Génie Logiciel
Introduction à NHibernate

BAPTISTE PESQUET

Exemples de code

https://github.com/ensc-glog/persistance-csharp/tree/master/examples

https://github.com/ensc-glog/HeritageNHibernate

Sommaire

- Présentation et configuration
- Possibilités de mapping
  - Classe
  - Association
  - Héritage
  - Export du schéma vers la BD
- Requêtage
  - HQL
  - Linq

Présentation et configuration

Présentation de NHibernate

ORM open source dérivé de Hibernate pour Java

Alternative plus simple à l'ORM Microsoft Entity Framework

S'obtient via NuGet (nécessite .NET Framework 4.6.1)

http://nhibernate.info

5 sur 37

Fichier de configuration

```
<!-- Hibernate.cfg.xml -->
<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
  <session-factory>
    <property name="connection.provider">NHibernate.Connection.DriverConnectionProvider</property>
    <property name="connection.driver_class">NHibernate.Driver.MySqlDataDriver</property>
    <property name="connection.connection_string">
      Server=localhost;Database=hoteldb;User ID=utilHotel;Password=secret;
    </property>
    <property name="dialect">NHibernate.Dialect.MySQL5Dialect</property>
    <mapping assembly="ORM" />
  </session-factory>
</hibernate-configuration>
```

6 sur 37

La notion de session

Session = unité de travail d'NHibernate

Créée par une **SessionFactory** initialisée à partir de la configuration

```
using NHibernate;
using NHibernate.Cfg;

// …

ISessionFactory sessionFactory = new
Configuration().Configure().BuildSessionFactory();

ISession session = sessionFactory.OpenSession();

// … Interactions avec la BD

session.Close();
```


Etapes de configuration

1. Version du framework .NET >= 4.6.1
2. Installation via NuGet
3. Fichiers ***hibernate.cfg.xml*** et ***MySql.Data.dll*** : copier dans le répertoire de destination
4. Fichiers de mapping ****.hbm.xml*** : ressource incorporée

Cf checklist

Mapping d'une classe

Le fichier de mapping d'une classe

Fichier XML nommé ***NomDeLaClasse*.hbm.xml**

Bonne pratique : regrouper tous les fichiers de mapping dans un répertoire **Mapping/**

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
                   assembly="NomDuProjetContenantLesClasses"
                   namespace="EspaceDeNomsDeCeProjet">
  <class name="NomDeLaClasse" table="NomDeLaTableAssocieeALaClasse">
    <!– Mapping des propriétés de la classe -->
    <!– Mapping des associations de la classe -->
  </class>
</hibernate-mapping>
```

La clé primaire (PK)

PK gérée manuellement par l'application

```
<id name="Id" column="nom_colonne" type="int">
  <generator class="assigned"></generator>
</id>
```

PK auto-incrémentée par NHibernate

```
<id name="Id" column="nom_colonne" type="int">
  <generator class="increment"></generator>
</id>
```

PK auto-incrémentée par MySQL (attribut AUTO_INCREMENT)

```
<id name="Id" column="nom_colonne" type="int">
  <generator class="native"></generator>
</id>
```

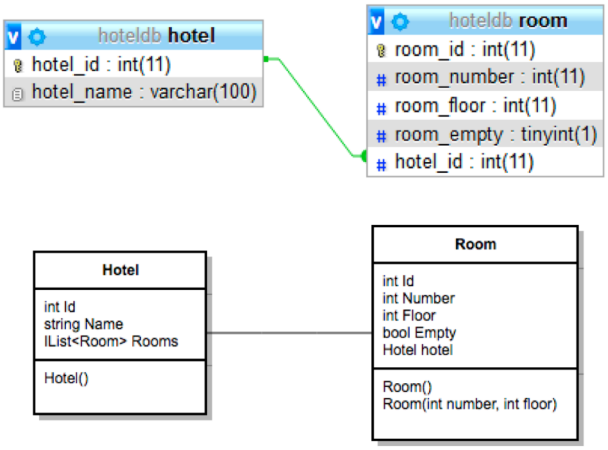

Les propriétés

Obligatoirement **virtuelles**

