Hibernate Adaptation

In the previous chapter, we did a detailed discussion why using ORM framework is advantageous over JDBC.

But things to remember ORM frameworks use JDBC internally, ORM put a wrapper on a JDBC so the developer can overcome the problem stated in the previous chapter and only concentrate on business coding.

Hibernate is one of the most used ORM frameworks. In this chapter, we discuss why we adopt Hibernate.

***Have you ever thought why there is so much responsibilities/complexities , when we try to integrate RDBMS with OOPS?***

The main reason in Impedance mismatches between OOP programming and Database layer. OOPS and RDBMS have their own style to store data, communicate with each other and maintain Identity.

These two are separate by all nature so when we try to integrate or communicate between these two we have to take care , how data is presented to each layer.

OOPS works with Objects so OOPs wants to see data as Object.

RDBMS store data in tabular format so it wants to see data in tabular format.

So Obviously we need an Adapter which is capable of transforming tabular data to an Object Structure and Object data to a tabular structure. Hibernate is doing the exactly same thing not only that it is more advanced so Hibernate is a good choice when we try to integrate Database.

I am saying a good choice not obvious choice as different parameters are acting to choose JDBC or ORM. This is not the scope of this article.

Let see the mismatches in a detailed way.

**OOPS VS RDBMS**

|  |  |  |
| --- | --- | --- |
| **Description** | **OOPS** | **RDBMS** |
| Uniqueness | In oops we use == or equals method to check y=two objects are equals or not | In RDBMS we use Primary key or composite key to identify unique tuple. |
| Inheritance | In oops it is one of the main property to extends properties from parent. | In RDBMS we can not extends a table. |
| Association | In OOPs we use HAS-A and IS-A relation, to associate one object to another. | In RDBMS we use foreign key to join two tables. |
| Semantics | In oops we create a class to define a real life object and use properties to define its attributes, | In RDBMS we create a table to define a real life object and use columns to define its attributes. |

Hibernate resolves all the problem so using hibernate you don’t have to bother about the impedance mismatch and Hibernate produce database table in an Object form so you can take action upon them as you do in regular java object/POJO. So integration with Database is very easy. Also, we can introduce Inheritance, Collection,Caching in our code as those are the features of OOPs so although underlying database does not support it Hibernate Support this features and changes them accordingly so it can be stored in the database as a tabular format.

**Advantages of using Hibernate**

1. Hibernate represents each tuple in a table as a simple POJO.
2. We can use HQL, which is standard domain specific language similar to SQL . By using this we don’t have to bother about database specific SQL , Hibernate translate hql to database specific SQL.
3. Hibernate takes care of Transaction automatically.
4. Hibernate can generate ID or can use native application to generates ID.
5. Hibernate use Caching to avoid hitting the database every time when a query fired, it increases performance.
6. Hibernate use lazy loading, by this, hibernate delays to fetch associated mapping objects until it is requested. So it increases performance and memory management.
7. Hibernate allows concurrency using the versioning technique , without impacting performance.