Advanced Java Programming

Hibernate

Hibernate: Introduction

- Hibernate is used to convert object data in JAVA to relational database tables.
- It is an open source Object-Relational Mapping (ORM) for Java.
- Hibernate is responsible for making data persistent by storing it in a database.

Object-Relational Mapping (ORM)

- It is a programming technique for converting object-type data of an object oriented programming language into database tables.
- Hibernate is used to convert object data in JAVA to relational database tables.

JDBC v/s Hibernate

JDBC	Hibernate
JDBC maps Java classes to database tables (and from Java data types to SQL data types)	Hibernate automatically generates the queries.
With JDBC, developer has to write code to map an object model's data to a relational data model.	Hibernate is flexible and powerful ORM to map Java classes to database tables.
With JDBC, it is developer's responsibility to handle JDBC result set and convert it to Java. So with JDBC, mapping between Java objects and database tables is done manually.	Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects, hence reducing the development time and maintenance cost.

JDBC vs Hibernate

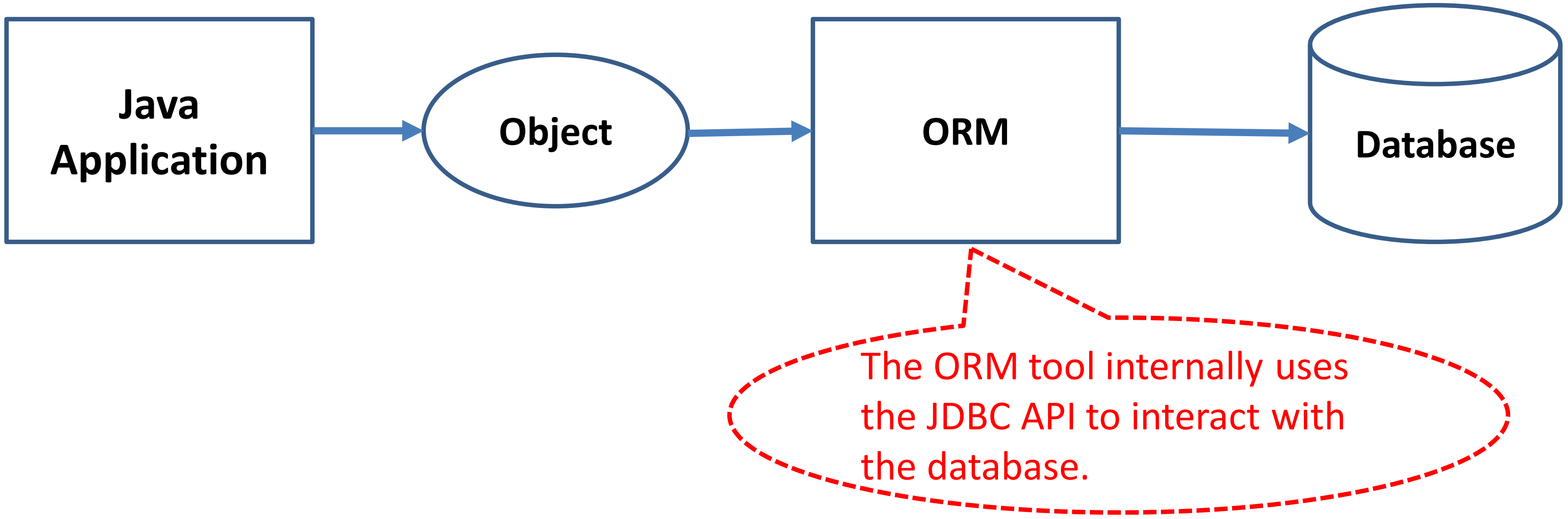
JDBC	Hibernate
Require JDBC Driver for different types of database.	Makes an application portable to all SQL databases.
Handles all create-read-update-delete (CRUD) operations using SQL Queries.	Handles all create-read-update-delete (CRUD) operations using simple API; no SQL
Working with both Object-Oriented software and Relational Database is complicated task with JDBC.	Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.
JDBC supports only native Structured Query Language (SQL)	Hibernate provides a powerful query language Hibernate Query Language-HQL (independent from type of database)

