JPA 2.2 API And Reference Implementation

The following maven dependency adds the API jar of the JPA 2.2 specification to your project.

```
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>javax.persistence-api</artifactId>
  <version>2.2</version>
</dependency>
```

You can implement your application with the API jar but you, obviously, need an implementation at runtime. You can use EclipseLink 2.7 or Hibernate 5.2.

```
<dependency>
  <groupId>org.eclipse.persistence</groupId>
  <artifactId>eclipselink</artifactId>
  <version>2.7.0-RC3</version>
</dependency>
```

```
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>javax.persistence-api</artifactId>
  <version>2.2</version>
</dependency>
```

Limited Support For The Date And Time API In JPA 2.2

The JPA 2.2 specification maps only the classes of the Date and Time API for which the appendix section B4 of the JDBC 4.2 specification defines a mapping. These are:

Java Type	JDBC Type
java.time.LocalDate	DATE
java.time.LocalTime	TIME
java.time.LocalDateTime	TIMESTAMP
java.time.OffsetTime	TIME_WITH_TIMEZONE
java.time.OffsetDateTime	TIMESTAMP_WITH_TIMEZONE

JPA 2.2 supports the listed classes as basic types and you don't need to provide any additional mapping information.

```
@Entity
public class MyEntity {

    private LocalDate date;

    private LocalTime time;

    private LocalDateTime dateTime;

    private OffsetTime offsetTime;

    private OffsetDateTime offsetDateTime;

    ...
}
```

Extended Support in Hibernate 5

Hibernate supports the classes of the Date and Time API since version 5 as basic types. And it not only supports the 5 mappings defined in the JPA 2.2 specification. It also maps *Duration*, *Instant* and *ZonedDateTime* objects.

Java Type	JDBC Type
java.time.LocalDate	DATE
java.time.LocalTime	TIME
java.time.LocalDateTime	TIMESTAMP
java.time.OffsetTime	TIME_WITH_TIMEZONE
java.time.OffsetDateTime	TIMESTAMP_WITH_TIMEZONE
java.time.Duration	BIGINT
java.time.Instant	TIMESTAMP
java.time.ZonedDateTime	TIMESTAMP

Persisting A *ZonedDateTime*

The mapping of *ZonedDateTime* requires a more detailed explanation. Hibernate maps it to a JDBC *TIMESTAMP* without timezone information. It converts the *ZonedDateTime* to the local timezone and stores it in the database without timezone information.

This approach works fine as long as all instances of your application use the same local timezone. But you will run into problems as soon as you change the timezone setting or your cluster gets out of sync.

It's better to tell Hibernate which timezone it shall use. You can do that with the configuration parameter *hibernate.jdbc.time_zone*. You can set it in the *persistence.xml* file or on the *SessionFactory*.

```
<persistence>
  <persistence-unit name="my-persistence-unit">
    …
    <properties>
      <property name="hibernate.jdbc.time_zone"
value="UTC"/>
      …
    </properties>
  </persistence-unit>
</persistence>
```