Hibernate Tip: Map an Enum to a database column

Question:

How can I map an enum attribute to a database column? Which option should I prefer?

Solution:

JPA and Hibernate provide 2 standard options to map an Enum to a database column. You can either use its *String* representation or its ordinal value.

Both approaches have their drawbacks:

The *String* representation is verbose, and the renaming of an enum value requires you to also update your database.
The ordinal of an enum value is its position in the enum declaration. This value changes which requires you to update your database when you remove an existing value or don't add new values to the end of the Enum declaration.

You can define a custom mapping and avoid these issues with an *AttributeConverter*.

When you use JPA's and Hibernate's standard mapping, you can either rely on the default mapping using the ordinal or specify the mapping approach. You can do that with an *@Enumerated* annotation as I show you in the following examples.

Hibernate Tip: Map an Enum to a database column

If you don't provide an *@Enumerated* annotation or don't set an *EnumType* as its value, the ordinal of the enum value gets mapped to the database.

```
@Entity
public class Author implements Serializable {

    @Enumerated(EnumType.ORDINAL)
    private AuthorStatus status;

    …
}
```

	id [PK] bigint	firstname character varying(255)	lastname character varying(255)	status integer	version integer
1	1	John	Doe	0	0

Hibernate Tip: Map an Enum to a database column

If you want to map the *String* representation to the database, you need to annotate the entity attribute with *@Enumerated* and set *EnumType.STRING* as its value.

```
@Entity
public class Author implements Serializable {

    @Enumerated(EnumType.STRING)
    private AuthorStatus status;

    …
}
```

	id [PK] bigint	firstname character varying(255)	lastname character varying(255)	status character varying(255)	version integer
1	1	John	Doe	PUBLISHED	0

Learn more:

You can learn more about *AttributeConverter* and how you can use them to define custom mappings for enums and other Java types in:

- How to implement a JPA AttributeConverter
- JPA 2.1 Attribute Converter – The better way to persist enums
- How to persist LocalDate and LocalDateTime with JPA