Home All Tutorials Java Core JSF Spring Hibernate Struts Android Others

Hibernate – Cascade Example (Save, Update, Delete And Delete-Orphan)



Posted on February 2, 2010 , Last modified : August 30, 2012

By mkyong

Cascade is a convenient feature to save the lines of code needed to manage the state of the other side manually.

The "Cascade" keyword is often appear on the collection mapping to manage the state of the collection automatically. In this tutorials, this <u>one-to-many example</u> will be used to demonstrate the cascade effect.

Cascade save / update example

In this example, if a 'Stock' is saved, all its referenced 'stockDailyRecords' should be saved into database as well.

1. No save-update cascade

In <u>previous section</u>, if you want to save the 'Stock' and its referenced 'StockDailyRecord' into database, you need to save both individually.

```
Stock stock = new Stock();
StockDailyRecord stockDailyRecords = new StockDailyRecord();
//set the stock and stockDailyRecords data
stockDailyRecords.setStock(stock);
stock.getStockDailyRecords().add(stockDailyRecords);
session.save(stockDailyRecords);
```

Output

```
Hibernate:
    insert into mkyong.stock (STOCK_CODE, STOCK_NAME)
    values (?, ?)

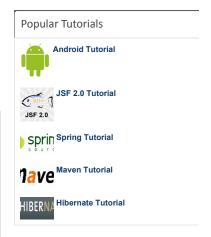
Hibernate:
    insert into mkyong.stock_daily_record
    (STOCK_ID, PRICE_OPEN, PRICE_CLOSE, PRICE_CHANGE, VOLUME, DATE)
    values (?, ?, ?, ?, ?, ?)
```

2. With save-update cascade

The cascade="save-update" is declared in 'stockDailyRecords' to enable the save-update cascade effect.

```
Stock stock = new Stock();
StockDailyRecord stockDailyRecords = new StockDailyRecord();
//set the stock and stockDailyRecords data
```





```
stockDailyRecords.setStock(stock);
stock.getStockDailyRecords().add(stockDailyRecords);
session.save(stock);
```

Output

```
Hibernate:
    insert into mkyong.stock (STOCK_CODE, STOCK_NAME)
    values (?, ?)

Hibernate:
    insert into mkyong.stock_daily_record
    (STOCK_ID, PRICE_OPEN, PRICE_CLOSE, PRICE_CHANGE, VOLUME, DATE)
    values (?, ?, ?, ?, ?, ?)
```

The code **session.save(stockDailyRecords)**; is no longer required, when you save the 'Stock', it will "cascade" the save operation to it's referenced 'stockDailyRecords' and save both into database automatically.

Cascade delete example

In this example, if a 'Stock' is deleted, all its referenced 'stockDailyRecords' should be deleted from database as well.

1. No delete cascade

You need to loop all the 'stockDailyRecords' and delete it one by one.

Output

```
Hibernate:

delete from mkyong.stock_daily_record
where DAILY_RECORD_ID=?

Hibernate:

delete from mkyong.stock
where STOCK_ID=?
```

2. With delete cascade

The **cascade="delete"** is declared in 'stockDailyRecords' to enable the delete cascade effect. When you delete the 'Stock', all its reference 'stockDailyRecords' will be deleted automatically.

```
Query q = session.createQuery("from Stock where stockCode = :stockCode ");
q.setParameter("stockCode", "4715");
Stock stock = (Stock)q.list().get(0);
session.delete(stock);
```

Output

```
Hibernate:

delete from mkyong.stock_daily_record
```

```
where DAILY_RECORD_ID=?

Hibernate:
    delete from mkyong.stock
    where STOCK_ID=?
```

Cascade delete-orphan example

In above cascade delete option, if you delete a Stock, all its referenced 'stockDailyRecords' will be deleted from database as well. How about if you just want to delete two referenced 'stockDailyRecords' records? This is called orphan delete, see example...

1. No delete-orphan cascade

You need to delete the 'stockDailyRecords' one by one.

Output

```
Hibernate:

delete from mkyong.stock_daily_record

where DAILY_RECORD_ID=?

Hibernate:

delete from mkyong.stock_daily_record

where DAILY_RECORD_ID=?
```

2. With delete-orphan cascade

The **cascade="delete-orphan"** is declared in 'stockDailyRecords' to enable the delete orphan cascade effect. When you save or update the Stock, it will remove those 'stockDailyRecords' which already mark as removed.

Output

```
Hibernate:

delete from mkyong.stock_daily_record

where DAILY_RECORD_ID=?

Hibernate:

delete from mkyong.stock_daily_record

where DAILY_RECORD_ID=?
```

In short, delete-orphan allow parent table to delete few records (delete orphan) in its child table.

How to enable cascade?

The cascade is supported in both XML mapping file and annotation.

1. XML mapping file

In XML mapping file, declared the cascade keyword in your relationship variable.

2. Annotation

In annotation, declared the CascadeType.SAVE_UPDATE (save, update) and CascadeType.REMOVE (delete) in @Cascade annotation.

```
//Stock.java
@OneToMany(mappedBy = "stock")
@Cascade({CascadeType.SAVE_UPDATE, CascadeType.DELETE})
public Set<StockDailyRecord> getStockDailyRecords() {
    return this.stockDailyRecords;
}
```

Further study - Cascade - JPA & Hibernate annotation common mistake.

inverse vs cascade

Both are totally different notions, see the <u>differential here</u>

Conclusion

Cascade is a very convenient feature to manage the state of the other side automatically. However this feature come with a price, if you do not use it wisely (update or delete), it will generate many unnecessary cascade effects (cascade update) to slow down your performance, or delete (cascade delete) some data you didn't expected.

Tags: cascade

hibernate



mkyong

Founder of Mkyong.com and HostingCompass.com, love Java and open source stuff. Follow him on Twitter, or befriend him on Facebook or Google Plus. If you like my tutorials, consider making a donation to this charity, thanks.

Related Posts

- ► <u>Cascade JPA & Hibernate annotation common</u> <u>mistake</u>
- ▶ <u>Different between cascade and inverse</u>
- ▶ Spring Security + Hibernate XML Example
- java.lang.ClassNotFoundException: org.hibernate.service.ita.platform.spi.JtaPlatform
- ▶ Hibernate component mapping example

Popular Posts

- ▶ Top 8 Java People You Should Know
- ► Top 20 Java Websites
- ► Top 5 Free Java eBooks
- ▶ Top 10 Java Regular Expression Examples
- ▶ Top 5 Open Source Q&A Systems

You might also like following tutorials :