Hibernate Tip: Use a timestamp for versioning and optimistic locking

Question:

My table model uses a timestamp instead of a numeric column for versioning. How can I use this column for Hibernates optimistic locking mechanism?

Solution:

The JPA specification supports numeric and timestamp columns for versioning. You can use the timestamp column in the same way as a numeric version column. You just need an entity attribute of *java.util.Date* and annotate it with *@Version*.

```
@Entity
public class Author {
    @Version
    private Date version;
    ...
}
```

Hibernate will retrieve the current time from the local JVM and use it to update the database column for each create or update operation.

```
13:44:49,494 DEBUG [org.hibernate.SQL] - select nextval
('hibernate_sequence')
13:44:49,551 DEBUG [org.hibernate.SQL] - insert into Author
(dateOfBirth, firstName, lastName, version, id) values (?, ?,
?, ?, ?)
```

```
13:44:49,557 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [1] as [DATE] - [null]
13:44:49,558 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [2] as [VARCHAR] - [Thorben]
13:44:49,558 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [3] as [VARCHAR] - [Janssen]
13:44:49,559 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [4] as [TIMESTAMP] - [2017-08-07 13:44:49.519]
13:44:49,561 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [5] as [BIGINT] - [1]
```

But please be aware that this approach has a few drawbacks:

1. The JVM usually doesn't provide the timestamp with millisecond accuracy.
2. Hibernate can't detect 2 concurrent updates that are executed at the same millisecond.
3. If you scale your application horizontally, you need to keep the timestamp of all instances in sync.

Get current time from the database

You can avoid drawback 2 and 3 by retrieving the version timestamp from your database. That is a Hibernate-specific feature that is not supported by all Hibernate *Dialect*s and requires an additional database query. So, you shouldn't use it, if you need to optimize your application for performance.

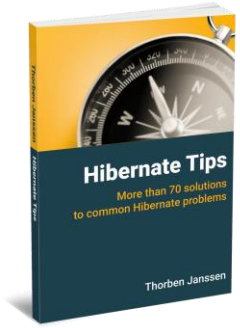
If you want to use this feature, you need to annotate your version attribute with an additional *@Type* annotation and set its *type* attribute to *dbtimestamp*

```
@Entity
public class Author {

    @Version
    @Type(type = "dbtimestamp")
    private Date version;

    ...
}
```

As you can see in the log output, Hibernate now performs an additional query to retrieve the current time from the database.

```
13:45:54,997 DEBUG [org.hibernate.SQL] - select nextval
('hibernate_sequence')
13:45:55,019 DEBUG [org.hibernate.SQL] - select now()
13:45:55,052 DEBUG [org.hibernate.SQL] - insert into Author
(dateOfBirth, firstName, lastName, version, id) values (?, ?,
?, ?, ?)
13:45:55,058 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [1] as [DATE] - [null]
13:45:55,059 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [2] as [VARCHAR] - [Thorben]
13:45:55,059 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [3] as [VARCHAR] - [Janssen]
13:45:55,060 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [4] as [TIMESTAMP] - [2017-08-07 13:45:54.999]
13:45:55,062 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [5] as [BIGINT] - [1]
```

Hibernate Tips Book



Get more recipes like this one in my book Hibernate Tips: More than 70 solutions to common Hibernate problems.

It gives you more than 70 ready-to-use recipes for topics like basic and advanced mappings, logging, Java 8 support, caching and statically and dynamically defined queries.