Yadab Raj Ojha

Sr. Java Developer / Lecture

Email: yadabrajojha@gmail.com

Blog: http://yro-tech.blogspot.com/

Java Tutorial: https://github.com/yrojha4ever/JavaStud

LinkedIn: https://www.linkedin.com/in/yrojha

Twitter: https://twitter.com/yrojha4ever

Website: http://javaenvagilist.com/

Hibernate:

- Introduction.
- Why ORM?
- What is ORM?
- Java ORM Frameworks.
- Advantages of Hibernate ORM Framework
- Hibernate Architecture
- Configuration Object
- Session factory and Session
- Transaction
- Query and Criteria Object
- Hibernate Properties/hibernate.cfg.xml
- Create first Hibernate Project using Maven.
- Queries: Insert/Update/Delete/Get
- Hibernate Criteria: (List, Restrictions and UniqueResult)
- Relationships(OneToOne, OneToMany, ManyToOne and ManyToMany)
- Auto Create Database using Hibernate hbm2ddl.auto=create properties.
- Example to add multiple records which contain all Hibernate Table Relationships.

Hibernate Tutorial

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 one of the most popular Object/Relational Mapping (ORM) framework in the Java world. It allows developers to map the object structures of normal Java classes to the relational structure of a database. With the help of an ORM framework the work to store data from object instances in memory to a persistent data store and load them back into the same object structure becomes significantly easier.

Hibernate is also a JPA provider, that means it implements the Java Persistence API (JPA). JPA is a vendor independent specification for mapping Java objects to the tables of relational databases.

Why Object Relational Mapping (ORM)?

When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.

What is ORM?

ORM stands for **O**bject-**R**elational **M**apping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc. An ORM system has following advantages over plain JDBC

S.N.	Advantages
1	Lets business code access objects rather than DB tables.

2	Hides details of SQL queries from OO logic.
3	Based on JDBC 'under the hood'
4	No need to deal with the database implementation.
5	Entities based on business concepts rather than database structure.
6	Transaction management and automatic key generation.
7	Fast development of application.

Java ORM Frameworks:

There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

- Enterprise JavaBeans Entity Beans
- Java Data Objects
- Castor
- TopLink
- Spring DAO
- Hibernate
- And many more

Advantages of Hibernate Framework

There are many advantages of Hibernate Framework. They are as follows:

1) Opensource and Lightweight: Hibernate framework is opensource under the LGPL license and lightweight.

2) Fast performance: The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled bydefault.

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.

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.

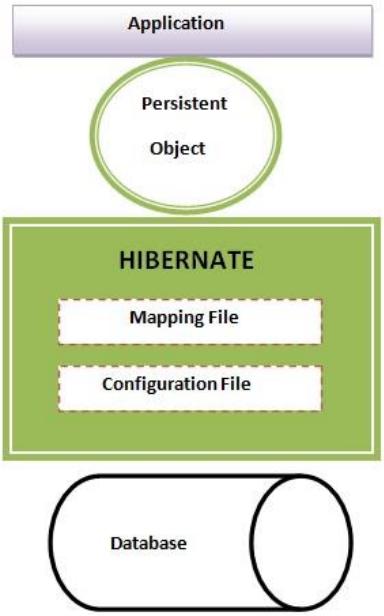
5) Simplifies complex join: To fetch data form multiple tables is easy in hibernate framework.

6) Provides query statistics and database status: Hibernate supports Query cache and provide statistics about query and database status.

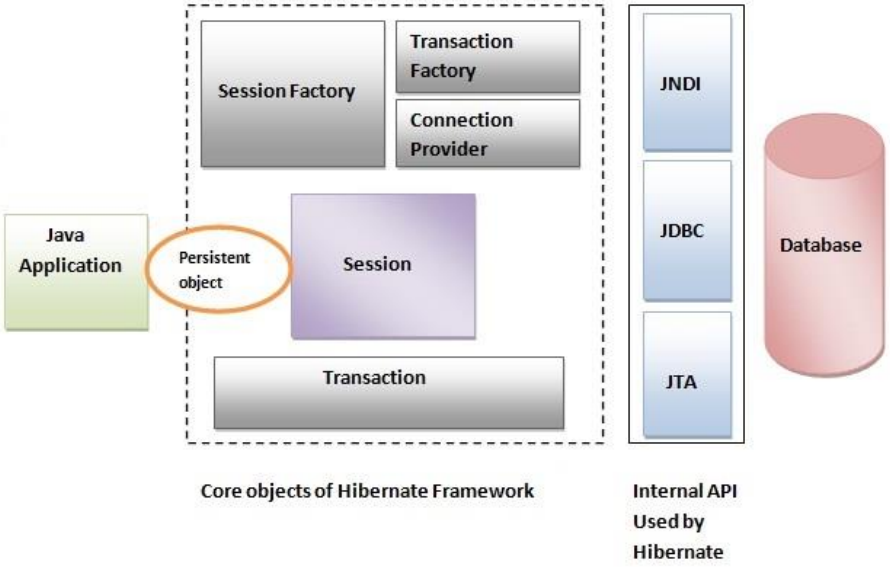
Hibernate Architecture

The Hibernate architecture includes many objects persistent object, session factory, transaction factory, connection factory, session, transaction etc.