Overview of Hibernate

Hibernate Framework

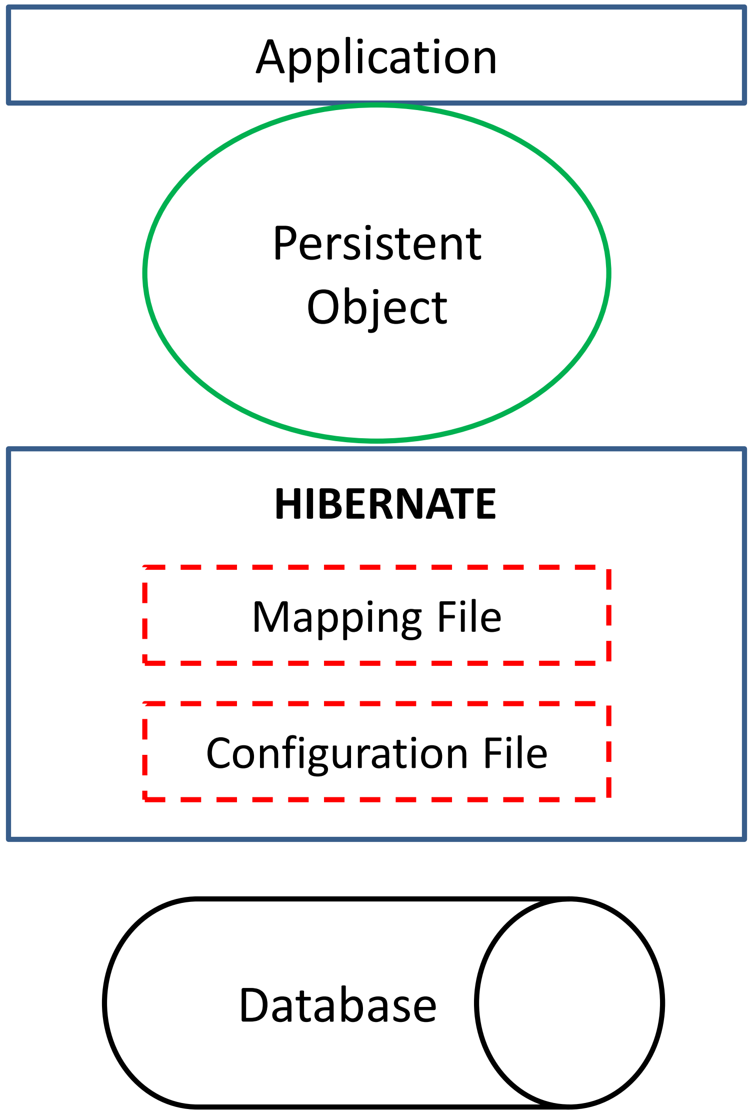
- Hibernate framework simplifies the development of java application to interact with the database.
- Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool.
- An ORM tool simplifies the data creation, data manipulation and data access.
- Hibernate is a programming technique that maps the object to the data stored in the database.

