Formation Hibernate

Exercices

TP 1 : Un DAO en JDBC

Créez la classe Java nommée «Agent» de type Bean et disposant des attributs suivants :

- id (Long)
- prenom (String)
- nom (String)

Cette classe (entité) va matérialiser un agent de police.

Utilisez le pattern DAO pour effectuer les opérations de CRUD sur l'objet Agent en JDBC

Aide

JDBC : http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html

TP 2 : Mise en place d'Hibernate

A partir du TP 1 , installez et configurez Hibernate dans votre application :

- Mappez la classe Java nommée «Agent» vue à l'exercice précédent.
- Videz entièrement la base de données et configurez l'option hbm2ddl pour qu'Hibernate vous génère la table « agents ».

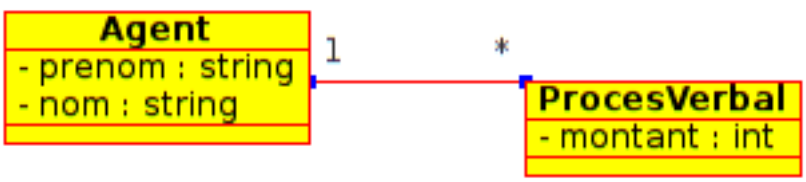
Pour l'instant aucune donnée n'est à lire ou écrire en base.

TP 3 : Implémentation Hibernate du DAO

A partir du TP 2, écrivez une nouvelle implémentation complète du DAO responsable des objets Agent, cette implémentation utilisera Hibernate.

TP 4 : Mapping de collection d'un type de base

Mettez en place le modèle objet métier tel que décrit dans le diagramme de classe ci-dessous :

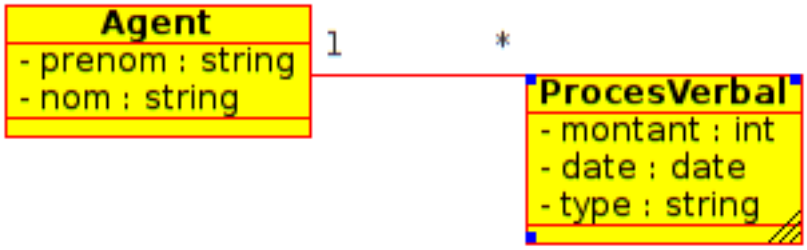


La classe Agent devra alors contenir un attribut appelé procesVerbaux qui est une implémenation de java.util.Set.

Votre application devra permettre de créer et lister des agents de police. Chaque agent disposera de la liste des PVs qu'il a établi sous la forme d'une simple collection du montant de l'amende.

TP 5 : Mapping de collection d'un objet composant

A partir du TP 4, modifiez votre application pour que « ProcesVerbal » devienne un objet Java tel que décrit dans le diagramme ci-dessous :



- Crééz au moins 2 agents disposant chacun d'au moins 2 procès verbaux à leur actif.
- L'un d'eux part en retraite, supprimez le de l'application en passant par la méthode delete() de Session et observez les conséquences.

TP 6 : Mapping d'associations unidirectionnelles

1ère partie

A partir du TP 5, faîtes en sorte qu'un agent puisse référencer une liste de procès verbaux (ProcesVerbal devient une Entity). ATTENTION : A ce stade on ne peut pas utiliser la contrainte « not-null » sur <key>.

2e partie

Désactivez l'association d'Agent à ProcesVerbal que vous venez de mettre en place en 1ère partie de TP.

Mappez alors l'association inverse de ProcesVerbal vers Agent.

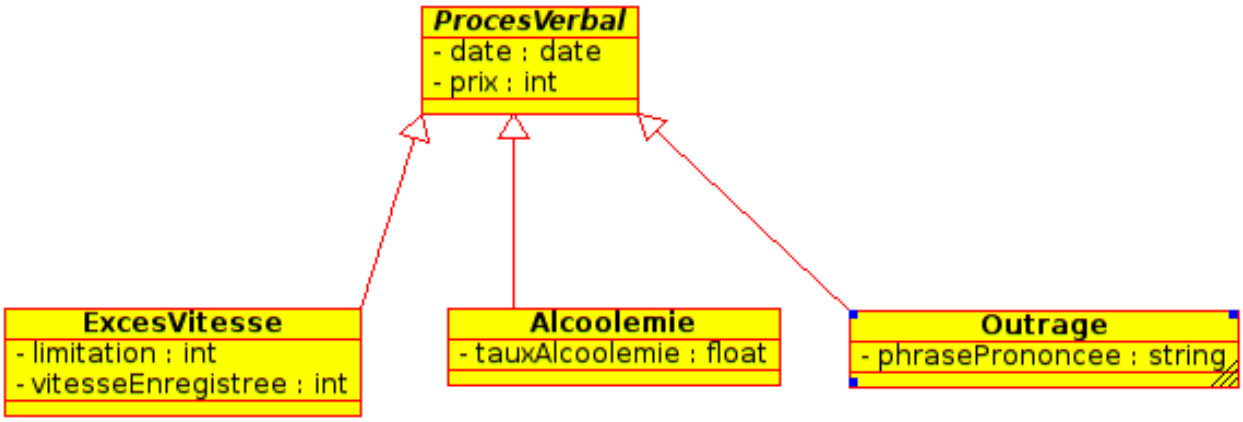
TP 7 : Mapping d'associations bidirectionnelles

