**Hibernate mutable example (class and collection)**

In hibernate, ‘**mutable**‘ is default to ‘true’ in class and its related collection, it mean the class or collection are allow to add, update and delete. On the other hand, if the mutable is changed to false, it has different meaning in class and its related collection. Let’s take some examples to understand more about it.

**Hibernate one-to-many example**

I will take this [one-to-many example](http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/) for the mutable demonstration. In this mapping file, a Stock is belong to many StockDailyRecord.

<!-- Stock.hbm.xml -->

<hibernate-mapping>

<class name=*"Stock"* table=*"stock"*>

<set name=*"stockDailyRecords"* mutable=*"false"* cascade=*"all"*

inverse=*"true"* lazy=*"true"* table=*"stock\_daily\_record"*>

<key>

<column name=*"STOCK\_ID"* not-null=*"true"* />

</key>

<one-to-many class=*"StockDailyRecord"* />

</set>

</class>

...

</hibernate-mapping>

**How to declare mutable ?**

The ‘mutable’ is support both in XML mapping file and annotation.

**1. XML mapping file**

In mapping file, the ‘**mutable**‘ keyword is use to implement the mutable function.

<!-- Stock.hbm.xml -->

...

<hibernate-mapping>

<class name=*"Stock"* table=*"stock"* mutable=*"false"*>

<set name=*"stockDailyRecords"* mutable=*"false"* cascade=*"all"*

inverse=*"true"* lazy=*"true"* table=*"stock\_daily\_record"*>

<key>

<column name=*"STOCK\_ID"* not-null=*"true"* />

</key>

<one-to-many class=*"StockDailyRecord"* />

</set>

</class>

...

</hibernate-mapping>

**2. Annotation**

In annotation, the keyword is changed to @Immutable (mutable=’false’).

@Entity

@Immutable

@Table(name = "stock")

**public** **class** Stock **implements** java.io.Serializable {

...

@OneToMany(fetch = FetchType.LAZY, mappedBy = "stock")

@Immutable

**public** Set<StockDailyRecord> getStockDailyRecords() {

**return** **this**.stockDailyRecords;

}

## Mutable in class

If mutable = “false” or @Immutable is declared in class element, it means the **updates to this class will be ignored, but no exception is thrown, only the add and delete operation are allow**.

## 1. Test insert

Stock stock = **new** Stock();

stock.setStockCode("7277");

stock.setStockName("DIALOG");

session.save(stock);

if mutable = “true” (default) or no @Immutable is declared in class.  
Output

Hibernate:

insert into stock (STOCK\_CODE, STOCK\_NAME) values (?, ?)

if mutable = “false” or @Immutable is declared in class.  
Output

Hibernate:

insert into stock (STOCK\_CODE, STOCK\_NAME)

values (?, ?)

**Mutable in class has no effect in the ‘insert’ operation.**

**2. Test update**

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

stock.setStockName("DIALOG123");

session.saveOrUpdate(stock);

if mutable = “true” or no @Immutable is declared in class.  
Output

Hibernate:

select ...from mkyong.stock stock0\_

where stock0\_.STOCK\_CODE='7277'

Hibernate:

update stock

set STOCK\_CODE=?,STOCK\_NAME=?

where STOCK\_ID=?

if mutable = “false” or @Immutable is declared in class.  
Output

Hibernate:

select ...from stock stock0\_

where stock0\_.STOCK\_CODE='7277'

**Mutable in class is not allow application to update it, the ‘update’ operation will be ignore and no exception is thrown**

**3. Test delete**

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

session.delete(stock);

if mutable = “true” (default) or no @Immutable is declared in class.  
Output

Hibernate: delete from stock where STOCK\_ID=?

if mutable = “false” or @Immutable is declared in class.  
Output

Hibernate: delete from stock where STOCK\_ID=?

**Mutable in class has no effect in the ‘delete’ operation.**

**Mutable in collection**

If mutable = “false” or @Immutable is declared in collection, it means the **add and delete-orphan are not allow in this collection, with exception throw, only update and ‘cascade delete all’ are allow**.

**1. Test insert**

Assume the cascade insert is enabled.

