# Previously we used to do like this

MySQL client DB

Request

Response Response

# Now we will use java client instead of MySQL client

Now db(chip) we have but we can’t use directly so we have driver(different for different db) which will be used from java code now different db version can have different driver but we are going to use driver directly so our code will be totally dependent on the driver and if in future we want to use different database then our code will be totally different that we don’t want

We want our code to be independent of any driver, We do not want our code to be directly dependent on a particular driver for this jdbc api is created in java

java client

jdbc driver DB

(low level code)

now Instead of using the driver directly you're going to use database connector API. Now you can move to any database your code will not get changed, because you are not directly dependent on driver, you are dependent on the standard specification.

Driver has the low level code to connect you application to the database

Let us see other problems now you have started using jdbc now let us see why jdbc is not used in industry that much I mean we never code on jdbc but it always gets used the reason is-

Jdbc is created for sql databases okay so you have sql database in which tables are in the metrics format(row column)

|  |  |  |
| --- | --- | --- |
| id | name | age |
| 1 | sohan | 22 |
| 2 | rohan | 23 |
| 3 | mohan | 24 |

# Now in java side we are not representing in this way we represent like

So you can see there is definitely mismatch here we have object oriented side, here we have metrics format. That means if you're going to write this details which you have in this object if you're going to insert in table, and let us suppose that row doesn’t exist and you're going to insert so you have to convert from object to row format and vice versa so at both side we will do conversion n number of times and this is a big big headache that’s why jdbc is not much used

So what we required here is ORM(object relational mapping) that will do conversion both side automatically

Now you have created java application and ORM library you are using is hibernate( that is vendor based implementation) so now in future if you want to use toplinks instead of hibernate so then lot of code changes will be there why because you are using the api which is not standard

Sun Microsystems actually created a standard at a later time, and that is very popular which is Jpa that is only used for sql based database

🡪JPA is a library and hibernate is implementation,JPA contains lot of interfaces

JPA

Hibernate TopLinks

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Now, your code will not be dependent on vendor based API your code will be using Jpa standard. So, in the future, if you want to move to another vendor like toplink you can move because your code is not directly dependent on particular implementation

conversion

conversion

so what kind of operation you are able to do obviously CRUD operation

C🡪Create the row

R🡪 Read/fetch the row

U🡪change the row

D🡪 delete the row

So the API we are going to use is EntityManager based API which is there in JPA so there we have CRUD operations method

C🡪persist(obj)

R🡪 find(id)

U🡪merge(obj)

D🡪 remove(row)

So using EntityManager object we are goin g to work on the Entity(ex-Product,Employee etc)

Now each entity manager is associated with persistent context(one to one relationship)

em1 pc1

em2 pc2

One entity manager cannot use persistent context of another entity manager, what a persistent context is persistent contents are the set of entity object which you have ,pc is like cache So you can see, this is your application.

Pc will contain the object of entities

🡪 it will only contain the unique entity object

For example you have product with id 1 so you can’t have another product object with id 1

# Pc(first level cache)

# DB

em

give

To whatever you are going to insert or going to fetch from the database , For example, you are doing insert operation, the before of our row is inserted, the object will be kept in the persistent context. Similarly, when you're fetching our object which is equivalent to row details First It will also be kept in the persistent context.

Now our entity object is stored in the pc that is cache memory now you have Samsung with id 1 and now of you are saying fetch me product details with id 1 your disc will not be hit ,your database is slow it will give you from the pc , if you will ask 20 times it will give you from pc ,disc will not be hit if it is not in the pc first it will be fetched from disc then will be kept in the pc then you can get from the pc so first time it will be kept in the pc and rest 19 times you can get from the pc itself your disc will not be hit discs are slow databases are not that fast your cache are first similarly for insertion object will first stored in the pc then it will inserted in the database

Now we already know pc is associated with the entity manager so if you em doesn’t exist pc will not exist it will be destroyed

Then you will create a new entity manager, you will get a new persistent context

Now whenever you object is created ,new object is created like this

Product product=new Product(1,”Samsung”,10000);

Jpa and hibernate know nothing about it at this time our object is in new state or transist state now you are doing

em.persist(obj) or em.merge(obj) so then this object will kept in the persistent context now hibernate knows about it, now our object is in managed state previously it was in new state

Now At a later time, let us suppose the object be removed from the Persistent contest or your entity manager gets close so pc will be destroyed now again object will be again in the detached state again hibernate knows nothing about it but the difference between the new state and the detached state is that in detached state you object will have Atleast the primary key, ID or whatever, which is representing the row in that table. But the new state your object might contain or might not contain ID. And it might not represent any of the row on the table , but your detects objects will surely be representing one of the row of the table

# Now do some practical

Now I will add hibernate core relocation dependency in pom.xml file but i'm not adding JPA dependency because hibernate has transitive dependency on the JPA, so if I am adding hibernate by default we will get JPA dependency