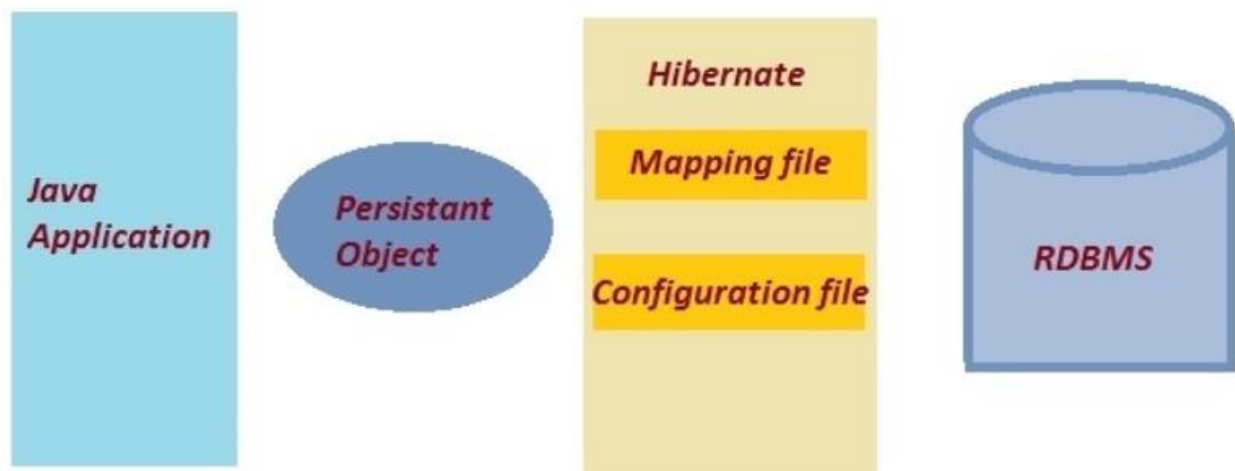


Hibernate

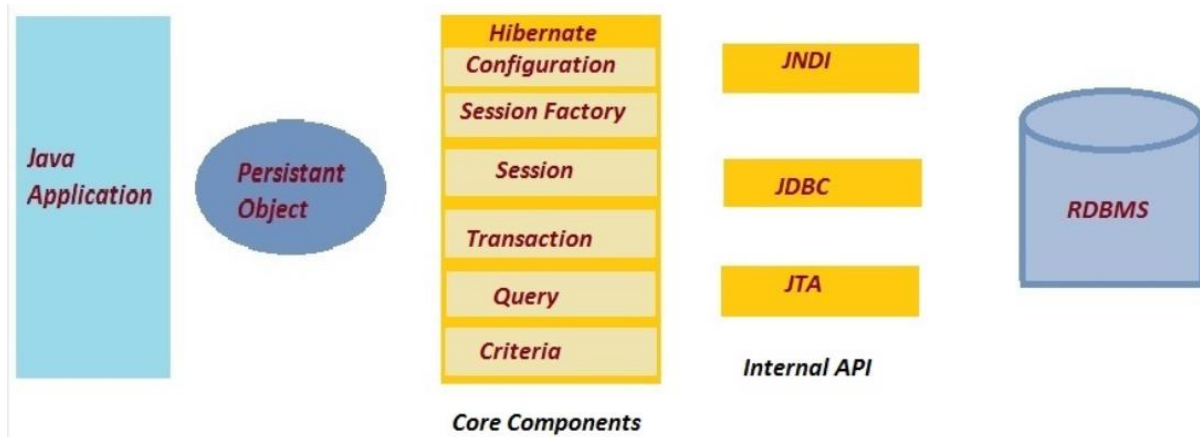
Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

Functionalities supported by Hibernate framework:

- Hibernate framework support **Auto DDL** operations. In JDBC manually we have to create table and declare the data-type for each and every column. But Hibernate can do **DDL operations** for you internally like creation of table, drop a table, alter a table etc.
- Hibernate supports **Auto Primary key generation**. It means in JDBC we have to manually set a primary key for a table. But Hibernate can this task for you.
- Hibernate framework is independent of Database because it supports **HQL (Hibernate Query Language)** which is not specific to any database, whereas JDBC is database dependent.
- In Hibernate, **Exception Handling is not mandatory**, whereas In JDBC exception handling is mandatory.
- Hibernate supports **Cache Memory** whereas JDBC does not support cache memory.
- Hibernate is a **ORM tool** means it support Object relational mapping. Whereas JDBC is not object oriented moreover we are dealing with values means primitive data. In hibernate each record is represented as a Object but in JDBC each record is nothing but a data which is nothing but primitive values.



Core Components of Hibernate architecture:



Configuration object:

The configuration object consists of the configuration file used by Hibernate. It mainly consists of the information about database connection (`hibernate.cfg.xml`) and class mapping (`.hbm.xml`). The configuration object is used to create the object of `SessionFactory`.

SessionFactory object:

The `SessionFactory` object is created from the configuration object. It is a thread-safe object and is used by all threads of the application. It acts as a factory for session objects and a client for `ConnectionProvider`. It maintains the second-level cache of the data.

