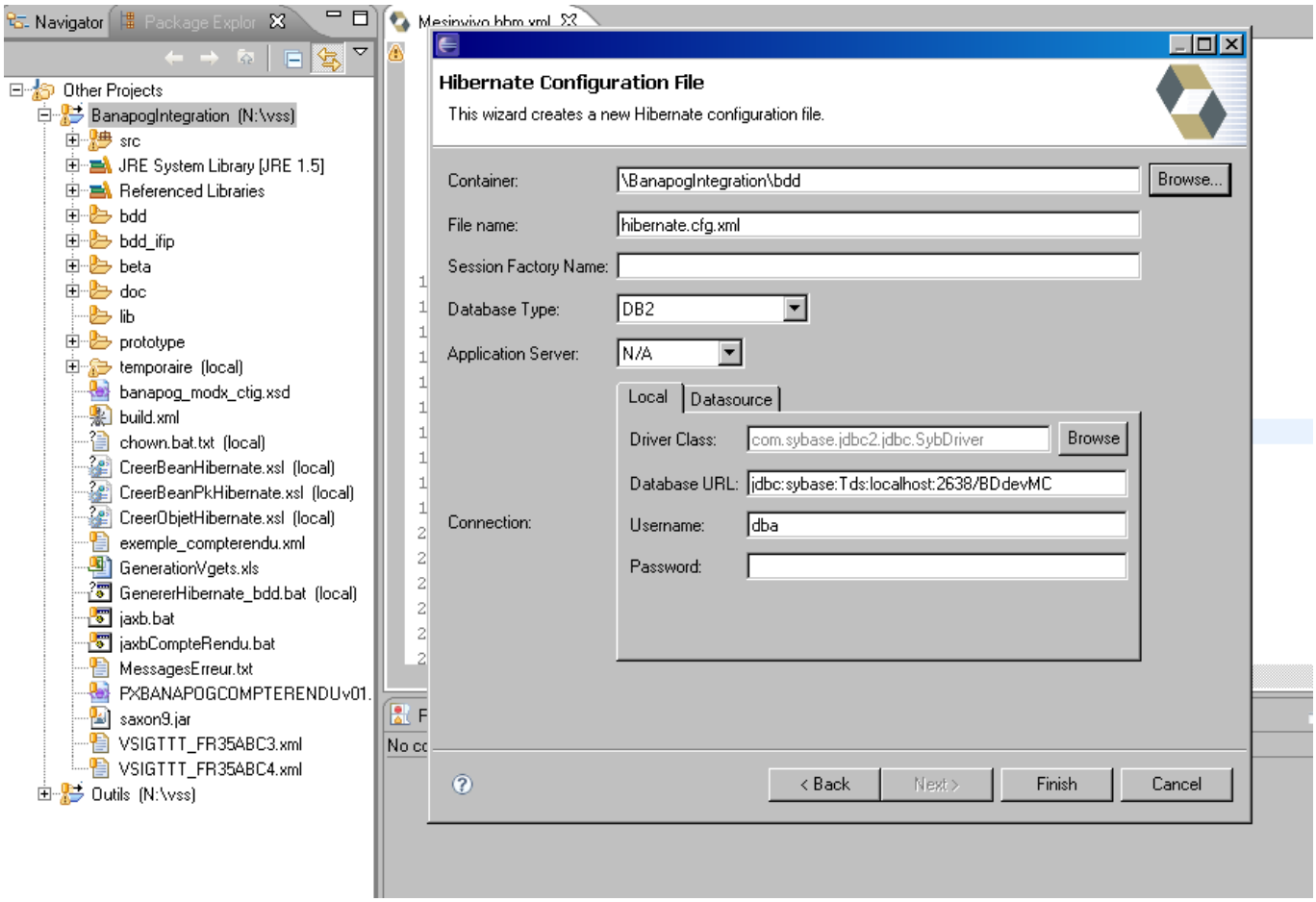

Ce menu donne alors accès à Hibernate Synchronizer ou à Hibernate tools

## Création du fichier de configuration hibernate.cfg.xml



Ce fichier de configuration Hibernate est à placer dans le dossier **bdd** du projet où se trouvent les fichiers correspondant au paramétrage de la base de données

Il faut ensuite fournir les informations de connexion à la base de données comme suit :

