I - Tests de persistance (h2, dbUnit, ...)

1. Embedded database (hsqldb , h2)

1.1. Fonctionnalités

Caractéristiques des bases embarquées

* pas de serveur (SGBDR) séparé , ni de communication réseau .

* **fonctionnement directement en mémoire dans le processus client qui effectue les appels JDBC/SQL.**

* comportement pourtant classique (traitement ordinaire des requêtes SQL avec transactions et verrous).

* **très rapide**

* pratique pour les phases de test.

1.2. Modes "in memory" et "local"

2 modes (classiques) d'utilisation (bases embarquées)

* seulement **en mémoire** (et à ré-initialiser à chaque fois) pour des séquences de test :
    a) create table
    b) insert into (jeux de données)
    c) opérations SQL diverses
    d) (éventuellement) drop table (pas nécessaire)

* **avec une réelle persistance locale (sur disque)**
    - même comportement (et fonctionnalités) qu'un SGBDR classique mais avec un fonctionnement local (pas d'accès réseau , meilleur performances)
    - structures et valeurs conservées après un arrêt et redémarrage.

1.3. hsqldb et h2

Bases embarqués "java" (H2 et HsqlDB)

	H2	HsqlDb
Editeur	Thomas Mueller *(H2 signifie Hypersonic 2)*	The HSQL Development Group
Driver jdbc	**org.h2.Driver**	**org.hsqldb.jdbc.JDBCDriver**
URL (memory)	**jdbc:h2:mem** *ou bien* **jdbc:h2:mem:dbName**	**jdbc:hsqldb:mem:mymemdb**
URL (disk)	**jdbc:h2:file:/opt/dbXy** *(file : par défaut)* **jdbc:h2:~/testDB** *(où ~ désigne $HOME)*	**jdbc:hsqldb:file:/opt/testdb**
Default auth.	username="sa" password=""	username="SA" password=""
...		

Principales spécificités de HsqlDB

Dépendance "maven" pour HsqlDB :

```
<dependency>
    <groupId>org.hsqldb</groupId>
    <artifactId>hsqldb</artifactId>
    <version>2.3.3</version>
</dependency>
```

Principales spécificités de H2

Dépendance "maven" pour H2 :

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.190</version>
</dependency>
```

Lancement console web de H2 (url=http://localhost:8082):

cd h2/bin
java -jar h2*.jar (ou bien ***h2.bat*** ou bien ***double click sur h2*.jar***)

Lancement d'un script sql (h2) :

java org.h2.tools.RunScript -url *jdbc:h2:~/test* -user *sa* -script *xy.sql*

2. DbUnit

Présentation de DbUnit

* ***DbUnit est une extension pour JUnit*** qui, comme son nom l'indique, est ***dédiée aux tests de persistance***.

* **DbUnit sert essentiellement à vérifier que les ordres java *(jdbc/sql ou jpa/hibernate ou …)* sont bien répercutés en base *(valeurs bien actualisées dans les tables?)*.**

* DbUnit permet de **facilement manipuler** des "**dataSet**" *(jeux de données)* et tout particulièrement de
  - *initialiser les valeurs en base au début des tests*
  - *récupérer des valeurs de certains enregistrements pour effectuer des comparaisons.*
  - *convertir automatiquement des jeux de valeurs au format XML (ce qui est pratique et lisible)*

* *En d'autres termes, avec DbUnit (qui ne s'appuie que sur JDBC) on peut tester si les valeurs insérées en base via JPA sont correctes (sachant qu'il ne vaut mieux pas vérifier avec la même technologie que le code effectif de l'application pour éviter des erreurs auto-compensées).*

Principales spécificités de DbUnit

Dépendance "maven" pour DbUnit :

```
<dependency>
	<groupId>org.dbunit</groupId>
	<artifactId>dbunit</artifactId>
	<version>2.5.1</version>
</dependency>
```

En complément de :

```
<dependency>
	<groupId>junit</groupId>
	<artifactId>junit</artifactId>
	<version>4.11</version>
	< !-- <scope>test</scope> -->
</dependency>
```

Exemple de jeux de données (en XML)

Jeu de données initial (src/test/resources/dataset/initialDataSet.xml):

```
<dataset>
    <!-- <TableName column1Name="value1_ofRow1" column2Name="value2_ofRow1""/> -->
    <!-- <TableName column1Name="value1_ofRow2" column2Name="value2_ofRow2""/> -->
    <!-- <Table eventuellement sans pkColumn="..." if auto_increment /> -->
   <!-- Attention: il faut respecter un ordre cohérent avec les contraintes d'intégrité référentielles →

     <Compte numCpt="1" label="compte courant"  solde="1200.0"  />
      ...

