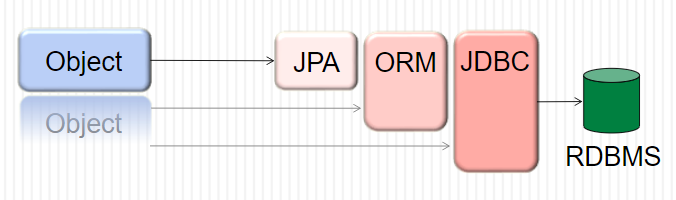
**JPA vs JDBC: How To Choose One Over The Other**

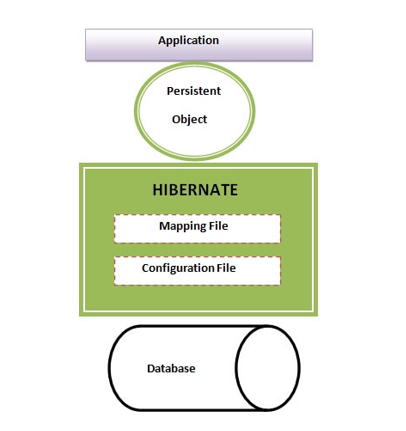


The software architects usually face significant technological challenges to decide how to communicate with back-end database systems. The debate between JPA and JDBC is often the deciding factor, as the two database technologies take very different approaches to work with persistent data when it comes to N-tier applications and applications in general.

But if you understand the bigger picture then it’s easier for you to choose and use them. So let’s analyze the key differences between them.

1. **Ease of migration**: In JDBC, the table name and column names are tightly coupled with the application, so any change in the database you have to modify the code, so it needs again development cost, as well as testing cost, which is waste of time, money and energy. But in JPA, the table and column names are not tightly coupled with classes, those are configured with mapping file(one xml), if any change in DB you can change in Mapping file.





**2. Exception Handling** : JDBC throws checked exceptions, such as *SQLException, and writing try-catch* block is must. But JPA framework uses only unchecked exceptions. Hence, we don’t need to catch it and thus saves a lot of typing for the programmers.

**3. Performance**: JDBC is generally faster then JPA, but in JPA you can benefit from caching, where it uses caching mechanism to keep the number of database requests to minimum. It also uses connection pooling, so that the database connections that are already established can be reused. This helps in improving the performance of the application.

**4. Object Mapping**: In JDBC we get the result in plain text and we need to take care of mapping the result with Java objects while in JPA, you deal with objects.

So instead of performing pure SQL on your database:

INSERT INTO STUDENT  
VALUES('1', 'XYZ');

JPA will do the mapping to your database table and it will look something like:

Student student= new Student();  
  
student.setID("1");  
  
student.setName("XYZ");  
  
entityManager.persist(student);  
  
entityManager.getTransaction.commit();

Now you can focus on your business logic and on your java objects only instead of mixing SQL with your java code.

5. **Level of Abstraction**: JDBC is a low level standard for interaction with databases where we’ll be talking to the database directly. JPA is higher level standard for the same purpose. JPA allows you to use an object model in your application which can make your life much easier.

Summarizing it all, JDBC is considered the preferable alternative if an application will use a simple database and we don’t plan to migrate it to a different database vendor.

Using JPA, we can code the Java applications using object-oriented principles and best practices without having to worry about database semantics.