```
public virtual int Id { get; set; }
public virtual string Name { get; set; }
public virtual IList<Room> Rooms { get; set; }
```

Mapping d'une association 1 à plusieurs

Contexte d'exemple



Mappings 1:n et n:1

- Un-à-plusieurs

```
<bag name="Rooms" inverse="true" lazy="true">
  <key column="hotel_id"/>
  <one-to-many class="Room"/>
</bag>
```

- Plusieurs-à-un

```
<many-to-one name="Hotel" class="Hotel" column="hotel_id"/>
```


Mapping de la classe Hotel

```
public class Hotel
{
    public virtual int Id { get; set; }
    public virtual string Name { get; set; }
    public virtual IList<Room> Rooms { get; set; }

    // …
```

```
<!-- Fichier Hotel.hbm.xml -->
<class name="Hotel" table="hotel">
  <id name="Id" column="hotel_id" type="int">
    <generator class="native"></generator>
  </id>
  <property name="Name" column="hotel_name" not-
null="true"/>
  <bag name="Rooms" inverse="true" lazy="true">
    <key column="hotel_id"/>
    <one-to-many class="Room"/>
  </bag>
</class>
```

Mapping de la classe Room

```
public class Room
{
    public virtual int Id { get; set; }
    public virtual int Number { get; set; }
    public virtual int Floor { get; set; }
    public virtual bool Empty { get; set; }
    public virtual Hotel Hotel { get; set; }

    // …
```

```
<!-- Fichier Room.hbm.xml -->
<class name="Room" table="room">
  <id name="Id" column="room_id" type="int">
    <generator class="native"></generator>
  </id>
  <property name="Number" column="room_number" not-null="true"/>
  <property name="Floor" column="room_floor" not-null="true"/>
  <property name="Empty" column="room_empty" not-null="true"/>
  <many-to-one name="Hotel" class="Hotel" column="hotel_id"/>
</class>
```


L'attribut « inverse »

Nécessaire pour les associations **bidirectionnelles**

Permet de définir le côté d'une association 1:n ou n:n responsable des mises à jour BD

https://stackoverflow.com/a/713666/2380880

L'attribut « lazy »

Permet d'activer le **lazy loading** (chargement tardif)

Les associations ne sont naviguées (= génération et exécution d'une requête SQL) qu'au moment de l'accès à la propriété

=> Gain potentiel en performances

L'attribut « cascade »

Permet d'appliquer automatiquement des actions aux objets associés à un objet persistant

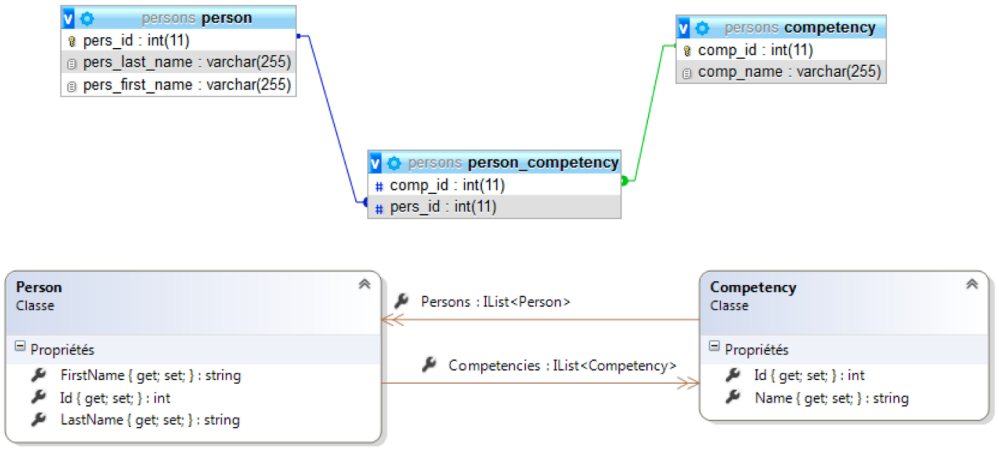
Valeurs fréquentes :

- *None*: aucune action
- *Delete*: supprime automatiquement les objets associés
- *All*: ajout/modifie/supprime automatiquement les objets associés

https://stackoverflow.com/a/1994447/2380880

Mapping d'une association plusieurs à plusieurs

Contexte d'exemple



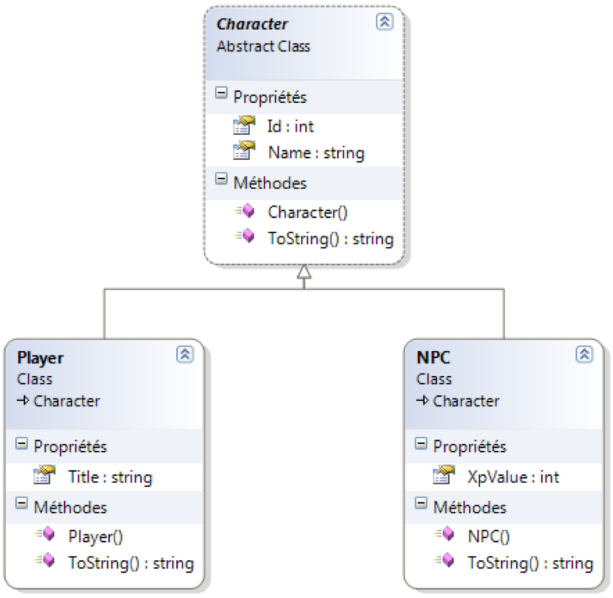
Mapping n:n