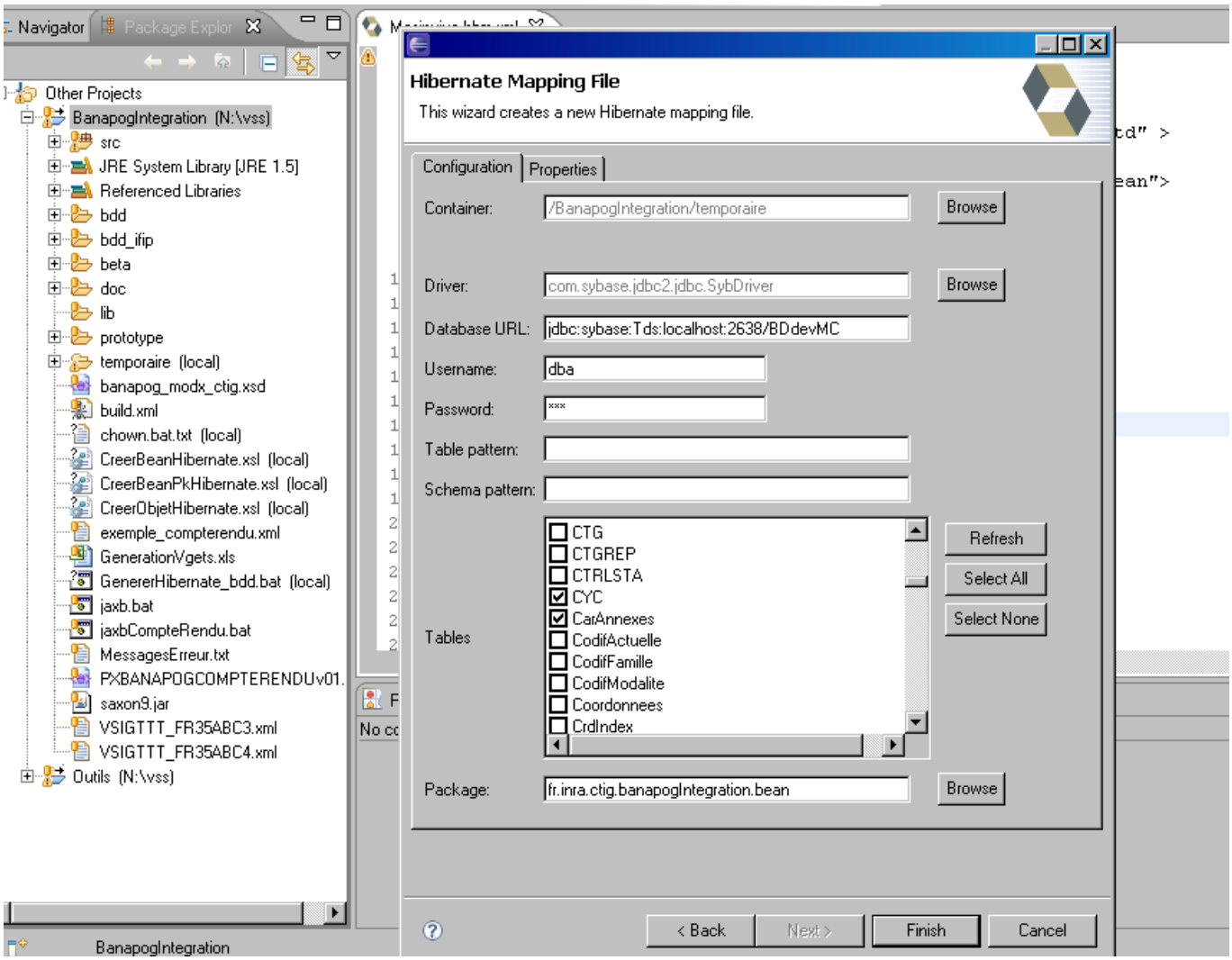



## Génération des fichiers de mapping hibernate.hbm.xml

Synchronizer et tools possèdent chacun un outil de reverse engineering pour la génération des fichiers de mapping et des classes persistantes à partir de la base de données :

Les objets de mapping Hibernate (*.hbm.xml) sont à placer dans un dossier **temporaire** du projet, créer auparavant par `New -> Folder`.

