Hibernate Tip: Escape table and column names

Question:

One of my entities represents an order which a customer placed in my online store. I would like to call that entity *Order*, but Hibernate generates SQL statements containing syntax errors and throws *SQLGrammarException*s. How can I fix that?

Solution:

By default, Hibernate maps an entity to a database table with the same name. So, in this case, Hibernate tries to map the *Order* entity to the *Order* table. But *Order* is a reserved word in SQL and can't be used as a database identifier. You either need to choose a different table name or use a delimited identifier.

The definition of a delimited identifier is pretty simple. It's defined in section 2.13 of the JPA specification. You just need to set the table name with a *@Table* annotation and enclose it in double quotes.

```
@Entity
@Table(name = "\"Order\"")
public class Order { ... }
```

You can not only use this approach with the *@Table* annotation. It's also supported by all other JPA annotations that allow you to define a database identifier. These are:

- *@EntityResult*
- *@FieldResult*
- *@ColumnResult*
- *@CollectionTable*
- *@Column*
- *@DiscriminatorColumn*

- *@ForeignKey*

- *@Index*

- *@JoinColumn*

- *@JoinTable*

- *@MapKeyColumn*

- *@MapKeyJoinColumn*

- *@NamedStoredProcedureQuery*

- *@OrderColumn*

- *@PrimaryKeyJoinColumn*

- *@SecondaryTable*

- *@SequenceGenerator*

- *@StoredProcedureParameter*

- *@Table*

- *@TableGenerator*

- *@UniqueConstraint*

When you specify the table name like this, Hibernate and all other JPA implementations will use the defined delimited identifier in all generated SQL statements.
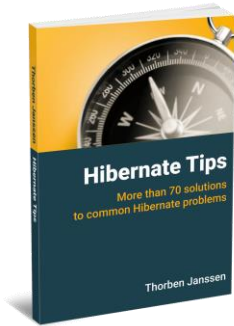
```
17:13:21,070 DEBUG [org.hibernate.SQL] - insert into
"Order" (orderNumber, version, id) values (?, ?, ?)
```

Learn more

If you like to learn more about JPA's *@Table* annotation, you should take a look at this Hibernate Tip: [How to define schema and table names](#).

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.