Hibernate Framework

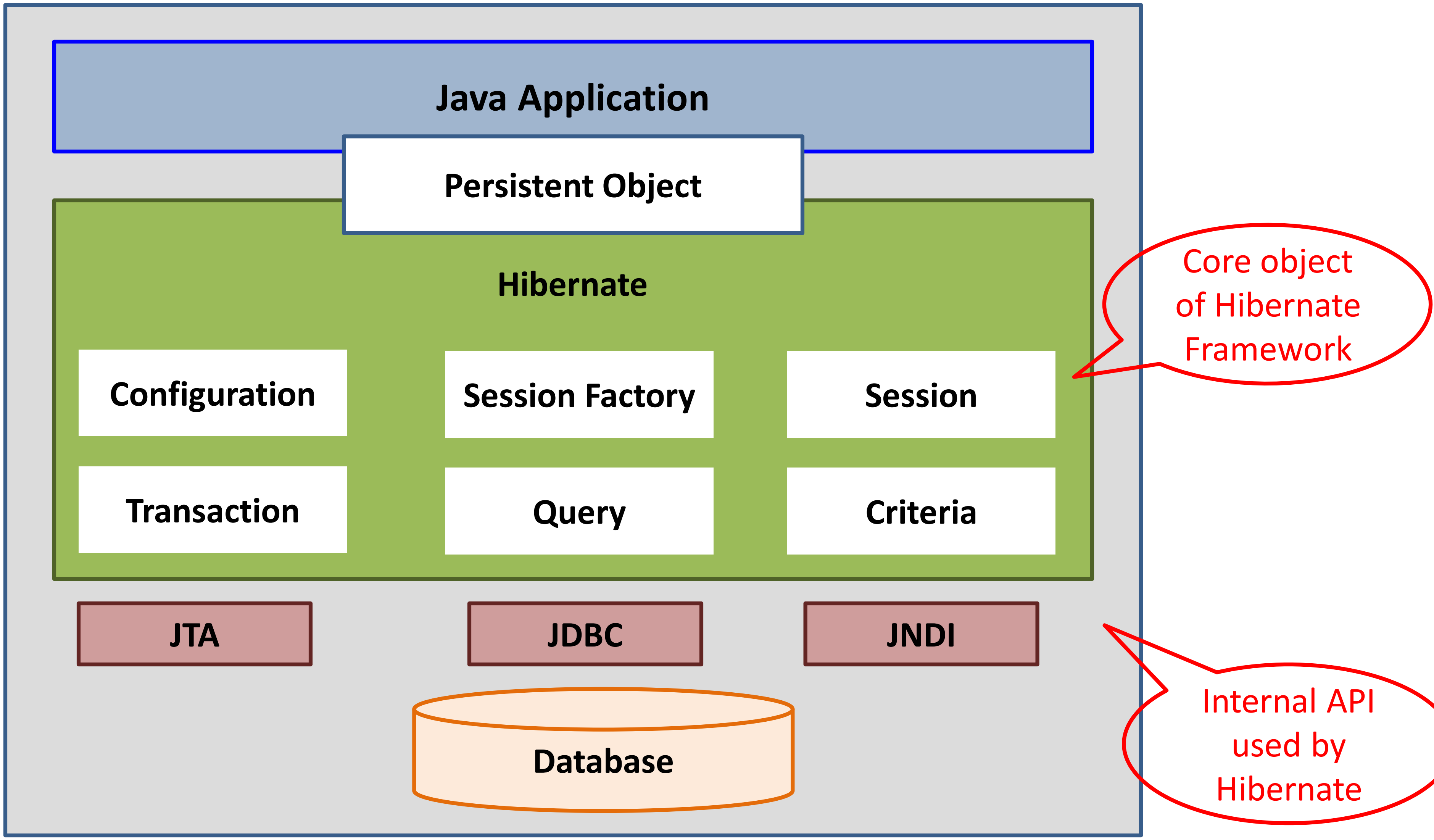

Java Application
Object
ORM
Database
The ORM tool internally uses the JDBC API to interact with the database.

Hibernate Architecture

- There are 4 layers in hibernate architecture
  1. Java application layer
  2. Hibernate framework layer
  3. Backend API layer
  4. Database layer.



Hibernate Architecture



Hibernate Architecture

What do you mean by *Persistence*?

Persistence simply means that we would like our application’s data to outlive the applications process. In Java terms, we would like the state of (some of) our objects to live beyond the scope of the JVM so that the same state is available later.

Hibernate Architecture

Internal API used by Hibernate

1. JDBC (Java Database Connectivity)
2. JTA (Java Transaction API)
3. JNDI (Java Naming Directory Interface)

https://www.youtube.com/watch?v=hfv9ZXUzjhk

Hibernate Architecture

- For creating the first hibernate application, we must know the objects/elements of Hibernate architecture.
- They are as follows:
    1. Configuration
    2. Session factory
    3. Session
    4. Transaction factory
    5. Query
    6. Criteria

Hibernate Architecture

[1] Configuration Object

- The Configuration object is the first Hibernate object you create in any Hibernate application.
- It is usually created only once during application initialization.
- The Configuration object provides two keys components:

  1. **Database Connection:**

     This is handled through one or more configuration files supported by Hibernate. These files are **hibernate.properties** and **hibernate.cfg.xml**.

