**Hibernate**

**What is Hibernate?**

A framework to **persist/store** **Java objects** into to a database.

Web app

Hibernate

DB

Benefits of Hibernate:

* Hibernate handles all of low level SQL code
* Minimizes the amount of JDBC code you have to develop
* It provides the Object-Relational Mapping (ORM)

How does ORM works?

As a developer we have to define mapping between Java class and database table

Database table

Java Class

id INT

first\_name varchar(20)

last\_name varchar(20)

email varchar(20)

Student

id: int

firstName:String

lastName:String

email:String

Hibernate

We tell hibernate that this java class maps to the database table and setup the mapping between these two tables.

This mapping can be done in two ways

* XML configuration
* Java Annotations

What is the difference between Hibernate and JDBC?

Hibernate actually uses JDBC for all the database communication. It is just another layer of abstraction on top of JDBC.

**Development Process:**

* Add Hibernate configuration file: This file tells Hibernate how to connect to the database.
* Annotate Java Class
* Write Java code to perform database operations.

**Hibernate Configuration file:**



This file is kept in the root directory of the application class path.

Different Properties:

|  |  |
| --- | --- |
| hibernate.dialect | This property tells Hibernate choose appropriate sql queries for the choosen database |
| hibernate.connection.pool\_size | Limits the number of connections waiting in the hibernate connection pool. |

**Annotate Java Class**:

Entity Class

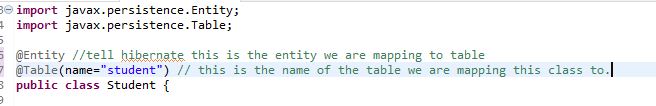
Java class that is mapped to a database table. This is just plain old java class with getters and setters, fields and with special annotation for mapping to database table.

There are two options for mapping Java class to database table.

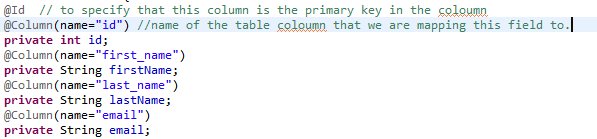
1. Xml configuration (legacy)
2. Java Annotations (Modern)

How to do mapping using java annotations?

Step1: Map the Java class with database table



Step2: Map Java fields with database columns.

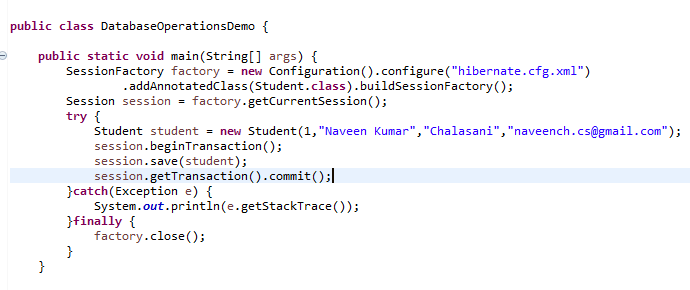


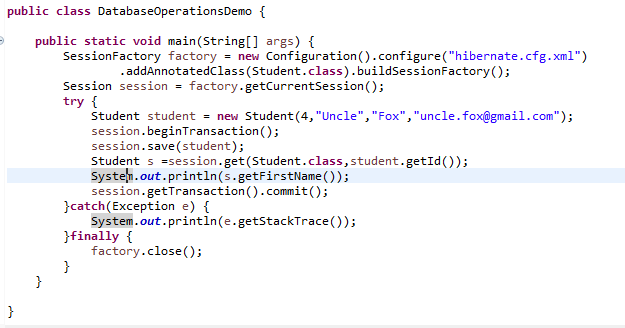
Before saving this java object to a database table we should know about two things in hibernate

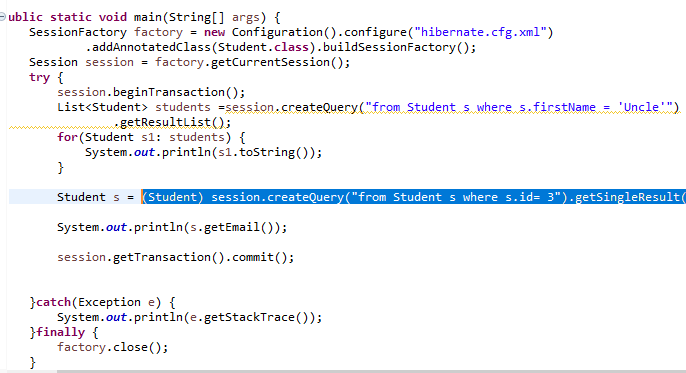
SessionFactory : this reads the hibernate config file and establishes database connection and creates session objects for database operations. Since this is a heavy weight object, it is created only once in the application.

