Hibernate is ORM(Object-Relational Mapping) tool.

(OOP-Model) (tool) (Relational Model)

JAVA Toplink/Hibernate Mysql/Oracle/..

Class<--(Objects)-----------ORM------------------->DB

> Open Source GUI Lesser General Public License

> Takes care of mapping from Java class to Database(//java to sql)

> provides data query and retrieval facilities

> HQL(Hibernate Query Language) database independent;

> Helps to reduce code compare to JDBC(because of large overheads)

ORM provide solution to handle below mismatches--

-Granularity(Object model which has more classes than corresponding table in db)

-Inheritance(no concept of inheritance in RDBMS)

-Identity

-Association

-Navigation

Advantage of ORM

-HIde details of SQL from OO logic

-Let the business code access object rather than DB tables.

-No need to deals with DB Implementation

-Transaction management and automatic key generation

ORM SOlution :4 Entities

- Language or API to perform CRUD operation

- configurable facility for specifying mapping(using config XML)

Hibernate

-open source persistent framework

-2001 Gavin King

-Provide APIs for storing and retrieval java objects directly to and from db

-if there is any change in DB or in table then only need to change XML file properties

-Doesn't required an application server to operate

-Fast because of catch is internally used in hibernate framework.

.....There are two catch level. First level catch is default

HQL is Object Oriented Query language

-don't need to write database specific query

-Automatic table creation

-no Join operation

-Query statistics and database

-support almost all the major RDBMS Language

Hibernate

Core Classes

* Configuration
* Session Factory
* Session
* Transaction
* Query
* Criteria

Hibernate uses JDBC, JTA(Java Transaction API),JNDI(Java Naming and Directory Interface)

* **Configuration Objects-**

First Hibernate Object you create

Only once it is created

Provide two key component

1. Database connection- hibernate.properties(standard java properties) and hibernate.cfg.xml(XML file)
2. Class Mapping setup

* **Session Factory Object**

Configuration object is used to create Session Factory Object

It is a heavy weight , Thread safe object.

One object for one database

* **Session Object**

Used to get a phy. Connection with database

Should not be kept open for long.

* **Transaction Object**
* **Query Object**

Use HQL/SQL string to retrieve data from database.

JDK

Eclipse IDE

Download Hibernate

Download Hibernate Prerequisites-

* Dom4j XML Paser
* Xalan
* Xerces
* Cglib
* Log4j
* Commons
* SLF4J

|  |  |
| --- | --- |
|  | **Properties & Description** |
| 1 | **hibernate.dialect**  This property makes Hibernate generate the appropriate SQL for the chosen database. |
| 2 | **hibernate.connection.driver\_class**  The JDBC driver class. |
| 3 | **hibernate.connection.url**  The JDBC URL to the database instance. |
| 4 | **hibernate.connection.username**  The database username. |
| 5 | **hibernate.connection.password**  The database password. |
| 6 | **hibernate.connection.pool\_size**  Limits the number of connections waiting in the Hibernate database connection pool. |
| 7 | **hibernate.connection.autocommit**  Allows autocommit mode to be used for the JDBC connection. |

**Mysql Dialect**

**Org.hibernate.dialect.MySQLDilect**

**Session Object**

Instances may exist in one of the following three states

* Transient-which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
* Persistent-persistent instance has a representation in the database, an identifier value and is associated with a Session.
* Detached-once we close the Hibernate Session, the persistent instance will become a detached instance

**Persistent Class**

Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate

Hibernate works best if these classes follow some simple rules, also known as the **Plain Old Java Object** (POJO) programming model.

All the properties should be private

Have getters and setters

<ClassName>.hbm.xml

Cl

<generator> within id is used to automatically generate primary key value.

* Assigned -default generator application assign id
* Increment- generates unique id automatically only if no process insert data into the table.
* Native-uses identity, sequence or hilo depending on the data vendor
* Identity – responsibility of db to generate unique id

<property> is used to map java class property in a table

Mapping type….ANSI SQL Type:

Integer……………. INTEGER

Long.........……… BIGINT

String…………….. VARCHAR

Class……………... VARCHAR

Date……………… DATE

BLOB……………..BYTE

Create hibernate Application

1. Create java project
2. Add jar file for hibernate
3. Create persistent class
4. Create mapping file for persistent class
5. Create configuration file
6. Create the class that retrieve or store the persistent object import org.hibernate.\*;
7. Run the application

**Hbm2ddl.auto**

Automatically validates or export schema DDL to the database when SessionFactory creates

* Validate
* Update
* Create
* Create-drop

**HQL**

-Object oriented query language

Advantages

1. Db independent
2. Support polymorphic queries

Query Object is used to create HQL string to retrieve data from database

Methods-

1. createQuery()
2. Public int executeUpdates()
3. Public List list()
4. Public Query setFirstResult(int rowno)
5. Public Query setMaxResult(int rowno)
6. Public Query setParameter(int position,Object value)
7. Public Query setParameter(String name,Object value)

**FROM Clause** fetch all the columns

String hql=”**From** Employee”;

Query query = session.createQuery(hql);

List result = query.list();

**AS Clause**

String hql=”From Employee **AS** Emp”;

Query query = session.createQuery(hql);

List result = query.list();

**SELECT Clause** fetch specific column

String hql=”SELECT E.firstName FROM Employee **AS** E”;

Where E.firstName is property of employee object

**WHERE Clause**

**ORDER BY**

String hql=”DELETE FROM Employee E WHERE E.id = **:employee\_id**”;

Query query = session.createQuery(hql);

Query.setParameter(“employee\_id”,10);

Int result =query.executeUpdate();

SYsout(“Row Affected” + result)

**GROUP BY**

**Named Parameters**

This makes HQL query that accept input from the user easy and you don’t have to defend against SQL injection attacks

String hql=”FROM Employee E WHERE E.id = **:employee\_id**”;

Query query = session.createQuery(hql);

Query.setParameter(“employee\_id”,10);

**Criteria API**

Apply filtration rules and logical condition

Criteria cr =session.createCriteria(User.class);

cr.add(Restrictions.eq(“salary”,2000));

-.gt(“salary”,2000));

-.lt(“salary”,2000)); .between(“salary”,2000,4000)); .like(“name”,”R%”));

List result = cr.list();