There are 4 layers in hibernate architecture java application layer, hibernate framework layer, backhand api layer and database layer.Let's see the diagram of hibernate architecture:



This is the high level architecture of Hibernate with mapping file and configuration file.



Hibernate framework uses many objects session factory, session, transaction etc. alongwith existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface).

Configuration Object:

The Configuration object is the first Hibernate object you create in any Hibernate application and usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate. The Configuration object provides two keys components:

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

- **Class Mapping Setup**

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

SessionFactory Object:

Configuration object is used to create a SessionFactory object which inturn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

```
SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
```

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. Persistent objects are saved and retrieved through a Session object.

```
Session session = sessionFactory.openSession();

session.close();
```

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).

```
session.beginTransaction();

....

session.getTransaction().commit();
```

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.

Criteria Object:

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

Hibernate Properties:

hibernate.cfg.xml

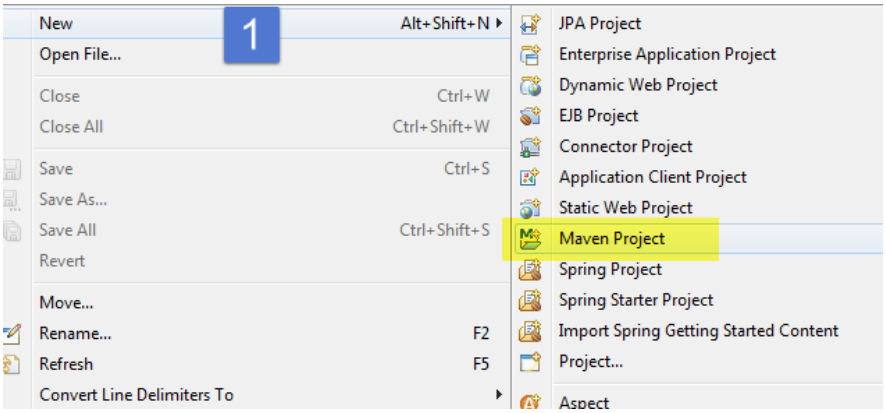
Hibernate Properties

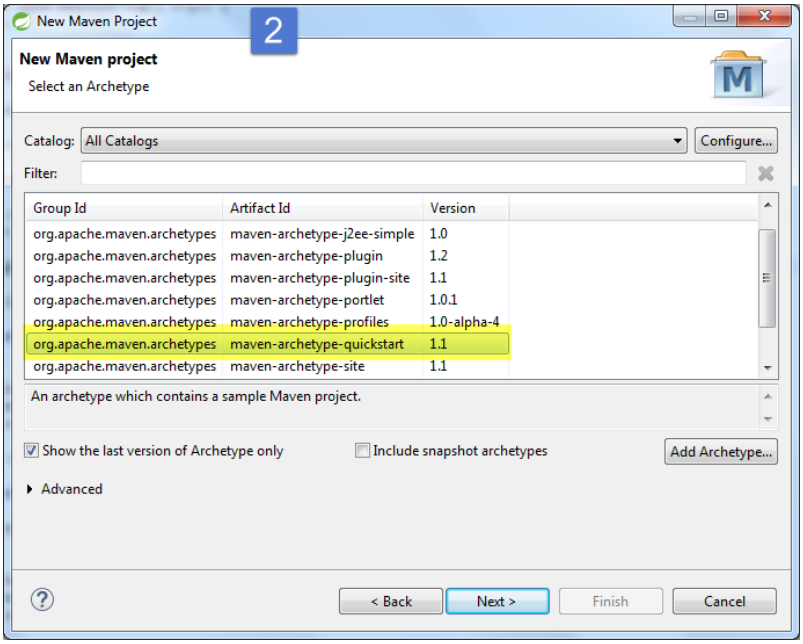
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/test</property>
        <property name="connection.username">root</property>
        <property name="connection.password" />

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache -->
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">validate</property>

        <!-- Mapping to Entity -->
        <mapping class="yrojha.hannotation.Employee" />
    </session-factory>

