**HIBERNATE WITH JPA**

ORM Tool:

* ORM is an acronym for OBJECT-RELATIONAL MAPPING.
* ORM is a technique for converting Java objects and relational databases(table).
* ORM internally implements JDBC itself. Although it was developed to overcome the drawbacks of JDBC, with the help of ORM, the java programmer doesn’t need to worry about database related properties & we also don’t need to write the SQL queries on our own.
* With the help of ORM, the classes in Java are directly converted to the tables in the database, the properties of the classes are converted as columns of the table & the objects created for the classes are converted into rows in the table.

Advantages of ORM tool:

* It saves time and efforts.
* It gives pace to development process.
* It reduces the development cost.
* It provides connectivity to the database.
* It makes development more object-oriented.
* Easy transaction management.
* No need to implement database manually.

ORM tools:

* Hibernate
* TopLink
* EclipseLink
* OpenJPA
* MyBatis(formerly known as iBatis)

Hibernate:

* Hibernate is a Java persistence framework that simplifies the development of Java application to interact with the database.
* It is an open source, widely used, lightweight, ORM tool.
* Hibernate implements the specifications of [JPA](https://www.javatpoint.com/jpa-introduction) (Java Persistence API) for data persistence.
* The performance of Hibernate framework is fast because cache is internally used.

JPA:

* Java Persistence API is a specification of java used to persist data between java object & relational database.
* It acts as a bridge between object-oriented domain models & relational database systems.

Difference between JPA & Hibernate:

|  |  |
| --- | --- |
| JPA | Hibernate |
| JPA is responsible for managing relational database in java. | Hibernate is an ORM tool used for saving the state of java object in database. |
| It is defined in javax.persistence package. | It is defined in org.hibernate package. |
| JPA is a specification. | Hibernate is an implementation. |
| It uses JPQL(Java Persistence Query Language) as an object oriented query language to perform database operation. | It uses HQL(Hibernate Query Language) as an object oriented query language to perform database operation. |

Steps to create Simple Maven Project:

* Go to create new wizard & type maven.
* Select Maven Project and click on next.
* Check the Create a simple project check box and click on next.
* Simple Maven Project will be created with a default structure as follows:
* Project\_Name
  + src/main/java
  + src/main/resources
  + src/test/java
  + src/test/resources
  + JRE System Library
  + src
  + target
  + pom.xml

pom.xml:

It is an XML(Extensible Markup Language) file that contains information about the project and configuration details used by Maven to build the project.

Dependencies required:

* mysql connector
* hibernate core

Note: The dependencies are available in https://mvnrepository.com/

Steps to add dependencies in pom.xml:

* Go to pom.xml
* Before the end of project tag, open a tag called as dependencies tag
* Within this dependencies tag, we’ll be adding our dependency.

Layers in Hibernate:

Database

DTO

(Data Transfer Object)

DAO

(Data Access Object)

* DTO layer will consist of the class that is supposed to be the table in the database. As the object of this class is transferred between database & java application. Hence the name Data Transfer Object.
* DAO will consist of the class where operations on DTO is access in the DAO layer. Hence the name Data Access Object.

Configuration of Maven Project for Hibernate:

Inside the src/main/resources folder, create a folder with the name “META-INF”. Inside the folder, create a file with the name “persistence.xml”.

Steps to configure persistence.xml file:

* Go to Maven Dependencies folder & look for ‘javax.persistence-api-2.2.jar’ & then expand it.
* Go to ‘javax.persistence’ package & expand it.
* Scroll down to the bottom & look for ‘persistence\_2\_1.xsd’ (xsd stands for xml schema definition)
* Open the xsd file & scroll down to line 50.
* Copy the content from line 50 to 56 & add to persistence.xml file.
* Remove the 3 dots within the persistence tag & add the persistence-unit, provider, properties & property tags in a proper structure as required.

Properties of JDBC & Hibernate are declared & defined in the persistence.xml file. The properties included are:

1. Driver Path
2. DBUrl
3. Username
4. Password
5. Hibernate Mapping
6. Display Query
7. Database dialect

* To configure the provider, we need to add fully qualified name of the provider class. The name of the provider class is HibernatePersistenceProvider. In Eclipse, we can find the class with a shortcut ctrl+shift+T.
* Search for the HibernatePersistenceProvider class & open it. Copy the fully qualified name & paste in the provider tag.

EntityManagerFactory:

* It is an interface present in javax.persistence package. It is used to provide an EntityManager.
* It provides an efficient way to construct multiple EntityManager instances for the database.

EntityManager:

* It is an interface.
* It is an API that manages the lifecycle of an entity instance.
* It performs some of the operations & it has methods to insert, update, delete & fetch the data.

EntityTransaction:

* It is an interface which provides the important method to handle transactions in JPA application.
* Transaction is set of operations that either fail or succeed.
* A database transactions consists of a set of SQL operations that are committed or rolled back.

persist(obj):

* It is used to insert a data into the database.

merge(obj):

* It is used to update the data in the database based on primary key(PK).
* If PK is already present, then update query will get executed.
* If PK is not present, then insert query will get executed.

remove(obj):

* It is used to delete the particular row based on PK only.

find(className.class,PK):

* It is used to fetch the data from the database using PK only.
* It will give the single row as output.