Lancer le gestionnaire de base de données (Sybase).




**<u>Génération des classes Java hibernate (bean) et étendues (metier)</u>**
Pour la génération de ces classes (prêtes à l'emploi dans les développements) on utilise des « transformateurs » xsl qui copient dans des répertoires distincts les nouveaux éléments en les nommant suivant les rèbles en vigueur.

Ces transformations se font grâce à un .bat : **GenererHibernate_bdd.bat**

On aura dans le répertoire **temporaire** : les répertoires **metier bean** et **bdd** avec les objets et classes générées.

Si on doit récupérer les associations non générées par Hibernate, on fait la comparaison avec l'existant dans le package du projet

Ces outils (GenererHibernate_bdd.bat qui utilise CreerBeanHibernate.xsl CreerBeanPkHibernate.xsl et CreerObjetHibernate.xsl ainsi que saxon9.jar) sont pour l'instant stockés dans le projet Banapog.

# Fonctionnement d'Hibernate

Les objectifs d'Hibernate sont :

- Faire la correspondance entre un schéma relationnel et un shéma objet,
- Masquer aux applications les opérations d'interactions avec la base de données,
- Rendre transparentes ces opérations,
- Etc.

Pour atteindre ces objectifs, le développeur doit fournir un certain nombre d'informations.

Celles – ci se présentent sous forme de fichiers de mapping.

## Gestion des aspects de configuration d'Hibernate

### Structure d'un fichier de configuration 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>

     <property name="hibernate.connection.driver_class">com.sybase.jdbc2.jdbc.SybDriver</property>
    <property name="hibernate.connection.password">sql</property>
     <property name="hibernate.connection.url">jdbc:sybase:Tds:localhost:2638/BDdevMC</property>
     <property name="hibernate.connection.username">dba</property>
     <property name="hibernate.dialect">org.hibernate.dialect.SybaseAnywhereDialect</property>

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

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

     <mapping resource="ANI.hbm.xml"/>
     <mapping resource="CYC.hbm.xml"/>
     <mapping resource="SAI.hbm.xml"/>

  </session-factory>

</hibernate-configuration>
```

Ce fichier permet de configurer Hibernate, les élements importants sont :

- Les informations sur la base de données utilisées (driver + url de la BD),

- Les relations avec les fichiers de mapping des tables (Il est permis d'avoir un seul fichier de mapping pour toutes les tables mais il est de bonne pratique d'utiliser un fichier de mapping par),

- La documentation officielle d'Hibernate en français http://www.hibernate.org/hib_docs/v3/reference/fr/html explique le role et les valeurs possibles de chaque paramètre.

- Le role du paramètre current_session_context_class est expliqué dans la partie « Gestion des aspects fonctionnels d'Hibernate ».

**Structure d'un fichier de mapping hibernate.hbm.xml**

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

<hibernate-mapping>

   <class name="fr.inra.ctig.banapog.bdd.SAI" table="DBA.SAI">

         <composite-id name="saipk" class="fr.inra.ctig.banapog.bdd.SAIPK" >
               <key-property name="sitesai" column="SiteSai"/>
               <key-property name="nonat" column="NoNat"/>
               <key-property name="nocyc" column="NoCyc"/>
         </composite-id>

         <property name="nosai" column="NoSai" type="integer" not-null="true" length="10"/>
         <property name="datesai" column="DateSai" type="date" not-null="true" length="23"/>
         <property name="typesai" column="TypeSai" type="string" not-null="false" length="2"/>
         <property name="typeapsai" column="TypeApSai" type="string" not-null="false" length="1"/>
         <property name="nobande" column="NoBande" type="string" not-null="false" length="6"/>
         <property name="dcre" column="dcre" type="timestamp" not-null="true" length="23"/>
         <property name="dmaj" column="dmaj" type="timestamp" not-null="true" length="23"/>
         <property name="orid" column="orid" type="string" not-null="true" length="1"/>
         <property name="vali" column="vali" type="string" not-null="true" length="256"/>

         <many-to-one name="cyc" class="fr.inra.ctig.banapog.bdd.CYC" insert="false" update="false">

               <column name="SiteCycle"/>
               <column name="NoNat"/>
               <column name="NoCyc"/>

         </many-to-one>

   </class>

</hibernate-mapping>
```

Ce fichier correspond au mapping d'une table en base de données (ici la table SAI).