Session object:

The session object is created from the `SessionFactory` object. It acts as a factory for `Transaction`, `Query`, and `Criteria`. Session objects are not thread-safe. It maintains the first-level cache.

Transaction object:

The transaction object is created from the session object. A transaction object represents an atomic unit of work.

Query object:

The Query object is created from Session object. It uses SQL and HQL to perform database operations.

Criteria object:

The Criteria object is created from the Session object. It is used to define and execute the object oriented criteria queries.

hibernate configuration file mainly contains three types of information:

1. Connection Properties related to the database.
2. Hibernate Properties related to hibernate behaviour.
3. Mapping files entries related to the mapping of a POJO class and a database table.

Hibernate.cfg.xml:

```
<hibernate-configuration>
<session-factory>
  // Connection Properties
  <property name="connection.driver_class">driverClassName</property>
  <property name="connection.url">jdbcConnectionURL</property>
  <property name="connection.user">databaseUsername</property>
  <property name="connection.password">databasePassword</property>

  // Hibernate Properties
  <property name="show_sql">true/false</property>
  <property name="dialect">databaseDialectClass</property>
  <property name="hbm2ddl.auto">like create/update</property>

  // Mapping files entries
  <mapping resource="mappingFile1.xml" />
  <mapping resource="mappingFile2.xml" />

</session-factory>
</hibernate-configuration>
```

NativeSQL:

Native SQL refers to the real SQL for used database. Hibernate provides the facility to use native SQL. We can create a native SQL by createSQLQuery() method of Session interface.

Syntax: Query query = session.createSQLQuery("nativeSQL");

Dialect:

Dialect provides the way to identify the language used by the database for communication.

DB2 dialect in hibernate

org.hibernate.dialect.DB2Dialect

DB2 AS/400 dialect in hibernate

org.hibernate.dialect.DB2400Dialect

DB2 OS390 dialect in hibernate

org.hibernate.dialect.DB2390Dialect

PostgreSQL dialect in hibernate

org.hibernate.dialect.PostgreSQLDialect

MySQL dialect in hibernateL

org.hibernate.dialect.MySQLDialect

MySQL with InnoDB dialect in hibernate

org.hibernate.dialect.MySQLInnoDBDialect


MySQL with MyISAM dialect in hibernate

org.hibernate.dialect.MySQLMyISAMDialect

Oracle 8 dialect in hibernate

org.hibernate.dialect.OracleDialect

Oracle 9i/10g dialect in hibernate

 org.hibernate.dialect.Oracle9Dialect

Sybase dialect in hibernate

 org.hibernate.dialect.SybaseDialect

Sybase Anywhere dialect in hibernate

 org.hibernate.dialect.SybaseAnywhereDialect

Microsoft SQL Server dialect in hibernate

 org.hibernate.dialect.SQLServerDialect

SAP DB dialect in hibernate

 org.hibernate.dialect.SAPDBDialect

Informix dialect in hibernate

 org.hibernate.dialect.InformixDialect

HypersonicSQL dialect in hibernate

 org.hibernate.dialect.HSQLDialect

Ingres dialect in hibernate

 org.hibernate.dialect.IngresDialect

Progress dialect in hibernate

 org.hibernate.dialect.ProgressDialect

Mckoi SQL dialect in hibernate

 org.hibernate.dialect.MckoiDialect

Interbase dialect in hibernate

 org.hibernate.dialect.InterbaseDialect

Pointbase dialect in hibernate

 org.hibernate.dialect.PointbaseDialect

FrontBase dialect in hibernate

 org.hibernate.dialect.FrontbaseDialect

Firebird dialect in hibernate

`org.hibernate.dialect.FirebirdDialect`

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. So it is database independent query language.

Query Interface

It is an object oriented representation of Hibernate Query. The object of Query can be obtained by calling the `createQuery()` method Session interface.

The query interface provides many methods. There is given commonly used methods:

1. **`public int executeUpdate()`** is used to execute the update or delete query.
2. **`public List list()`** returns the result of the relation as a list.
3. **`public Query setFirstResult(int rowno)`** specifies the row number from where record will be retrieved.
4. **`public Query setMaxResult(int rowno)`** specifies the no. of records to be retrieved from the relation (table).
5. **`public Query setParameter(int position, Object value)`** it sets the value to the JDBC style query parameter.
6. **`public Query setParameter(String name, Object value)`** it sets the value to a named query parameter.

Examples:

```
1) Query query=session.createQuery("from Emp");//here persistent class name is Emp  
  
List list=query.list();
```

Example of HQL to get records with pagination:

```
Query query=session.createQuery("from Emp");  
  
query.setFirstResult(5);
```

```
query.setMaxResult(10);
```

```
List list=query.list();//will return the records from 5 to 10th number
```

Example of HQL update query

```
Transaction tx=session.beginTransaction();
```

```
Query q=session.createQuery("update User set name=:n where id=:i");
```

```
q.setParameter("n","Udit Kumar");
```

```
q.setParameter("i",111);
```

```
int status=q.executeUpdate();
```

```
System.out.println(status);
```

```
tx.commit();
```

Example to get minimum salary of employee

```
Query q=session.createQuery("select min(salary) from Emp");
```