```
<class name="Person" table="person">
  <bag name="Competencies" table="Person_Competency">
    <key column="pers_id"/>
    <many-to-many class="Competency" column="comp_id"/>
  </bag>

<class name="Competency" table="competency">
  <bag name="Persons" inverse="true" table="Person_Competency">
    <key column="comp_id"/>
    <many-to-many class="Person" column="pers_id"/>
  </bag>
```


Mapping de l'héritage

Contexte

https://github.com/ensc-glog/HeritageNHibernate



Problématique

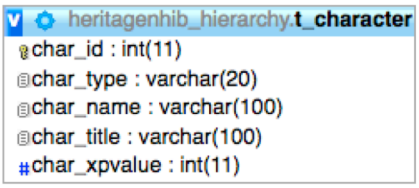
Problème épineux : les SGBDR ne supportent pas le concept d'héritage

Plusieurs stratégies possibles :

- Une table pour toute la hiérarchie
- Une table par classe concrète
- Une table par type

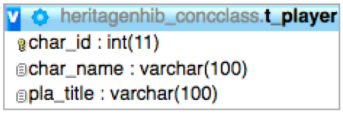
Une table pour toute la hiérarchie

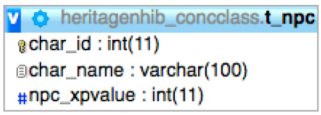
```
<class name="Character" table="t_character" abstract="true">
    <id name="Id" column="char_id" type="int">
        <generator class="native">
        </generator>
    </id>
    <discriminator column="char_type" type="String" />
    <property name="Name" column="char_name" />
    <subclass name="Player" discriminator-value="PLA">
        <property name="Title" column="char_title" />
    </subclass>
    <subclass name="NPC" discriminator-value="NPC">
        <property name="XpValue" column="char_xpvalue" />
    </subclass>
</class>
```



Une table par classe concrète

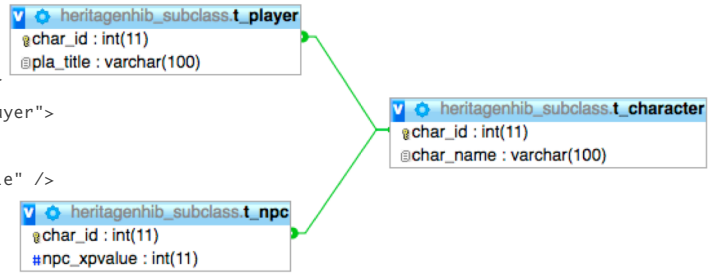
```
<class name="Character" abstract="true">
    <id name="Id" column="char_id" type="int">
        <generator class="increment">
        </generator>
    </id>

