Hibernate Tip: Count the executed queries in a Session

Question:

Some of my use cases are slow and seem to perform too many queries. How can I count all queries executed within a Hibernate Session?

Solution:

The easiest way to count all executed queries is to activate Hibernate's statistics component. Hibernate then collects a lot of internal statistics and provides them as a log message and via the Statistics API. Collecting all these information takes some time, so do not use this in production!

Hibernate's statistics component is deactivated by default. You can activate it by setting the *hibernate.generate_statistics* parameter to true. You can either do this by providing a system property with the same name or by setting the parameter in the persistence.xml file.

Hibernate Tip: Count the executed queries in a Session

```
<persistence>
 <persistence-unit name="my-persistence-unit">
 <description>Hibernate Tips</description>
<provider>org.hibernate.jpa.HibernatePersistenceProvider</
provider>

 <properties>
 <property name="hibernate.generate_statistics" value="true"
/>
        …
 </properties>
 </persistence-unit>
</persistence>
```

You have 2 options to access the statistics. Hibernate can write a subset with the most important information of each session to the log file or you can access them via the Statistics API.

Let's have a look at the log messages first.

Hibernate writes a log message, similar the following one, at the end of each session. You can find the number of SQL statements and the time spent for their preparation and execution in line 4 and 5.

Hibernate Tip: Count the executed queries in a Session

```
16:24:55,318 INFO
[org.hibernate.engine.internal.StatisticalLoggingSession
EventListener] – Session Metrics {

25659 nanoseconds spent acquiring 1 JDBC
connections;

22394 nanoseconds spent releasing 1 JDBC
connections;

1091216 nanoseconds spent preparing 12 JDBC
statements;

11118842 nanoseconds spent executing 12 JDBC
statements;

0 nanoseconds spent executing 0 JDBC batches;

0 nanoseconds spent performing 0 L2C puts;

0 nanoseconds spent performing 0 L2C hits;

0 nanoseconds spent performing 0 L2C misses;

16999942 nanoseconds spent executing 1 flushes
(flushing a total of 17 entities and 17 collections);

63915 nanoseconds spent executing 1 partial-flushes
(flushing a total of 0 entities and 0 collections)

}
```

You can access the Statistics API via Hibernate's *SessionFactory*. It provides a lot more information than the log output.

```
Statistics stats = sessionFactory.getStatistics();

long queryCount = stats.getQueryExecutionCount();
```

Learn more:

If you want to learn more about finding and fixing Hibernate performance issue, you should have a look at the following posts:

- How to activate Hibernate Statistics to analyze performance issues
- Hibernate Logging Guide – Use the right config for development and production
- 7 Tips to boost your Hibernate performance
- Free Mini Course: How to find and fix n+1 select issues with Hibernate