</hibernate-configuration>
```

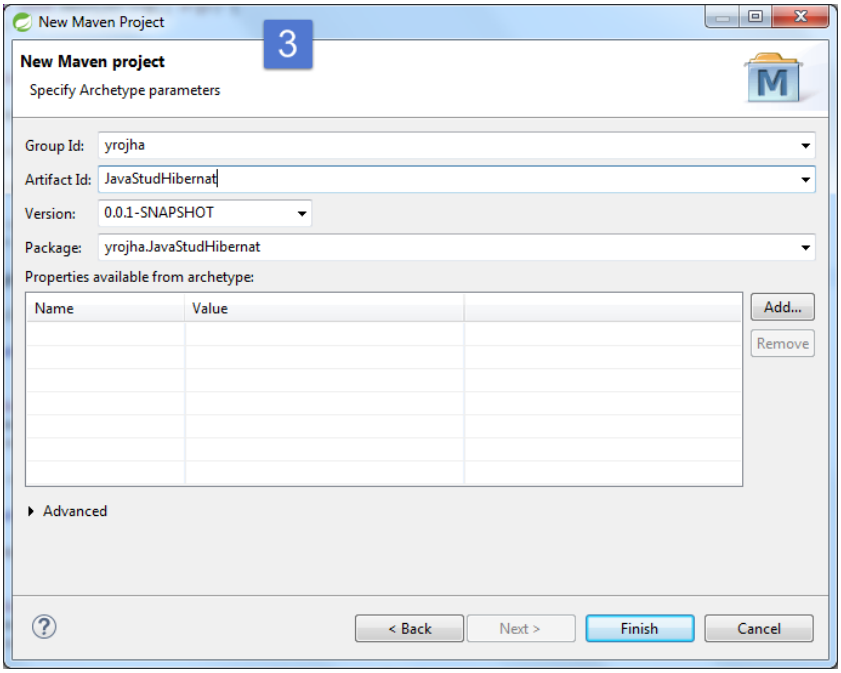
Employee.java

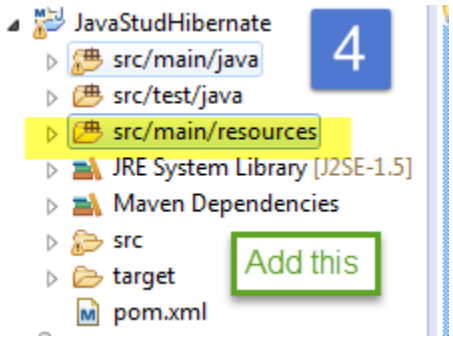
```java
package yrojha.hannotation;

import java.sql.Date;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue
    private Long id;

    @Column(name = "firstname")
    private String firstname;

    @Column(name = "lastname")
    private String lastname;

    @Column(name = "birth_date")
    private Date birthDate;

    @Column(name = "cell_phone")
    private String cellphone;

    public Employee() {

    }

    public Employee(String firstname
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthDate = birthdate;
        this.cellphone = phone;
    }

    public Long getId() {
        return id;
```

*Create the Hibernate Application using Maven:




New Maven Project
2
New Maven project
Select an Archetype
Catalog: All Catalogs
Configure...
Filter:
Group Id
Artifact Id
Version
org.apache.maven.archetypes maven-archetype-j2ee-simple 1.0
org.apache.maven.archetypes maven-archetype-plugin 1.2
org.apache.maven.archetypes maven-archetype-plugin-site 1.1
org.apache.maven.archetypes maven-archetype-portlet 1.0.1
org.apache.maven.archetypes maven-archetype-profiles 1.0-alpha-4
org.apache.maven.archetypes maven-archetype-quickstart 1.1
org.apache.maven.archetypes maven-archetype-site 1.1
An archetype which contains a sample Maven project.
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Finish
Cancel


New Maven Project
3
New Maven project
Specify Archetype parameters
Group Id: yrojha
Artifact Id: JavaStudHibernat
Version: 0.0.1-SNAPSHOT
Package: yrojha.JavaStudHibernat
Properties available from archetype:
Name
Value
Add...
Remove
Advanced
< Back
Next >
Finish
Cancel


JavaStudHibernate
4
src/main/java
src/test/java
src/main/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml
Add this

Project Explorer

- com.yro.hibernate
- demo
- excel
- hannotation
- helloservlet
- JavaStud [JavaStud master]
- JavaStudHibernate
  - src/main/java
  - src/test/java
  - src/main/resources
    - hibernate.cfg.xml
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- Servers
- springmvcapp
- StudManagProj [StudManagPr
- StudWebApp
- ThreadsExamples [ThreadsExam

JavaStudHibernate/pom.xml

5

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>yrojha</groupId>
  <artifactId>JavaStudHibernate</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>JavaStudHibernate</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
   <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>4.0.0.Final</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.10</version>
    </dependency>
  </dependencies>

</project>
```

plorer

- yro.hibernate
- otation
- ervlet
- aStud [JavaStud master]
- tudHibernate
- c/main/java
- yrojha.JavaStudHibernate
  - table
  - Address.java
  - Employee.java
  - App.java
  - EmployeeDao.java
- c/test/java
- c/main/resources
- hibernate.cfg.xml
- E System Library [J2SE-1.5]
- aven Dependencies
- c
- rget
- om.xml
- rs
- gmvcapp
- dManagProj [StudManagPr
- VebApp
- dsExamples [ThreadsExam