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

StockDailyRecord sdr = **new** StockDailyRecord();

sdr.setDate(**new** Date());

sdr.setStock(stock);

stock.getStockDailyRecords().add(sdr);

session.save(stock);

if mutable = “true” (default) or no @Immutable is declared in collection.  
Output

Hibernate:

insert into stock\_daily\_record

(STOCK\_ID, PRICE\_OPEN, PRICE\_CLOSE, PRICE\_CHANGE, VOLUME, DATE)

values (?, ?, ?, ?, ?, ?)

if mutable = “false” or @Immutable is declared in collection.  
Output

Exception in thread "main" org.hibernate.HibernateException:

changed an immutable collection instance:

[Stock.stockDailyRecords#111]

**Mutable in collection is not allow the ‘add’ operation, an exception will throw.**

**2. Test update**

Assume the cascade update is enabled.

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

StockDailyRecord sdr = stock.getStockDailyRecords().iterator().next();

sdr.setPriceChange(**new** Float(1.30));

session.saveOrUpdate(stock);

if mutable = “true” (default) or no @Immutable is declared in collection.  
Output

Hibernate: update stock\_daily\_record set PRICE\_CHANGE=?, ...where DAILY\_RECORD\_ID=?

if mutable = “false” or @Immutable is declared in collection.  
Output

Hibernate: update stock\_daily\_record set PRICE\_CHANGE=?, ...where DAILY\_RECORD\_ID=?

**Mutable in collection has no effect in the ‘update’ operation.**

**3. Test delete-orphan**

Assume the [cascade delete-orphan](http://www.mkyong.com/hibernate/hibernate-cascade-example-save-update-delete-and-delete-orphan/) is enabled.

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

StockDailyRecord sdr = stock.getStockDailyRecords().iterator().next();

stock.getStockDailyRecords().remove(sdr);

session.saveOrUpdate(stock);

if mutable = “true” (default) or no @Immutable is declared in collection.  
Output

Hibernate: delete from stock\_daily\_record where DAILY\_RECORD\_ID=?

if mutable = “false” or @Immutable is declared in collection.  
Output

Exception in thread "main" org.hibernate.HibernateException:

changed an immutable collection instance:

[Stock.stockDailyRecords#111]

**Mutable in collection is not allow the ‘delete-orphan’ operation, an exception will throw.**

**4. Test delete**

Assume the cascade delete is enabled.

Stock stock = (Stock)session.createQuery(

" from Stock where stockCode = '7277'").list().get(0);

session.saveOrUpdate(stock);

if mutable = “true” (default) or no @Immutable is declared in collection.  
Output

Hibernate: delete from stock\_daily\_record where DAILY\_RECORD\_ID=?

Hibernate: delete from stock where STOCK\_ID=?

if mutable = “false” or @Immutable is declared in collection.  
Output

Hibernate: delete from stock\_daily\_record where DAILY\_RECORD\_ID=?

Hibernate: delete from stock where STOCK\_ID=?

**Mutable in collection has no effect in the ‘delete’ operation, if parent is deleted, all its child will be delete as well, even it’s mutable.**

**Why mutable ?**

Mutable can avoid many unintentional database operation, like add, update or delete some records which shouldn’t be. In addition, according to Hibernate documentation, the mutable do has some minor performance optimizations, it’s always recommend to analysis your mapping relationship and implement the mutable as needed.

**Summary**

##### **1. mutable = “false” or @Immutable is declared in class**

it means the updates to this class will be ignored, but no exception is thrown, only the add and delete operation are allow.

In Class with mutable=”false” – insert=allow, delete=allow , update=not allow

##### **2. mutable = “false” or @Immutable is declared in collection**

it means the add and delete-orphan are not allow in this collection, with exception throw, only update allow. However, if cascade delete is enable, when the parent is deleted, all it’s child will be delete as well, even it is mutable.

In Collection with mutable=”false” – insert=not allow, delete-orphan=not allow, delete=allow , update=allow

**Completely immutable ?**

Can a class completely immutable to any actions? Yes, put a mutable=”false” to all it’s relationship (insert=not allow, delete-orphan=not allow), and a mutable=”false” to the class you want to immutable (update=not allow). Now, you have a completely immutable class, however, if the cascade delete option is enabled, when the parent of your immutable class is deleted, your immutable class will still be deleted as well.