Session: Wraps a JDBC connection and this is the one that helps with database operations. This is a short lived object that means for a given operation we use it and then throw it away. This object is retrieved from SessionFactory.

**Create and Read:**



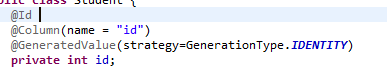


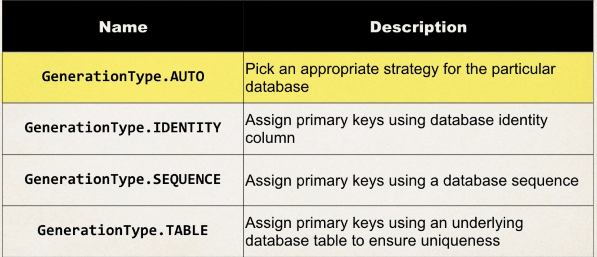


Primary Keys:

Uniquely identifies each row in a table and must be a not null and unique value.

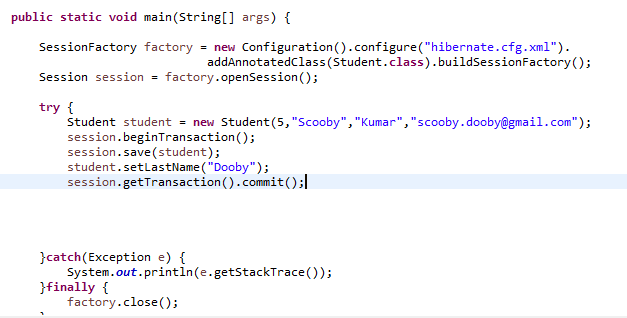
Primary key value generation strategy can be defined for a primary field. By default the strategy will be AUTO which means whatever is appropriate, hibernate will follow it. But if generation strategy is IDENTITY, then the column defined increment strategy will be taken into consideration. In this student table case, it is AUTO\_INCREMENT. Also there are two more strategies called TABLE and SEQUENCE.

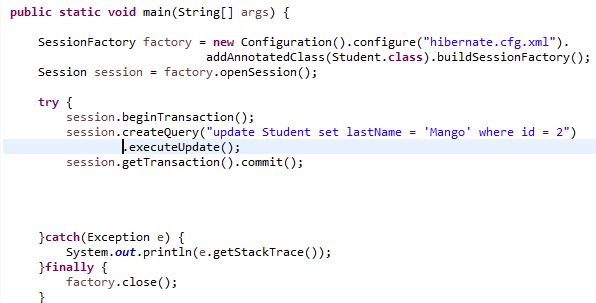




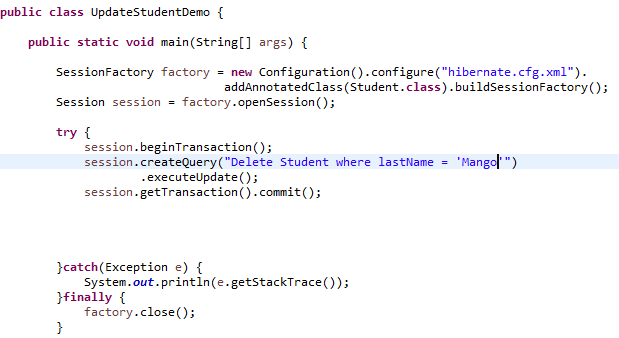
There can be CUSTOM generation logic that can be written in java as well.. Refer to udemy video for that number 190 video.

**Update Objects:**





**Delete Objects:**



**Hibernate Advance Mapping:**