Mapping Java classes to the tables

Hibernate

Overview

- Understanding how the mapping between a class and a table works.
  - Adding a new column to the table from Java
  - Different column names from the name of the variable/attribute in the class
  - Different table name from the class name
  - Adding a new column to the table from Java
- Field and property access
- Reference Documentation for Hibernate

Mapping

- The @Entity annotation marks a class as a Hibernate entity, mapping it to a database table of the same name.
- All the fields/attributes that we are introducing in this class, will be columns in the table with the same name.

Using a Different Table Name

If the table name differs from the class name, use the @Table annotation.

For example we can have another name for the student table:

So in the Student class, we add an annotation @Table(name="TBL_STUDENT")

```
@Entity
@Table(name="TBL_STUDENT")
public class Student {
```

The @Table annotation comes from the jakarta.persistence package.

Mapping Class Attributes to Different Column Names

Use the @Column(name="column_name") annotation to specify a different column name.

For example:

```
@Column (name="NUM_COURSES")
  private Integer numberOfCourses;
```

The Column annotation is in the jakarta.persistence package.

Adding a new column in the table

To automatically update the table schema when adding new attributes, modify hibernate.cfg.xml:

```
<property name="hbm2ddl.auto">update</property>
```

Hibernate will automatically update the table with the new column.

Field access vs. Property access

- Annotations are placed directly above class attributes.
- Hibernate accesses fields directly, even if they are private.

Example:

```
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
private int id;
```

Field access vs. Property access

Property Access

- Annotations are placed above getter methods instead of fields.
- Requires explicit getter (getX()) and setter (setX()) methods.

Example:

```
@Id
@GeneratedValue(strategy=GenerationType.AUTO)
public int getId() { return id; }
```

Important Notes

- Do not mix field and property access in the same class.
- Different classes may use different access strategies.

Advantages and disadvantages of field/property access

Field access:

- Requires less code (no need for getters and setters).
- Keeps all annotations in one place for better readability.

Property access:

- Allows more control over how values are read and written.
- Enables additional logic in getter methods
  - (e.g., return name.toUpperCase();).

Ignoring Fields in Database Mapping

Use @Transient to exclude fields or methods from persistence.

Example:

```
@Transient

private String temporaryValue;
```

Hibernate will not map temporaryValue to a column in the database.

Reference Documentation for Hibernate

The following link is a manual for hibernate that you can refer to and search for your topic with Ctrl-f

https://docs.jboss.org/hibernate/orm/6.4/introduction/html_single/Hibernate_Introduction.html

For example we can look for @Table which will return 25 hits.

Comes soon

In the next chapters we will see explore:

- How to use logging.
- Creating multiple tables.
- Establishing relationships (one-to-many, many-to-one, etc.) using Hibernate