**Value Object Embedded as List to Entity**

* Consider I want to have track of user lived over the years, so I can’t instantiate the one Address field statically.
* There might be many addresses he lived over the years, so the idea here is we are going to inject the VO to entity as the list of Addresses using the @ElementCollection annotation.
* Earlier the VO will be embedded with entity table itself, since we had only one static address field in Entity, so there will be one row available for each user. But now we have multiple address.
* Since we have multiple address I doesn’t look good when we have same user data associated with multiple address in the entity, so Hibernate cleverly creates a new separate table for list of address.
* Hibernate creates the separate table for Address. What is the table name it creates for us? How does it associate the user with multiple address? What is the column name in address(VO) table associate with user entity?
* Table name = Entity name + filed name of address(VO) reference in entity.
* Column name = Entity name + field name of primary key in entity.

@Entity(name="UserInfo")

**public** **class** User

{

@Id

@GeneratedValue

**private** Integer userId;

@ElementCollection

**private** Set<Address> listOfaddress=**new** HashSet<Address>();

JOIN TABLE named by Hibernate is below

TABLE NAME = UserInfo + listOfaddress

JOIN COLUM(Foreign Key Column) named by Hibernate is below

COLUMN NAME = UserInfo + userId

@Embeddable

**public** **class** Address {

@ElementCollection annotation helps us to add the Embeddable address class to add as a list to entity to user.

Consider I want to have my own table name and Column name for the address VO, since hibernate creates the VO table name and VO primary column name that maps the user entity. So we are overriding the table name by @JoinTable annotation and overriding the column name by @JoinColumn annotation.

@ElementCollection

@JoinTable (name="JOIN Table name",

joinColumns= @JoinColumn (name="(JOIN COLUMN)Foreign key column name"))

**private** Collection<Address> listOfaddress=**new** HashSet<Address>();

@JoinTable annotation helps us in providing the parameter to naming the child table and foreign key column.

JoinTable annotation takes **name=** parameter where you can give ur own table name.

JoinTable annotation takes **joinColumns=** parameter where you can give ur own column name for foreign key column.

joinColumns=takes@JoinColumn annotation as the value where @JoinColumn annotation take foreign key column name.