A partir du TP 6, faîtes en sorte qu'un agent puisse référencer une liste de procès verbaux et qu'un procès verbal pointe en retour sur un agent.

Observez les effets de la double requête de mise à jour et corrigez le problème.

TP 8 : Mapping de l'héritage

A partir du TP 7, modifiez ProcesVerbal pour qu'il corresponde à la hiérarchie d'héritage décrite dans le diagramme ci-dessous :



Utilisez un mapping d'héritage de type une table par hiérarchie de classes.

Votre application devra permettre de créer un ProcesVerbal de chaque type ainsi que de récupérer une instance de ProcesVerbal sans savoir à l'avance de quelle classe concrète il s'agit.

Modifiez votre application en utilisant cette fois un mapping d'héritage de type une table par classe.

TP 9 : Sessions contextuelles

Modifiez votre application pour qu'elle supporte les sessions contextuelles de type « ThreadLocal ». Créez une couche de service au besoin.

TP 10 : Requêtes HQL

Ecrivez une méthode de DAO capable de rechercher la liste des instances de ProcesVerbal dont la somme est supérieure ou égale à une somme passée en paramètre, le tout entre deux dates données elles aussi passées en paramètres : getProcesVerbalList(minPrix,minDate,maxDate)

Ecrivez une méthode de DAO capable de rechercher la moyenne du prix des instances de ProcesVerbal entre deux dates données passées en paramètres : getProcesVerbalAveragePrice(minDate,maxDate)

TP 11 : Requêtes Criteria

Reprenez le TP 10 :

Dans votre DAO en remplacez les requêtes précédement écrite en HQL par leur équivalent criteria.

Considérez qu'aucun des paramètres des méthodes du DAO ne sont obligatoires. S'il sont tous renseignés à null, c'est la liste complète qui doit être renvoyée en tant que réponse.

TP 12 : Persistance transitive

Faîtes en sorte que la sauvegarde, mise à jour ou suppression d'un agent entraîne automatiquement une sauvegarde, mise à jour ou suppression complète des procès verbaux qui lui sont rattachés.

Observez l'effet de l'enregistrement « orphelin » en supprimant un élément de la liste des procès verbaux d'un agent.

TP 13 : Traitement par lots

Rajoutez un attribut « payé » de type booléen sur la classe « ProcesVerbal ». Faîtes en sorte d'avoir des procès verbaux dans un état « payé=true » et d'autres dans un état « payé=false ». Créez une méthode de DAO appelée « amnistie » qui supprime en une seule opération directement en base l'ensemble des procès verbaux non-payés.

TP 14 : Stratégies de chargement

Mettez en œuvre la stratégie de chargement de type batch fetching afin d'optimiser le chargement d'un agent depuis une liste de proces verbaux (many-to-one).

Vérifiez les impacts de ces stratégies en affichant dans les journaux de l'application les requêtes SQL envoyées à la base (propriété hibernate.show_sql=true).

Modifiez votre stratégie de chargement afin d' utiliser le join fetching et mesurez les impacts toujours en terme d'appels à la base.

TP 15 : Cache de second niveau

Mettez en place un cache de second niveau à l'aide de l'implémentation EhCache. Les instances d'Agent doivent pouvoir être mises en cache. Activez puis désactivez le cache vous devriez observer un changement dans le nombre de requêtes SQL envoyées à la base pour les mêmes opérations (propriété hibernate.show_sql=true).

TP 16 : Query Cache

Mettez en place le Query cache afin que 2 récupérations successives de la liste des agents (au travers d'une requête HQL ou Criteria) n'envoie qu'une seule requête à la base.