JPA Tip: Map a Duration attribute

Question:

JPA 2.2 didn't add support for *java.time.Duration*. How can I map an attribute of that type with JPA?

Solution:

Unfortunately, JPA 2.2 only supports some of the classes of the Date and Time API and *java.time.Duration* isn't one of them. If you are limited to plain JPA, you need to implement a custom mapping for attributes of type *Duration*. As you will see, that is not as complicated as it might seem.

Hibernate makes it easy

But before we dive into the details, I want to show you a more comfortable approach. Since version 5, Hibernate supports *java.time.Duration* as a basic type. So, if you are allowed to use proprietary Hibernate features, you can use entity attributes of type *Duration* without any additional mapping annotations.

```
@Entity
public class OnlineCourse {

    @Id
    @GeneratedValue
    private Long id;

    private String title;

    private Duration videoDuration;

    ...
}
```

JPA Tip: Map a Duration attribute

JPA requires a little bit of work

If you don't use Hibernate or if some internal regulations prevent you from using proprietary features, you can implement a custom mapping with a simple *AttributeConverter*. Within this converter, you need to map the unsupported *Duration* object to an object of a supported type.

You can, for example, convert the *Duration* object to a *long* which represents the number of nanoseconds of the duration. Just be aware that this limits your *Duration* to a little bit more than 292 years. That should be enough for most applications. If you need to store a longer duration, you will need to reduce the precision, e.g., persist the number of milliseconds.

The implementation of such a converter is relatively simple. You just need to implement the *AttributeConverter<Duration, Long>* interface and annotate the class with a *@Converter* annotation. You should set the *autoApply* attribute of the annotation to true. This tells your persistence provider to use the converter for all entity attributes of type *java.time.Duration*.

Here you can see an example of such an *AttributeConverter*.

```
@Converter(autoApply = true)
public class DurationConverter implements
AttributeConverter<Duration, Long> {

    Logger log =
Logger.getLogger(DurationConverter.class.getSimpleName()
);

    @Override
    public Long convertToDatabaseColumn(Duration attribute)
{
        log.info("Convert to Long");
        return attribute.toNanos();
    }
```

```
    @Override
    public Duration convertToEntityAttribute(Long duration) {
        log.info("Convert to Duration");
        return Duration.of(duration, ChronoUnit.NANOS);
    }
}
```

Within the *convertToDatabaseColumn* method, I call the *toNanos* method of the *Duration* object to convert it to *long*. And the *convertToEntityAttribute* method uses the *of* method with *ChronoUnits.NANOS* to implement the inverse conversion.

That's all you need to do. You can now use the *OnlineCourse* entity which I showed you at the beginning of this post. Your persistence provider applies the *DurationConverter* automatically so that you don't need to adapt your entity mapping.

```
// Transaction 1: Persist a new OnlineCourse entity
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();

OnlineCourse c = new OnlineCourse();
c.setTitle("Hibernate Performance Tuning Online Training");
c.setVideoDuration(Duration.parse("PT5H55M"));
em.persist(c);

em.getTransaction().commit();
em.close();

// Transaction 2: Read an OnlineCourse entity
em = emf.createEntityManager();
em.getTransaction().begin();
em.find(OnlineCourse.class, c.getId());
log.info("The "+c.getTitle()+" contains " +
        c.getVideoDuration().toMinutes()/60 + " hours and " +
        c.getVideoDuration().toMinutes()%60 +
        " minutes of video");
```

```
em.getTransaction().commit();
em.close();
```

As you can see in the log output, the *DurationConverter* gets called twice to map the *videoDuration* attribute:

1. When the entity gets persisted, the *convertToDatabaseColumn* method gets called to map the *Duration* to a *Long* object
2. When the entity gets read from the database, the *convertToEntityAttribute* method gets called to map the *Long* to a *Duration* object

```
17:47:15,197 DEBUG [org.hibernate.SQL] - select nextval
('hibernate_sequence')
17:47:15,241 DEBUG [org.hibernate.SQL] - insert into
OnlineCourse (title, version, videoDuration, id) values (?, ?,
?, ?)
17:47:15,246 INFO  [DurationConverter] - Convert to Long
17:47:15,276 DEBUG [org.hibernate.SQL] - select
onlinecour0_.id as id1_0_0_, onlinecour0_.title as
title2_0_0_, onlinecour0_.version as version3_0_0_,
onlinecour0_.videoDuration as videoDur4_0_0_ from
OnlineCourse onlinecour0_ where onlinecour0_.id=?
17:47:15,290 INFO  [DurationConverter] - Convert to
Duration
17:47:15,294 INFO
[org.thoughts.on.java.model.TestAttributeConverter] - The
Hibernate Performance Tuning Online Training contains 5
hours and 55 minutes of video
```

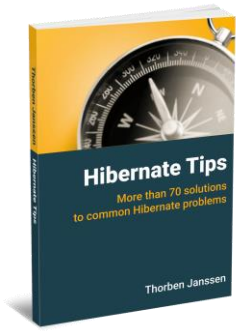
JPA Tip: Map a Duration attribute

Learn more

The *AttributeConverter* is a powerful and easy to use feature. You can use it to:

- Implement a custom enum mapping
- Persist *LocalDate* and *LocalDateTime* with JPA 2.1

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.