Les élements importants de ce fichier sont la déclaration de la clé primaire et l'association de type « many-to-one » vers la table CYC. Les élements de type « proporty » correspondent aux colonnes de la table en BDD.

Le role de ces élements est expliqué dans la partie suivante.

## Analogie entre elements relationnels et elements objets d'Hibernate

| BDD | HIBERNATE |
|---|---|
| Tables relationnelles | Classes persistantes |
| Enregistrements d'une table | Instances de classes (Objets) |

| | |
|---|---|
| Associations relationnelles (Clés étrangères, tables de jointure, …) | Associations objets (Héritage, polimorphisme, …) |
| SQL (Requetes sur des enregistrements) | HQL (requetes sur des objets) |
| Schéma relationnel | Schéma objets (Fichiers de mapping) |

**Mapping des tables avec une clé primaire simple**

Le mapping d'une table avec clé primaire formée d'une seule colonne est très simple.

Exemple avec ANI :

```
<?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="fr.inra.ctig.banapog.bdd.ANI" table="DBA.ANI">
                <id name="nonat" type="string" column="NoNat"/>
                <property name="datenaiss" column="DateNaiss" type="date" not-null="true" length="23"/>
                <property name="sexe" column="Sexe" type="string" not-null="true" length="1"/>
                <property name="noorig" column="NoOrig" type="string" not-null="false" length="20"/>
                <property name="ctg" column="Ctg" type="string" not-null="false" length="5"/>
                <property name="nonatpere" column="NoNatPere" type="string" not-null="false" length="16"/>
                <property name="ctgpere" column="CtgPere" type="string" not-null="false" length="5"/>
                <property name="nonatmere" column="NoNatMere" type="string" not-null="false" length="16"/>
                <property name="ctgmere" column="CtgMere" type="string" not-null="false" length="5"/>
                <property name="tetinesdb" column="TetinesDB" type="integer" not-null="false" length="10"/>
                <property name="tetinesgb" column="TetinesGB" type="integer" not-null="false" length="10"/>
                <property name="halothane" column="Halothane" type="string" not-null="false" length="3"/>
                <property name="statutrep" column="StatutRep" type="string" not-null="false" length="1"/>
                <property name="ctgnaiss" column="CtgNaiss" type="string" not-null="false" length="5"/>
                <property name="ctgrep" column="CtgRep" type="string" not-null="false" length="5"/>
                <property name="codfiliation" column="CodFiliation" type="string" not-null="false" length="2"/>
                <property name="nonatport" column="NoNatPort" type="string" not-null="false" length="16"/>
                <property name="cocr" column="Cocr" type="string" not-null="false" length="1"/>
                <property name="siteani" column="SiteAni" type="string" not-null="false" length="7"/>
                <property name="typetat" column="TypeTat" type="string" not-null="false" length="2"/>
                <property name="dcre" column="dcre" type="timestamp" not-null="true" length="23"/>
                <property name="dmaj" column="dmaj" type="timestamp" not-null="true" length="23"/>
                <property name="orid" column="orid" type="string" not-null="true" length="1"/>
                <property name="vali" column="vali" type="string" not-null="true" length="256"/>

                <set name="cycs" inverse="true">
                        <key column="NoNat"/>
                        <one-to-many class="fr.inra.ctig.banapog.bdd.CYC"/>
                </set>
   </class>
</hibernate-mapping>
```

Ce fichier déclare les elements suivants :

- « class » : Mapping de la table « **DBA.ANI** » en classe persistante Hibernate **fr.inra.ctig.banapog.bdd.ANI**,
- « id » : Mapping de clé primaire « **NoNat** » en attribut de classe « **nonat** » de type Hibernate « string »,
- « proporty » : Mapping des autres **colonnes** de la table en **attributs** de classe de type Hibernate simple, Les attributs de classes sont renseignés dans l'attribut name de la balise « proporty ».
- « set » : Mapping de la **dépendance relationnelle** entre une table « DBA.ANI » et une table « DBA.CYC » en **dépendance Objet** entre la classe « fr.inra.ctig.banapog.bdd.ANI » et la classe « fr.inra.ctig.banapog.bdd.CYC »,

- « one-to-many » : Précise la **nature de la dépendance** entre « DBA.ANI » et « DBA.CYC ». Ici à un objet fr.inra.ctig.banapog.bdd.ANI, est associée une liste d'Objets fr.inra.ctig.banapog.bdd.CYC.