     <Operation numOp="1" dateOp="2010-12-24" label="achat xy"
                montant="-50"  ref_compte="1" />
 </dataset>
```

(sous) jeu de données (pour vérifier mise à jour):

```
<dataset>
    <Client  prenom="alex" nom="therieur"  dateNaissance="1980-02-11"
      ref_adressePrincipale="1"    password="pwd2"  telephone="0504030201"
       email="alex.therieur@ici_ou_la" />
 </dataset>
```

Exemple d'initialisation de valeurs (java + dbUnit)

```
import org.dbunit.DataSourceDatabaseTester;
import org.dbunit.IDatabaseTester;
import org.dbunit.dataset.IDataSet;
import org.dbunit.dataset.xml.FlatXmlDataSetBuilder;

public class BasicDBUnitHelper {

protected String initialDataSetFileName="initialDataSet.xml"; //by default
protected String dataSetDirectory="src/test/resources/dataset"; //by default for maven
protected IDatabaseTester   databaseTester;
protected IDataSet   initialDataSet=null;
protected DataSource   dataSource; //+get/set

public void reInitDataBaseFromInitialDataSet() throws Exception{
//databaseTester = new JdbcDatabaseTester("org.hsqldb.jdbcDriver","jdbc:hsqldb:sample", "sa", "");
this.databaseTester = new DataSourceDatabaseTester(this.dataSource);

// initialize your dataset here
String initialDataSetPathName = this.dataSetDirectory+"/"+this.initialDataSetFileName;
FlatXmlDataSetBuilder fxdsb = new FlatXmlDataSetBuilder();
this.initialDataSet = fxdsb.build(new File(initialDataSetPathName));
databaseTester.setDataSet( this.initialDataSet );
databaseTester.onSetup();// will call default setUpOperation (clean_insert by default)

}}
```

3. Automatisation tests de persistance

Pistes pour l'automatisation des tests de persistance

* préparer idéalement un "***DAO générique***" de façon à ce que les ordres CRUD élémentaires puissent être lancés d'une manière uniforme .
Chaque DAO concret de l'application pourra ainsi hériter de ce "DAO générique" et n'ajouter que le code supplémentaire spécifique (findByXy,...) .

* coder ensuite un ***mini framework de "test de persistance"*** en s'appuyant à fond sur DbUnit . Ce framework pourra ainsi tester automatiquement les ordres CRUD élémentaires via une séquence automatisée du type :
1) **ré-initialiser les valeurs en bases** via un "*initial-dataSet.xml*"
2) récupérer les valeurs d'un fichier "*new-xy.xml*" pour peupler (par réflexion/introspection) les valeurs d'un objet persistant java .
3) **appeler automatiquement une méthode** de type ***persist***(entity) ou ***saveNewEntity***(entity) sur le DAO (héritant d'une partie générique).
4) **vérifier l'absence d'exception , relire en java (via le dao) et comparer.**
5) **récupérer via dbUnit un dataSet de l'état d'une table *Xy*** de la base de donnée . Effectuer une **comparaison** entre le ***dataSet*** *récupéré* et la *fusion des* ***dataSet*** *"initial-dataSet.xml" et "new-xy.xml"* (***coïncidence attendue***).
6) idem pour test de "mise à jour", de "suppression" , de "recherche" , ...

Exemple de DAO générique

