Hibernate Tip: Map a *java.util.Date* to a column

Question:

I use a *java.util.Date* to persist a date as an entity attribute.

But Hibernate maps it to a timestamp with nanoseconds. How can I change the mapping so that Hibernate only stores the years, months and days?

Solution:

The SQL standard supports three different data types to store date and time information. Hibernate can map all of them to a *java.util.Date* or a *java.util.Calendar*. You need to decide which of the following SQL types Hibernate shall use:

- *TIMESTAMP*: Persists the date and time with nanoseconds. Hibernate uses this type by default.
- *TIME*: Stores only the time of day without nanoseconds.
- *DATE*: Persists only the date with years, months and days.

You can define the preferred mapping with the *@Temporal* annotation. As you can see in the following code snippet, it takes a *TemporalType* enum as a value. The enum allows you to select the SQL type (*DATE*, *TIME* or *TIMESTAMP*) which you want to use.

Hibernate Tip: Map a *java.util.Date* to a column

```
@Entity
public class Author {

    @Temporal(TemporalType.DATE)
    private Date dateOfBirth;
    ...
}
```

As you can see in the following log output, the *dateOfBirth* attribute of the *Author* entity gets mapped to an SQL *DATE* without any time information.

```
07:22:50,454 TRACE
[org.hibernate.type.descriptor.sql.BasicBinder] - binding
parameter [1] as [BIGINT] - [1]

07:22:50,464 TRACE
[org.hibernate.type.descriptor.sql.BasicExtractor] - extracted
value ([dateOfBi2_0_0_] : [DATE]) - [1980-01-01]

07:22:50,465 TRACE
[org.hibernate.type.descriptor.sql.BasicExtractor] - extracted
value ([firstNam3_0_0_] : [VARCHAR]) - [John]

07:22:50,465 TRACE
[org.hibernate.type.descriptor.sql.BasicExtractor] - extracted
value ([lastName4_0_0_] : [VARCHAR]) - [Doe]

07:22:50,466 TRACE
[org.hibernate.type.descriptor.sql.BasicExtractor] - extracted
value ([version5_0_0_] : [INTEGER]) - [0]
```

Learn more:

Since Hibernate 5, you can also use the classes Java 8's Date and Time API as entity attribute types. The new classes solve a lot of issues of the *java.util.Date* and provide all information Hibernate needs to map them to the correct JDBC types.

I explain the mapping of the Date and Time API classes in more details in: Hibernate 5: how to persist LocalDateTime and Co with Hibernate.