**Mapping des tables avec une clé primaire composé**

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

<hibernate-mapping>

   <class name="fr.inra.ctig.banapog.bdd.SAI" table="DBA.SAI">

         <composite-id name="saipk" class="fr.inra.ctig.banapog.bdd.SAIPK" >
               <key-property name="sitesai" column="SiteSai"/>
               <key-property name="nonat" column="NoNat"/>
               <key-property name="nocyc" column="NoCyc"/>
         </composite-id>

         <property name="nosai" column="NoSai" type="integer" not-null="true" length="10"/>
         <property name="datesai" column="DateSai" type="date" not-null="true" length="23"/>
         <property name="typesai" column="TypeSai" type="string" not-null="false" length="2"/>
         <property name="typeapsai" column="TypeApSai" type="string" not-null="false" length="1"/>
         <property name="nobande" column="NoBande" type="string" not-null="false" length="6"/>
         <property name="dcre" column="dcre" type="timestamp" not-null="true" length="23"/>
         <property name="dmaj" column="dmaj" type="timestamp" not-null="true" length="23"/>
         <property name="orid" column="orid" type="string" not-null="true" length="1"/>
         <property name="vali" column="vali" type="string" not-null="true" length="256"/>

         <many-to-one name="cyc" class="fr.inra.ctig.banapog.bdd.CYC" insert="false" update="false">

               <column name="SiteCycle"/>
               <column name="NoNat"/>
               <column name="NoCyc"/>

         </many-to-one>

   </class>

</hibernate-mapping
```

Ce fichier déclare les elements suivants :

- « composite-id » : Mapping de la **clé composé** (SiteSai, NoNat, NoCyc) de la table SAI en classe Hibernate **fr.inra.ctig.banapog.bdd.SAIPK**.

  Hibernate mappe les clés composées de cette façon. La classe persistante est fr.inra.ctig.banapog.bdd.SAI alors que fr.inra.ctig. banapog.bdd.SAIPK est une simple classe POJO,

- « many-to-one » : Mapping de la **dépendance relationnelle** entre une table « DBA.SAI » et une table « DBA.CYC » en **dépendance Objet** entre la classe « fr.inra.ctig. banapog.bdd.SAI » et la classe « fr.inra.ctig.banapog.bdd.CYC ».
  Ici plusieurs Objets « fr.inra.ctig.banapog.bdd.SAI » sont associés à un Objet « fr.inra.ctig.banapog.bdd.CYC »,

**Remarque importante sur le mapping 1**: L'attribut « type » de la balise « proporty » ne correspond ni à un type Java ni à un type Sql mais à un type intermédiaire Hibernate.
Cet attribut est optionnel sauf pour les types « date » et « timestamp », Hibernate faisant de l'introspection pour déterminer le type Java correspondant.

**Remarque importante sur le mapping 2**: Il faut obligatoirement renseigner l'attribut « type » de « proporty » pour les attributs de type « date » ou « timestamp » car ce sont aussi des types Sql.

**Remarque importante sur le mapping 3** : Dans une association bidirectionnelle, comme c'est le cas ici entre « fr.inra.ctig.banapog.bdd.SAI » et « fr.inra.ctig.banapog.bdd.CYC », doit se faire en déclarant un « **set** » dans le fichier de mapping du contenant (CYC) et une relation « **many-to-one** » dans le fichier de mapping du contenu (SAI).

**Conseil** : Utiliser Hibernate Synchronizer pour la génération des fichiers de mappings et définir soit même les associations pour éviter tout risque d'erreurs.

## Gestion des aspects fonctionnels d'Hibernate

Dans les aspets fonctionnels d'Hibernate, on peut distinguer deux couches :

- La gestion des sessions et des contextes,
- La gestion des objets métiers sans utiliser le langage HQL
- La gestion des objets métiers en utilisant le langage HQL

### Gestion des sessions et des transactions

La balise « session-factory » du fichier de configuration « hibernate.cfg.xml » représente une base de données et permet d'obtenir une classe Hibernate : la **SessionFactory** dont le role est de mettre à la dispositon de l'application des sessions (sur cette base de données) créées à la demande

Il faut donc plusieurs fichiers de configuration ou un fichier de configuration avec plusieurs balises « session-factory » pour pouvoir utiliser plusieurs bases de données dans une meme application.

Une session, elle, encapsule une connexion JDBC et représente une discussion plus ou moins longue avec la base de données. Une application peut ouvrir et fermer plusieurs sessions

La valeur « thread » du paramètre suivant du fichier de configuration « hibernate.cfg.xml » :

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

simplifie la gestion des sessons en rattachant toutes les sessions utilisées par l'application au Thread d'exécution de celle-ci.

Les contextes sont obtenus à partir d'une session et permettent, grace à un appel à « **commit** » ou à « **rollback** », de valider ou d'invalider les opérations sur la base de données.

**Remarque importante sur les sessions 1**: La multiplication des sessions au sein d'une même application Hibernate peut couter en performance, le modèle de developpement le plus pratiqué dans Hibernate consiste à regrouper le maximum d'opérations au sein d'une session quitte à utiliser plusiers contextes de validation.

**Remarque importante sur les sessions 2**: La couche de gestion des sessions et des contextes est complètement séparée de celle de gestion des objets persistants.
Sous Hibernate, une pratique courante consiste à créer une classe « HibernateUtil » encapsulant cette couche, de façon à concentrer le code sur la couche métier.

