Hibernate Tip: Override the primary key generation strategy

Question:

You explained that the *GenerationType.SEQUENCE* is the most efficient primary key generation strategy and that MySQL doesn’t support it.

What shall I do if I need to support PostgreSQL and MySQL with the same application?
Do I need to use the slower *GenerationType.IDENTITY* with PostgreSQL as well?

Solution:

No, you can use an external mapping file to override the mappings defined via annotations. So, you can use annotations to define your default mapping and override them if necessary.

Define The Default Mapping

In this case, you can use the *@GeneratedValue* annotation to define the *GenerationType.SEQUENCE* to generate the primary key values for the *Author* entity. This is the default strategy for your application. You can use it with all databases that support sequences.

```
@Entity
public class Author {
    @Id
    @GeneratedValue(
            strategy = GenerationType.SEQUENCE)
    @Column(name = "id", updatable = false, nullable = false)
    private Long id;
    ...
}
```

When you now persist a new *Author* entity, Hibernate retrieves a new primary key value from the database sequence.

```
20:45:18,203 DEBUG [org.hibernate.SQL] -
    select
        nextval ('hibernate_sequence')
20:45:18,245 DEBUG [org.hibernate.SQL] -
    insert
    into
        Author
        (firstName, lastName, version, id)
    values
        (?, ?, ?, ?)
```

Override Parts Of Your Mapping

And when you install your application with a MySQL database, you need to override the generation strategy with an additional mapping file.

By default, JPA and Hibernate check if an *orm.xml* file exists in the *META-INF* directory and use it to override the mappings defined by the annotations. So, you just need to provide the mappings you want to change.

In this case, it's only the generation strategy for the *id* attribute of the *Author* entity.

```
<entity-mappings>
    <entity class="org.thoughts.on.java.model.Author"
name="Author">
        <attributes>
            <id name="id">
                <generated-value strategy="identity"/>
            </id>
        </attributes>
   </entity>
</entity-mappings>
```

As you can see in the following log message, Hibernate now uses the *GenerationType.IDENTITY* which uses an auto-incremented database column to generate the primary key value.

```
20:42:47,414 DEBUG [org.hibernate.SQL] -
    insert
    into
        Author
        (firstName, lastName, version)
    values
        (?, ?, ?)
```

Learn more

You can read more about the different strategies to generate unique primary key values in:

- How To Generate Primary Keys With JPA And Hibernate
- How To Generate UUIDs As Primary Keys

And if you want to learn more database-specific mapping and query features, you should take a look at:

- 5 Things You Need To Know When Using Hibernate With MySQL
- Hibernate With PostgreSQL – 6 Things You Need To Know

Hibernate Tips Book