  2. **Class Mapping Setup**:

     This component creates the connection between the Java classes and database tables.

Hibernate Architecture

[2] SessionFactory Object

- The SessionFactory is a thread safe object and used by all the threads of an application.
- Configuration object is used to create a SessionFactory object which in turn configures Hibernate for the application.
- You would need one SessionFactory object per database using a separate configuration file.
- So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Hibernate Architecture

[3] Session Object

- A Session is used to get a physical connection with a database.
- The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.
- The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed as needed.

Hibernate Architecture

[4] Transaction Object

- A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality.
- Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

Hibernate Architecture

[5] Query Object

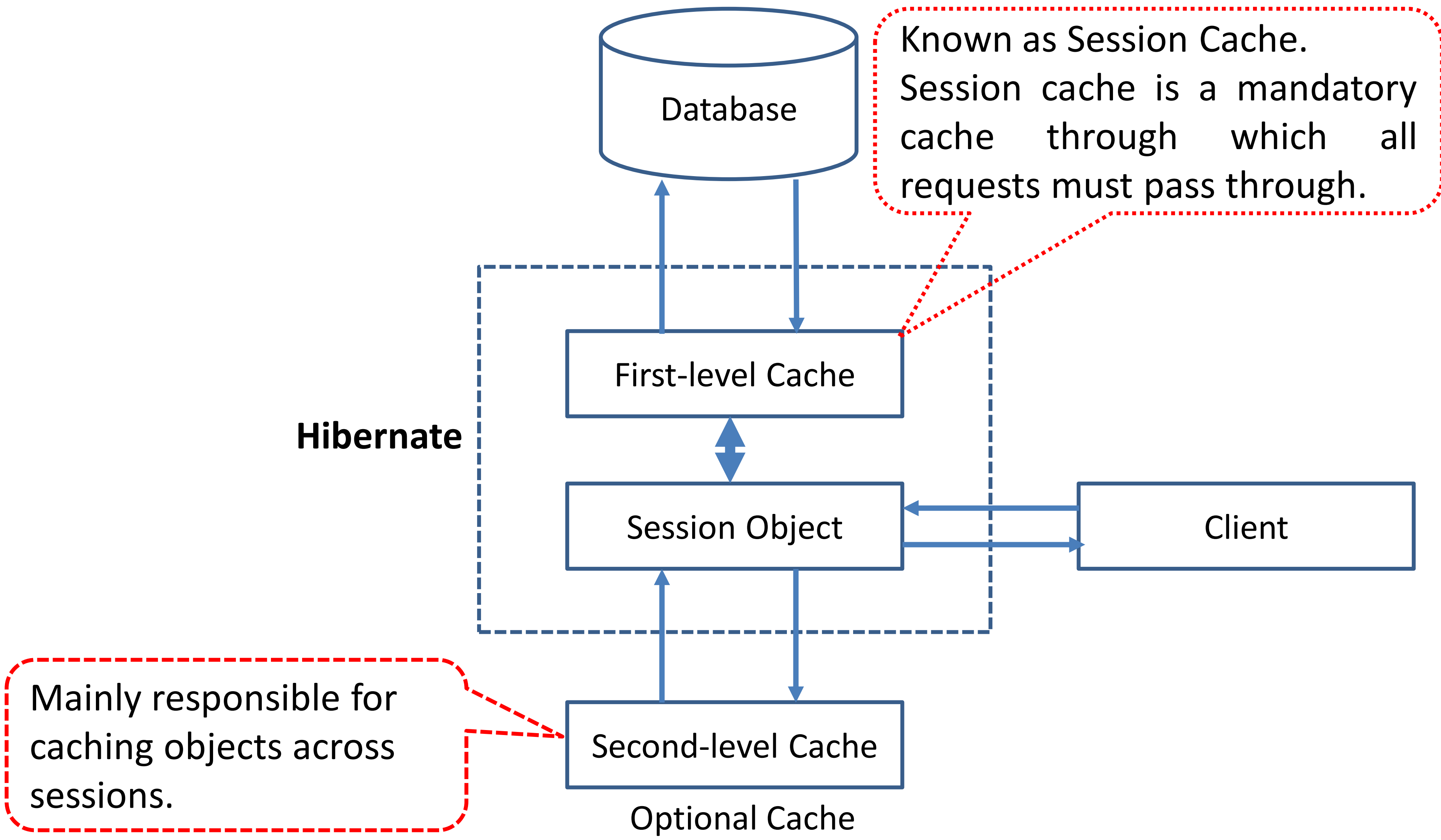
- Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.
- A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Hibernate Architecture

[6] Criteria Object

Criteria objects are used to create and execute object oriented criteria queries to retrieve objects.

