Initiation aux frameworks : *Hibernate*

Hibernate : utilisations avancées

Thomas Duchatelle (duchatelle.thomas@gmail.com)

Capgemini, pour Yves Rocher

January 3, 2013

Sommaire

Comment dégrader les performances avec Hibernate ?

Plus facile qu'il n'y parait !

Les opérations secrètes

Hibernate réalise beaucoup d'opérations de façon transparente pour nous simplifier la vie. Est-ce que cela nous la simplifie vraiment ?

Comment dégrader les performances avec Hibernate ?

Plus facile qu'il n'y parait !

Les opérations secrètes

Hibernate réalise beaucoup d'opérations de façon transparente pour nous simplifier la vie. Est-ce que cela nous la simplifie vraiment ?

Quelques idées pour dégrader les performances :

- Ne jamais fermer la session : plus il y a d'objets dans la session, plus le `flush` sera long.

Comment dégrader les performances avec Hibernate ?

Plus facile qu'il n'y parait !

Les opérations secrètes

Hibernate réalise beaucoup d'opérations de façon transparente pour nous simplifier la vie. Est-ce que cela nous la simplifie vraiment ?

Quelques idées pour dégrader les performances :

- Ne jamais fermer la session : plus il y a d'objets dans la session, plus le `flush` sera long.
- Charger toutes les dépendances d'un objet à chaque fois, même si on en a pas besoin

Comment dégrader les performances avec Hibernate ?

Plus facile qu'il n'y parait !

Les opérations secrètes

Hibernate réalise beaucoup d'opérations de façon transparente pour nous simplifier la vie. Est-ce que cela nous la simplifie vraiment ?

Quelques idées pour dégrader les performances :

- Ne jamais fermer la session : plus il y a d'objets dans la session, plus le `flush` sera long.
- Charger toutes les dépendances d'un objet à chaque fois, même si on en a pas besoin
- Ne pas vérifier les `select` générés : les jointures d'Hibernate sont idéales

Sommaire

Les requêtes SELECT

HQL : ami ou ennemi

Le langage HQL facilite l'écriture des requêtes SQL en éliminant la partie technique, et en axant la requête sur un point de vue *objets*.

Les requêtes SELECT

HQL : ami ou ennemi

Le langage HQL facilite l'écriture des requêtes SQL en éliminant la partie technique, et en axant la requête sur un point de vue *objets*.

Jointures

Mais il ne remplace pas la connaissance SQL ! La facilité d'écriture du HQL cache certaines jointures, il ne faut pas les ignorer.

Les requêtes SELECT

HQL : ami ou ennemi

Le langage HQL facilite l'écriture des requêtes SQL en éliminant la partie technique, et en axant la requête sur un point de vue *objets*.

Jointures

Mais il ne remplace pas la connaissance SQL ! La facilité d'écriture du HQL cache certaines jointures, il ne faut pas les ignorer.

Optimisations

Une requête HQL s'optimise, comme une requête SQL.

Chargement des dépendances

Chargement feignant (LAZY)

LAZY

Lorsque le chargement dit `LAZY` est activé sur une relation, la dépendance n'est chargée (requête en base) que lors de l'accès à la méthode `get`.

[1] persistant = rattaché à une session *non fermée*

Chargement des dépendances
Chargement feignant (LAZY)

LAZY

Lorsque le chargement dit `LAZY` est activé sur une relation, la dépendance n'est chargée (requête en base) que lors de l'accès à la méthode `get`.

- le mode est actif par défaut sur toutes les relations
- son contraire est `EAGER`
- l'objet doit être *persistant*[1] lors du premier accès à la méthode `getX`

[1] persistant = rattaché à une session *non fermée*

Chargement des dépendances
Chargement feignant (LAZY)

LAZY

Lorsque le chargement dit `LAZY` est activé sur une relation, la dépendance n'est chargée (requête en base) que lors de l'accès à la méthode `get`.

- le mode est actif par défaut sur toutes les relations
- son contraire est `EAGER`
- l'objet doit être *persistant*[1] lors du premier accès à la méthode `getX`

Initialisation d'une entité

L'initialisation des attributs et collections peut être forcé en appelant la méthode `Hibernate.initialize(entity.getX());`

[1] persistant = rattaché à une session *non fermée*

Grouper les requêtes

Charger les dépendances en une requête

Cas d'exemple

Chargement d'une liste d'employés dont on souhaite, entre autre, connaitre l'entreprise qui l'emploie.

Grouper les requêtes

Charger les dépendances en une requête

Cas d'exemple

Chargement d'une liste d'employés dont on souhaite, entre autre, connaitre l'entreprise qui l'emploie.

Nombre de requêtes `select` en base :

- 1 sur la table `employee`
- puis 1 par employé sur la table entreprise !

Grouper les requêtes

Forcer la jointure

Par requête

L'idée d'optimisation est de forcer la jointure entre l'employé, et l'entreprise :

```
1  SELECT e FROM Employee e OUTER JOIN FETCH e.enterprise
```

Grouper les requêtes
Forcer la jointure

Par requête

L'idée d'optimisation est de forcer la jointure entre l'employé, et l'entreprise :

```
1   SELECT e FROM Employee e OUTER JOIN FETCH e.enterprise
```

Par mapping

L'idée d'optimisation est de forcer la jointure entre l'employé, et l'entreprise :

```
1   @Fetch(FetchMode.JOIN)
2   public Enterprise getEnterprise() { ... }
```

Grouper les requêtes

Alternatives...

BatchSize

Charger simultanément les dépendances de `n` attribut avec l'annotation `@BatchSize`.

```
1  @BatchSize(size=20)
2  public class Employee ... {
3    ...
4
5    @BatchSize(size=20)
6    public Enterprise getEnterprise() { ... }
7  }
```

Les cascades

Ici le Niagara

Cascade

Le paramètre cascade est présent sur toutes les associations. Il définie le comportement de la session vis à vis des dépendances.

Les cascades

Ici le Niagara

Cascade

Le paramètre cascade est présent sur toutes les associations. Il définie le comportement de la session vis à vis des dépendances.

Les principales valeurs possibles :

- `PERSIST` : les dépendances sont sauvegardées si l'entité l'est
- `REMOVE` : les dépendances sont supprimées avec l'entité
- `ALL` : toutes les actions sur l'entité sont répercutées sur les dépendances.

Fin

Merci, des questions ?