# JDBC\_JPA\_Hibernat PREP

Hibernate is just an another layer of abstraction on top of JDBC. So basically internally hibernate uses JDBC only. So in the background its JDBC api which is getting called.

Q. Write down steps for setting up Hibernate in Eclipse ?

Ans - Below are the steps which needs to be done,

* Create Eclipse project
* Download Hibernate files
* Download MySQL JDBC Driver
* Add JAR files to the Eclipse project ...i.e Build Path

ok so we have completed above steps successfully.

Q. What are steps for Hibernate Development Process Overview ?

Ans - Below are the steps on high level,

* Add Hibernate configuration file
* Annotate Java class
* Develop Java code to perform database operations

Below are the two key players in JPA/Hibernate programming,

* SessionFactory
* Session

Concept of primary key :- when we write below SQL for creating table,

CREATE TABLE student (

id int(11) NOT NULL AUTO\_INCREMENT,

first\_name varchar(45) default null,

last\_name varchar(45) default null,

email varchar(45) default null,

primary key (id)

)

Please note that yellow high-lighted thing "AUTO\_INCREMENT" thing will be handled by MySQL.

@Id annotation tells hibernate that this is a primary key of class/table. Sometimes we can even explicitly specify on how to generate primary key value. Below is the example,

@Id  
@GeneratedValue(strategy=GenerationType.IDENTITY)  
@Column(name="id")  
private int id;  
Q. What are different type of ID generation strategies ?  
Ans - Below are different type of mapping strategy ,

GenerationType.AUTO

Q. What is a HQL (Hibernate Query Language) ?  
Ans - HQL is an object oriented query language. We write HQL on java objects, mainly pojos and use their properties in the HQL. Hibernate translate these HQL's into SQL which in turn perform action on database.

Q. How to read and write dates from and to the database ?  
Ans - <This is a remaining item. Refer this later>

Q. What are Hibernate advanced mappings ?  
Ans - Below are the advanced mapping from hibernate perspective,

* One to One
* One to Many
* Many to One
* Many to Many

Just note that "One to Many" and "Many to One" are inverse of each other.  
  
GenerationType.IDENTITY  
GenerationType.SEQUENCE  
GenerationType.TABLE  
Please note that for "MySQL" database , IDENTITY is a good option. Also for Oracle database "SEQUENCE" is a good option. We can even write our own custom generator in case a need arise.

ALTER TABLE hb\_student\_tracker.student AUTO\_INCREMENT=3000  
truncate hb\_student\_tracker.student - This command will delete all the records from the "student" table and will also reset identity / sequence associated with this.

Q - What are "Primary Key" and "Foreign Keys" of a table ?  
Ans -

* Primary Key - Uniquely identifies a row in a table.
* Foreign Key - This links table together. A field in one table that refers to primary key in another table.

Q - What is cascade ?  
Ans - We can cascade operations. We can apply same operations to related entities. Like cascase delete, if we delete an instructor table then it will also delete instructor\_detail table.

Q. What do we understand by Eager vs Lazy loading ?  
Ans - When we fetch data, should we retrieve everything ?

* Eager will retrieve everything.
* Lazy will retrieve on request

So let say we have "Instructor" and "Courses" entity. So when we retrieve Instructor should we retrieve all of its course ? or we just load Insttructor and load courses taught by them on demand. This is an example of Eager Vs Lazy Loading.

Q. From hibernate perspective what do we mean by uni-directional and bi-directional ?  
Ans - Let say there are two tables "Instructor" and 'Instructor-Detail". So one way is you load 'Instructor" first and from there you can load 'Instructor-Detail". This is example of uni-directional.

Lets take another scenario. Let say in this case you load "Instructor-Detail" first. Now once "Instructor-Detail" is loaded you load "Instructor". This is example of bi-directional.

Q. What are different lifecycle of an Entity ?  
Ans - Below are different lifecycle of an Entity,

* Detach
* Merge
* Persist
* Remove
* Refresh

Q. What are different @OneToOne cascade types ?  
Ans - Different cascade types are,

* PERSIST
* REMOVE
* REFRESH
* DETACH
* MERGE

We just completed example of one-to-one and one-to-many mapping examples. Now we are moving towards eager loading vs lazy loading. Best practices of "Eager loading vs Lazy loading"

* Only load data when it is absolutely needed
* Prefer lazy loading instead of eager loading .

So what's the recommended approach guys ?

* Lazy loading will load the main entity first
* Load dependent entities on demand (lazy)

Q. What are the techniques of loading lazy data ?  
Ans - Below are the techniques using which data can be loaded lazily,

* First of all Hibernate session should be open or available. Use session.get and call appropriate getter methods.
* Use Hibernate query or HQL

fetch=FetchType.EAGER and fetch=FetchType.LAZY