Hibernate Cache Architecture



Known as Session Cache. Session cache is a mandatory cache through which all requests must pass through.

Mainly responsible for caching objects across sessions.

Why Cache Architecture?

- Caching is all about application performance optimization.
- It is situated between your application and the database to avoid the number of database hits as many as possible.
- To give a better performance for critical applications.

Hibernate Cache Architecture

First-level cache:

- The first-level cache is the Session cache.
- The Session object keeps an object under its own control before committing it to the database.
- If you issue multiple updates to an object, Hibernate tries to delay doing the update as long as possible to reduce the number of update SQL statements issued.
- If you close the session, all the objects being cached are lost.

Hibernate Cache Architecture

Second-level cache:

- It is responsible for caching objects across sessions.
- Second level cache is an optional cache and first-level cache will always be consulted before any attempt is made to locate an object in the second-level cache.
- Any third-party cache can be used with Hibernate. An **org.hibernate.cache.CacheProvider** interface is provided, which must be implemented to provide Hibernate with a handle to the cache implementation.

Advantages of Hibernate Framework

1. **Open source and Lightweight:** Hibernate framework is open source under the LGPL (GNU Lesser General Public License) license and lightweight.
2. **Fast performance:** The performance of hibernate framework is fast because cache is internally used in hibernate framework.
3. **Database Independent query:** HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

Advantages of Hibernate Framework

4. **Automatic table creation:** Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.
5. **Simplifies complex join:** To fetch data from multiple tables is easy in hibernate framework.
6. **Provides query statistics and database status:** Hibernate supports Query cache and provide statistics about query and database status.

Hibernate Query Language (HQL)

- The Hibernate ORM framework provides its own query language called Hibernate Query Language .
- Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL.

  Therefore, it is database independent query language.

Hibernate Query Language (HQL)

Characteristics of HQL

1. **Similar to SQL**

   HQL's syntax is very similar to standard SQL. If you are familiar with SQL then writing HQL would be pretty easy.

2. **Fully object-oriented:** HQL doesn't use real names of table and columns. It uses class and property names instead. HQL can understand inheritance, polymorphism and association.

3. **Reduces the size of queries**

HQL vs SQL

SELECT QUERY

SQL

ResultSet rs=st.executeQuery("**select * from diet**");

HQL

Query query= session.createQuery("**from diet**");

//here persistent class name is diet

HQL vs SQL

SELECT with WHERE clause

SQL

ResultSet rs=st.executeQuery("**select * from diet where id=301**");

HQL

Query query= session.createQuery("**from diet where id=301** ");

//here persistent class name is diet

HQL vs SQL

UPDATE QUERY

SQL

1. String query = "**update User set name=? where id = ?";**
2. PreparedStatement preparedStmt = conn.prepareStatement(query);
3. preparedStmt.setString (1, “DIET_CE”);
4. preparedStmt.setInt(2, 054);
5. preparedStmt.executeUpdate();

HQL

1. Query q=session.createQuery("**update User set name=:n where id=:i**");
2. q.setParameter("n", "DIET_CE");
3. q.setParameter("i",054);
4. int status=q.executeUpdate();

HQL vs SQL

INSERT QUERY

SQL

String sql = "**INSERT INTO Stock VALUES (100, 'abc')**";

int result = stmt.executeUpdate(sql);

HQL

Query query = session.createQuery("**insert into Stock(stock_code,**

stock_name) select stock_code, stock_name from backup_stock");

int result = query.executeUpdate();

Hibernate Implementations

- ➢Hibernate XML based configuration implementation
- ➢Hibernate Annotation based implementation
- ➢Hibernate Web Based Implementation