    <property name="Name" column="char_name" />
    <union-subclass name="Player" table="t_player">
        <property name="Title" column="pla_title" />
    </union-subclass>
    <union-subclass name="NPC" table="t_npc">
        <property name="XpValue" column="npc_xpvalue" />
    </union-subclass>
</class>
```





Une table par type

```
<class name="Character" table="t_character" abstract="true">
    <id name="Id" column="char_id" type="int">
        <generator class="native">
        </generator>
    </id>
    <property name="Name" column="char_name" />
    <joined-subclass name="Player" table="t_player">
        <key column="char_id" />
        <property name="Title" column="pla_title" />
    </joined-subclass>
    <joined-subclass name="NPC" table="t_npc">
        <key column="char_id" />
        <property name="XpValue" column="npc_xpvalue" />
    </joined-subclass>
</class>
```



Choix d'une stratégie

Elles ont toutes des avantages et des inconvénients

Le choix dépend du contexte

- Variable-clé : nombre de propriétés communes

Dans le doute, aller au plus simple : table par hiérarchie

Export du schéma vers la BD

Permet de synchroniser le schéma BD avec le mapping des classes métier

```
using NHibernate.Cfg;
using NHibernate.Tool.hbm2ddl;
…
Configuration cfg = new Configuration();
cfg.Configure();
// Update database according to mapping files and DB credentials
new SchemaExport(cfg).Execute(true, true, false);
```

Supprime la BD existante

Requêtage

Le langage HQL

Hibernate Query Language

Fortement inspiré du SQL, mais utilise les classes et non les tables

```
// Liste des chambres de l'hôtel ayant l'id 1
string requete = "select r from Room r where r.Hotel.Id=?";
IList<Room> rooms = session.CreateQuery(requete).SetInt32(0,1).List<Room>();

// Liste des chats dont le poids > moyenne du poids des chats domestiques
requete = "from Cat as fatcat where fatcat.Weight > (
    select avg(cat.Weight) from DomesticCat cat
    )"
```


HQL – Fonctions d'agrégat

```
// Couleur, année de naissance et nombre de chats pour chaque couleur
var results = session.CreateQuery(
        "select cat.Color, min(cat.Birthdate), count(cat) from Cat cat " +
        "group by cat.Color")
    .Enumerable<object[]>();
foreach (object[] row in results)
{
    Color type = (Color) row[0];
    DateTime oldest = (DateTime) row[1];
    int count = (int) row[2];
    …
}
```

Linq

Language INtegrated Query

Standard .NET pour l'interrogation de données

```
// Liste des chats de couleur blanche
IList<Cat> cats = session.Query<Cat>().Where(c => c.Color == "white").ToList();

// Supprime les chats dont le poids > 20
session.Query<Cat>().Where(c => c.BodyWeight > 20).Delete();
```


Requêtes de sélection

```
// Liste de tous les hôtels
IList<Hotel> hotels = session.Query<Hotel>().ToList();

// Accès au premier hôtel de la liste
Hotel hotel = session.Query<Hotel>().SingleOrDefault<Hotel>();

// Nombre total de chambres
string requete = "select count(*) from Room";
int nbChambres = (int) session.CreateQuery(requete).UniqueResult<long>();

// Liste des chambres de l'hôtel ayant l'id 1
string requete = "select r from Room r where r.Hotel.Id=?";
IList<Room> rooms = session.CreateQuery(requete).SetInt32(0,1).List<Room>();
```

Mise à jour de données

```
Room room = new Room();
room.Hotel = session.Query<Hotel>().SingleOrDefault<Hotel>();
room.Number = 6667;
room.Floor = 4;
session.SaveOrUpdate(room); // Ajout d'une ligne dans la table Room de la BD (SQL INSERT)
session.Flush(); // Synchronisation des changements avec la BD

room.Empty = false;
session.SaveOrUpdate(room); // Mise à jour de la chambre dans la BD (SQL UPDATE)
session.Flush();

session.Delete(room); // Suppression de la chambre en BD (SQL DELETE)
session.Flush();
```