##### **HB video 1 to 6 covered here** #####

This video is about the some of the basic annotation

@Entity // This tells Hibernate to make a table out of this class

**public** **class** UserInfo {

* @Entity is applied only above the class.
* Above annotation denotes the Hibernate that there is a Table named UserInfo available in DB, so map the table to this UserInfo Class.

@Entity(name="UserInfo")

**public** **class** User {

* since User is the reserved keyword in DB so we can’t create the table name in such a way. But we can have the POJO class name as User.
* So we are denoting the Hibernate that the Table name is UserInfo in the DB and Map the table to User POJO in our POJO with the help of name parameter which is available in @Entity(name="UserInfo").
* If U are not going to use the reserved keyword for UR POJO class name then just leave it as @Entity annotation only.
* In this Case we also @Entity(name="UserInfo") provide the name to the Entity Itself.

@Id

**private** Integer userid;

* @Id is applied only above the instance variable.
* Each instance variable in UR Entity will be mapped to each one of the column but you have to specify which instance variable is mapped to Primary key column.
* @Id annotation maps the particular variable to primary key column.
* you can’t have @Id annotation above multiple fields in the class, since the table can have only one primary key column.

@Column

**public** String User;

* @Column is applied only above the instance variable.
* It’s not mandatory to put @Column annotation above each fields in the entity.

@Column(name="userid")

**private** String user;

* Since User is the reserved keyword in DB so we can’t create the column in such a way. But we can have the POJO instance variable name as User.
* So we are denoting the Hibernate that the Column name in the Table is Userid and Map the column to User field in our POJO with the help of name parameter which is available in @Column(name="userid").

@Transient

@Column(name="Last\_name")

**private** String lastName;

Consider U don’t want to save or select field “**last name**” to DB when JPA auto create the query for U. Then the filed should be annotated with @Transient. So U don’t want to set the value in

setter method in entity for last name.

@Temporal(TemporalType.***DATE***)

**private** Date DOB;

* Annotation temporal is applied above the Entity field of java.util.Date.
* Unless U say Hibernate explicitly(Even U didn’t add the @Temporal annotation it will be TemporalType.***TIMESTAMP***). By default Hibernate will insert Date in the format YYY/MM/dd HH:MM:ss that is the TemporalType.***TIMESTAMP***
* By default Hibernate will insert TemporalType.***TIMESTAMP*** but I may be focused only on Data so providing the TemporalType.***DATE*** Hibernate will insert only the DD/MM/YYY.
* By default Hibernate will insert TemporalType.***TIMESTAMP*** but I may be focused only on Time so providing the TemporalType.***TIME*** Hibernate will insert only the HH:MM:ss.

@Entity

@Table(name="UserInfo")

**public** **class** User {

* Above annotation @Table(name="UserInfo") we are saying to hibernate just create the table name as "UserInfo" and entity name still stays the same as “User”.
* So when u write a native query u can use the table name not the entity name.
* When We write the HQL query we use the Entity name and When we write the Native Query we use the table name.

@Lob

**private** String description;

* By default, the String field in Entity class will be created as Varchar Type column in table with size 255.
* But I have a situation that my **input** might be greater than the Fixed size 255, so annotating the field with @Lob will help us in dynamic increase in column size.
* There are other U can do for CLOB and BLOB as same as LOB