# Day25

## ***Loading of object using sess.load() method***

* This method can perform both eager loading and lazy loading (by default it will perform lazy loading)
* When this method is called first it returns In-memory proxy class object (which is sub class Entity class), when we start non-identifier (other than id property method) the hit to db s/w take place to get the record and store the record by creating another object for entity class (real Object)

## ***Proxy Class***

* The class that contains additional logics to apply before using real class logics.
* Working with proxy class is gives the felling of working with real class. Because both contains same methods and implements or extends from same class.

|  |
| --- |
| Client App m1()  m1(){  Real logic  }  Real object  m1(){  .. additional logics  real.m1()  }  Additional logics  Proxy Object |

* ***Normal Java class***

|  |
| --- |
| .java(HDD) 🡪 .class(HDD) 🡪 JVM loads .class file 🡪 execution |

* ***In Memory Class***

|  |
| --- |
| Source code generation(JVM) 🡪 compilation & generation of .class (JVM) 🡪 Load of byte code 🡪 (.class code 🡪 execution (JVM Memory) |

* While working with proxy design pattern we have two type of In Memory classes
  + **Normal In-Memory class** (created as .class, .java by programmer)
  + **Dynamic In-Memory Proxy class** (generated by special libraries at run time)
* ***NOTE:*** With respect to sess.load() method the hibernate framework uses **javaassist** library to generate proxy class as In-Memory class.
* ***NOTE:*** sess.load() method is designed based on Proxy DP because it internally generated In-Memory Proxy class.
* Diagrammatic Flow of sess.load(): Lazy Loading

#10

Table (output)

#6

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1  App1.java  Product proxy = sess.load(Product.Class, 9001);  #3  #2  In memory proxy class generation as the sub class of given entity class having given id values as the data of the object.  #4  Oracle DB   |  |  |  |  | | --- | --- | --- | --- | | 9001 | Table | 800 | 10 |   Jdbc code + select query  proxy.getPName(); (non-identifier method)  Select \* from product where PID=?  #5  Result Set rs   |  |  |  |  | | --- | --- | --- | --- | | 9001 | Table | 800 | 10 |   Record  #7  #9  real.getPname()  Object mapper will take care of mapping to the entity class   |  | | --- | | 9001 | | Table | | 800 | | 10 |   Entity class object inside proxy class  #8  Signature of load method:  public <T> T load(Class clazz<T>, Serializable id) |

# ***Day-26***

While calling ses.get() method if the given id based record not found then we get “null” values

While calling ses.load() method first it will return proxy object when we call getter method on proxy object when we call getter method on the proxy object then we get **ObjectNotFound Exception** if record is not available i.e., ObjectNotFound Exception is not thrown by ses.load() method that is thrown by getter method that are overridden in in-memory proxy class.

## ***How to perform Eager loading by using ses.load() method***

* Keep lazy=”false” in class tag of mapping file. Example:

|  |
| --- |
| <class name = "com.ab.entity.Product" table = "PRODUCT" dynamic-insert="true" dynamic-update="true" lazy="false"> |

* And call sess.load() metho in client application in regular manner
* NOTE: Generally changing mapping file info like lazy=”false” or lazy=”true” is very difficult in the middle of the application execution is very difficult. Se we use ses.get() method for eager loading of objects, similarly we use ses.load() method for lazy loading of records.