**Get Vs Load:**

**🡪get ():**

🡪Purpose:

-It is used to read the data from the underlying DB immediately and populates data into corresponding entity class obj, hence it called as "Eager-Initializing/ Eager-loading/ Early-loading".

🡪Internals of get() /How get() works:

Person person = (Person) session. get(Person.class,1);

session obj will contains the info about mapping and cfg details, whenever we call get() on session obj then session will reads canonical-name/ fully qualified name of the class(Person) and searches the same in the meta data of the mapping file and identifies its corresponding table, PK column using the <id> tag and gives this information to the dialect to generate the Query against the PK and then sets PK-value to the Query given by the programmer then immediately opens the connection with DB and then fetches one entire record and populates data into the corresponding entity-class obj.

Hence get() is called as "Eager-Initializer/ Eager loader /Early loader".

**🡪load ():**

-It is used to fetch the data from underlying DB on demand hence it is called as Lazy "Initializing/ Lazy loading".

🡪Where we can use load()/ Purpose of load():

-One person can have many/multiple addresses.

class Person {

private String ssn;

private String firstName;

private String lastName;

private String gender;

private String dob;

private List<Address> address;

public void setSsn(String ssn) {}

public void getSsn(String ssn) {}

//similarly all setters & getters

}

class Address {

private String addressLine1;

private String addressLine2;

private String city;

}

PERSON

SSN FIRST\_NM LAST\_NM GENDER DOB

1 JAMES G MALE 02-09-98

ADDRESS:

ADDRESS\_ID SSN ADDRESS\_LINE1 ADDRESS\_LINE2 CITY

1 1 AMEERPATE TS HYD

2 1 SR NAGAR TS HYD

3 1 HITECH-CITY TS HYD

Person person = (Person) session.

load(Person.class, 1);

-Here hibernate will not reads the data from underlying DB it simply creates an empty obj and holds the pk-value.

person.getFirstName ();

-Here we are calling the one of the accessor method of to get the first name hence it will goes to the underlying DB and reads the entire record of the PERSON table record against the pk-value(Here we are accessing only one accessor but hibernate will reads the entire record of the PERSON table to reduce the n/w round trips b/w the prog and DB if later we wants use the additional info related to the same Person) and populates into the corresponding entity class obj, but it will never reads the address class obj bcz we didn't specified and we dont want Address class obj which is an associated/relatinal class (child table).Hence it is called Lazy loading on demand.

-If we specify the address.addressLine1 (); then it will reads entire record of the person and and the its related data address also at single shot and populates into the Person entity-class obj.

-If we want person details and as well as and its related addresess also then we use get() to retrive data from from underlying DB and populates into an Person class obj at single shot, as follows

Person person = (Person) session.get(Person.class, 1);

🡪Internals of load() /How load() works:

Person person = (Person) session.

load(Person.class, 1);

person.getFirstName ();

Whenever we call load() method then

1. Hibernate will not returns prepopulated data bcz load() is an lazy-initializer hence it returns empty obj.
2. Then upon empty obj if we call any accessor method to get the data then we need to write the logic to access the data from the DB which is complicated for the programmer.

Hence Hibernate should not return the empty person obj rather should return some obj which contains logic to fetch the data from the DB.

That's where Hibernate will not returns empty entity class(Person)obj rather it will returns an obj which is seems /same/ similar to the original obj but it will contains the logic to access the data from the DB upon calling any one of the accessor method is called as "Proxy" class-obj.

i.e.

Person person = (Person) session.

load(Person.class, 1);

person.getFirstName ();

Here 'person' is not a original Person class obj it is an proxy-obj of the Person class which seems to be same as the original obj.

i.e. whenever we are calling load() method it will creates an proxy obj by creating an proxy class by extending from Person class and overrides all the attributes from the Person class as follows.

class Person$Proxy extends Person {

private Object id;

private Person person;

//Here person-obj is looking like a original obj but it is an proxy obj bcz it is populated with data in the proxy class by the setters and getters of the Proxy class (but not by the original setters and getters) by declaring as an attribute which is same as original obj.

public Person$Proxy (Person person) {

//Here Person$Proxy constructor is taking parameter as Person which is mentoned in the load() method

this.person = person;

}

public void setSsn(String ssn) {}

public void getSsn() {}

public void setFirstName(String firstName) {}

public void getFirstName() {}

//similarly all setters & getters

public Person load(Person person) {

if(person==null) {

//goes to DB and fetches the data on demand and populates data into attributes setters. sets the all attribute values to the person obj againts the PK.

}

return person;

}

}

Whenever we are calling the load() method then Hibernate understood that there is an load() then it will reads the class name and id and creates an Proxy class by extending original class and passes these are class and id are attributes to the proxy class and overrides the all the setters and accessor methods from the original class to the proxy class and creates an proxy obj and returns this proxy obj using the mapping meta-data of the session. Later whenever we are calling any accessor method then it will goes to the load() method of the session and then session will ask the dialect to generate the query then sets the PK value to the load() method then it will make a DB call to fetch the data from the DB and populates the entire record of the data into the proxy-obj and then return back to the java program.

Whenever we call another attribute,

person.getDob(); then agin it will not goes to the DB simply it chaeks in the proxy and returns the value form the proxy obj.

If we call person.getAddress(); then it will goes to the proxy but write now associated data is not available in the proxy hence it will goes to the DB and fetches the data from the associated table(by identifying Address class corresponding table from the mapping meta-data of the session) and returns back to the proxy obj.

🡪Internal code:

Note:

-If we call id as a accessor i.e person.getSsn(); then it will not make a call to the DB , whenever we call other than id attribute then only goes to the DB bcz we are only specifying the id to the load().

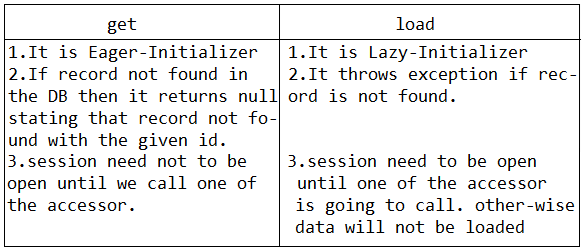
**-If the id is wrong then exception will not rise any exception in case of load(), bcz we are passing the id hence it will not show exception and whenever we are trying to call one of the accessor method(other than id accessor) then it goes to DB, if id not found then it throws an exception.**

**-get() will returns 'null' stating that there is no record in the DB.**

But load() method will throws an exception stating that record is not found in the DB, bcz it will creates an proxy for the actual entity class by holding the 'id' of the entity class that there exists an id in the DB hence it will never goes to the DB and if we call any accessor method then it will goes to the DB then checks the record is available or not if not there then if it returns null then it showing that there is a record but there is no data in the record which is sense-less

Hence to avoid this sense-less thought it shows an exception.

**🡪Difference between get & load:**



**Note:**

If lazy= "false" then uses only original obj but not creates the proxy class. i.e same as get() method.

lazy="true" (default lazy loading for load(-))

lazy="false" eager-loading same as get() for load(-))

lazy attribute effect will not be there for get(-).