```
public class HibernateUtil {
    public static final SessionFactory sessionFactory;
```

```
    static {
        try {
            // Création de la SessionFactory à partir de hibernate.cfg.xml
            sessionFactory = new
Configuration().configure("hibernate.cfg.xml").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 final ThreadLocal session = new ThreadLocal();

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

**Conseil à discuter** : Utilisation d'une seule session pour Banapog avec des contextes pour les différentes operations (ajout, suppression, mise à jour, etc.).
De même, une classe « HibernateUtil » permet de masquer l'utilisation de cette couche dans le code métier.

## Travailler avec des objets métier

Hibernate fournit un langage de requetage orientée objet sur les entités persistantes : le HQL

Mais pour rester dans une logique purement objet, Hibernate permet aussi de se passer de ce langage notamment lorsque les entités persistantes sont connues et identifiées. Une entité est connue par son identifiant persistant.

Exemple d'utilisation des objets métier :

**Recherche d'entités persistantes :**

```
CTIGANI ani = (CTIGANI) session.get("FR00000200000160", CTIGANI.class) ;
CTIGANI ani = (CTIGANI) session.load("FR00000200000160", CTIGANI.class) ;
```

Ces deux méthodes renvoient l'entité ani identifiée en base de données par FR00000200000160
« load » déclenche une exception non rattrapable lorsque l'entité n'est pas connue en base de données tandis que « get » renvoie « null »

Il sera, la plupart du temps plus interessant d'utiliser « get » au lieu de « load » quite à tester la valeur de retour.

**Insertion ou mise à jour d'entités persistantes :**

```
CTIGANI ani = new CTIGANI() ;
Ani.setNoNat("FR00000200000160") ;
Session.save("FR00000200000160") ; // Ajout
Session.update("FR00000200000160") ; // Mise à jour
Session.saveOrUpdate("FR00000200000160") ; // Ajout lorsque l'entité n'existe pas, mise à jour sinon
```

## HQL : langage de requetage objet

Le HQL est un langage de requetage pratique sur les Objets Hibernate.

Dans certains cas son utilisation s'impose comme la seule solution. Par exemple lorsqu'on recherche des entités dont les identifiants ne sont pas connues

Exemples de clauses HQL

Inserti

- Récupération de tous les objets ani : from ani
- Récupération de certains anis : select ani from ANI ani where ani.sexe = 'sexe_ani'

**Mises à jour dans une session et une transaction**

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();

String hqlUpdate = "update ANI a set a.ctg = :newCtg where a.nonat = :nonat and
                    a.ctg = :oldCtg";
int updatedEntities = session.createQuery( hqlUpdate )
        .setString( "NoNat", nonat )
        .setString( "newCtg", newCtg )
        .setString( "oldCtg", oldCtg )
        .executeUpdate();
tx.commit();
session.close();
```