```
package org.mycontrib.generic.persistence; //dans generic-jee-back-utils-abstract
...
public interface GenericDao<T,ID extends Serializable> {
  public void deleteEntity(ID pk) throws GenericException; // remove entity from pk
  public void removeEntity(T e) throws GenericException; // remove entity
  public  T updateEntity(T e) throws GenericException; // update entity (and return it ref )
  public  T getEntityById(ID pk) throws GenericException; //return null if not found
  public  T persistNewEntity(T e) throws GenericException; // persist and return e with pk
  public  ID persistIdNewEntity(T e) throws GenericException; // persist and return id/pk
}
```

Exemple d'utilisation par héritage

```
public interface DaoDevise extends GenericDao<_Devise,String> {
        public _Devise getDeviseByName(String deviseName);
        public List<_Devise> getAllDevise();
}

public class DaoDeviseJpa extends GenericDaoJpaImpl<_Devise,String>
                        implements DaoDevise {
…
}
```

Séquence théoriquement re-déclenchable sans conflit

Séquence CRUD classique :

0) *jeu initial de données*
1) **ajout** d'une entité
2) *relecture (en mémoire) d'une liste d'entité pour vérification de ajout (+vérif. DbUnit/en base)*
3) **modification**
4) *relecture (en mémoire) de l'entité modifiée pour vérification mise à jour (+vérif. DbUnit/en base)*
5) **suppression** de l'entité ajoutée
6) *tentative de relecture de l'entité retirée qui doit normalement remonter une exception.*

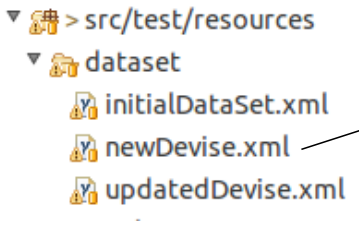
Exemple de code source au bout de **l'URL** suivante (pour ***git clone***) :

https://github.com/didier-mycontrib/mycontrib-utils/tree/master/generic-jee-back-test-common/src/main/java/org/mycontrib/generic/test

Exemple de test de persistance basé sur le mini-framework

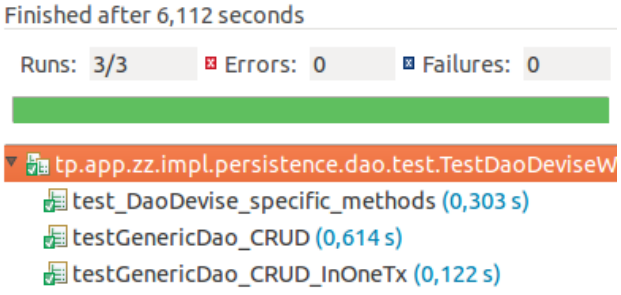
```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={"/serviceSpringConf.xml","/dataSourceForTestSpringConf.xml"})
@TransactionConfiguration(transactionManager="txManager",defaultRollback=false)
public class TestDaoDeviseWithGenericAndDbUnit
                    extends GenericDaoTestWithDbUnit<_Devise,String>{

    private DaoDevise dao = null; // dao à tester via test CRUD hérité

    @Override
    public String getPkOfEntity(_Devise entity){ return entity.getCodeDevise();
    }

    @Inject
    public void setDao(DaoDevise dao) { this.dao = dao;    this.setGenericDao(dao);
    }

    @Inject
     public void setDataSource(DataSource ds){ super.setDataSource(ds);
     }

    @Test
     public void test_DaoDevise_specific_methods() {...
     } }
```

En entrées:




```
<dataset>
    <Devise codeDevise="C"  dChange="1.0"
                monnaie="credit international"  />
</dataset>
```

Résultats:




```
** test CRUD sur T avec plusieurs petites transactions (via DAO GenericDao<T>) **

** test CRUD sur T en une seule transaction (via DAO GenericDao<T>) **
   id(pk) du nouveau Compte créé: C
   valeurs initiales de l'entité (créée):
            _Devise(dChange=1.0,monnaie=credit international,codeDevise=C)
   nouvelle valeur de l'entité (modifiée):
            _Devise(dChange=1.0,monnaie=credit intergalactique,codeDevise=C)
   entité bien supprimée                    ** end of CRUD test en une seule transaction **
```