**PROXY OBJECT & LAZY Initialization and EAGER Initialization**

If you are having a collection in your entity class as-

@Entity

public class UserDetails{

@Id

private int userId;

private String userName;

@ElementCollection

Set<Address>listOfAddresses=new HashSet();

.......

.......

}

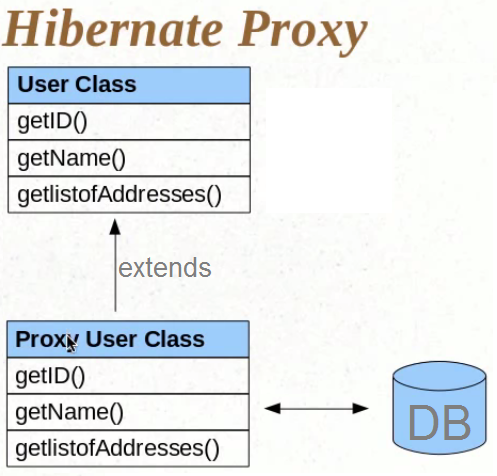
and suppose that each UserDetails object have a collection of hundreds of Address objects in it.(i.e. Each user has thousands of addresses). So if we are fetching a record as-

UserDetails user=(UserDetails)session.get(UserDetails.class, 1); // line 1

Then it is fetching a record corresponding to primary key value=1. So ideally it should contain the list of all the addresses of the first user. But think for a while about the performance if user has hundreds of addresses. Performance will drastically decrease.

But this does NOT happen by default. i.e. whenever you fetch the UserDetails object (as in line 1) then initially it will NOT contain the list of addresses.

i.e. 'listOfAddresses' will not contain anything. But as you call the getter (getListOfAddresses()) then at that time hibernate will fetch the list of addresses from DB. All this happens with the help of proxy class.



A proxy class (at runtime) is created and this proxy class extends the user class (i.e. UserDetails class). And by using the following line we get the object of proxy class NOT of user class-

UserDetails user = (UserDetails)session.get(UserDetails.class, 1);

and when we call getters then the getters of proxy class are called and from inside the proxy class’s getters the getters of parent(i.e. user class) are called.

But in case of getter corresponding to collection (getListOfAddresses()) as the getListOfAddresses() is invoked then first of all list of all the addresses is fetched from DB then the actual user’s getter is called from inside the proxy’s getter.

This whole process is called *lazy initialization*. As the list of addresses are fetched only when we needed it. In lazy initialization, first level fields are initialized beforehand. And remaining fields are initialized when we need them.

The opposite of lazy is *eager initialization.* In eager initialization, all the fields are initialized beforehand.

@ElementCollection(fetch=FetchType.EAGER)

Set<Address>listOfAddresses=new HashSet();

So when you call get() as follows-

UserDetails user=(UserDetails)session.get(UserDetails.class, 1);

listOfAddresses will be initialized immediately.