Employee.java

6

```java
package yrojha.JavaStudHibernate.table;

import java.sql.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue
    private Long id;

    @Column(name = "firstname")
    private String firstname;

    @Column(name = "lastname")
    private String lastname;

    @Column(name = "birth_date")
    private Date birthDate;

    @Column(name = "cell_phone")
    private String cellphone;

    public Employee() {

    }

    public Employee(String firstname, String lastname, Date birthdate, String phone) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.birthDate = birthdate;
        this.cellphone = phone;
    }

    Getter..Setter..
```

id	firstname	lastname	birth_date	cell_phone
* (Auto)	(NULL)	(NULL)	(NULL)	(NULL)

Column Name	Data Type	Length	Default	PK?	Not Null?	Unsigned?	Auto In
id	bigint	10		☑	☑	☐	☑
firstname	varchar	50		☐	☐	☐	☐
lastname	varchar	50		☐	☐	☐	☐
birth_date	date			☐	☐	☐	☐
cell_phone	varchar	15		☐	☐	☐	☐

hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/studentdb</property>
        <property name="connection.username">root</property>
        <property name="connection.password" />

        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>

        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Enable Hibernate's automatic session context management -->
        <property name="current_session_context_class">thread</property>

        <!-- Disable the second-level cache -->
        <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">validate</property>

        <!-- Mapping to Entity -->
        <mapping class="yrojha.JavaStudHibernate.table.Employee" />
    </session-factory>

</hibernate-configuration>
```

Queries:

Insert: (save)

```
EmployeeDao.java
19    private static void insert() {
20
21        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
22
23        Session session = sessionFactory.openSession();
24
25        session.beginTransaction();
26
27        Employee emp = new Employee("Rajan", "Shrestha", new Date(System.currentTimeMillis()), "977-8941456790");
28        Long id = (Long) session.save(emp);
29
30        session.getTransaction().commit();
31
32        session.close();
33
34        System.out.println("Data Inserted. Id: " + id);
35    }
```

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Progress  Package Explorer
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 10:54:38 PM)
Jan 01, 2016 10:54:40 PM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000037: Columns: [firstname, birth_date, id, lastname, cell_phone]
Hibernate: insert into employee (birth_date, cell_phone, firstname, lastname) values (?, ?, ?, ?)
Data Inserted. Id: 11
```

Get:

```
EmployeeDao.java
39    private static Employee get() {
40
41        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
42
43        Session session = sessionFactory.openSession();
44
45        Employee emp = (Employee) session.get(Employee.class, 11L);
46
47        System.out.println(emp);
48
49        return emp;
50    }
```

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Progress  Package Explorer
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 10:57:37 PM)
INFO: HHH000037: Columns: [firstname, birth_date, id, lastname, cell_phone]
Hibernate: select employee0_.id as id0_0_, employee0_.birth_date as birth2_0_0_, employee0_.cell_phone as c
Id: 11, firstname: Rajan, lastname: Shrestha, birthDate: 2016-01-01, cellphone: 977-8941456790
```

Update:

EmployeeDao.java

```java
    public static void update() {
        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        Employee emp = (Employee) session.get(Employee.class, 11L);
        emp.setLastname("Joshi");

        session.update(emp);

        session.getTransaction().commit();
        session.close();
        System.out.println("Updated Employee Info: " + emp);
    }
```

Console | Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Package Explorer

```
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 10:59:20 PM)
INFO: HHH000261: Table found: studentdb.employee
Jan 01, 2016 10:59:22 PM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000037: Columns: [firstname, birth_date, id, lastname, cell_phone]
Hibernate: select employee0_.id as id0_0_, employee0_.birth_date as birth2_0_0_, employee0_.cell_phone as cell3_0_0_,
Hibernate: update employee set birth_date=?, cell_phone=?, firstname=?, lastname=? where id=?
Updated Employee Info: Id: 11, firstname: Rajan, lastname: Joshi, birthDate: 2016-01-01, cellphone: 977-8941456790
```

id	firstname	lastname	birth_date	cell_phone
11	Rajan	Joshi	2016-01-01	977-8941456790

Delete:

EmployeeDao.java

```java
    private static void delete() {
        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        session.beginTransaction();

        Employee employee = (Employee) session.get(Employee.class, 11L);
        session.delete(employee);

        session.getTransaction().commit();
        session.close();
    }
```

Console | Markers | Properties | Servers | Data Source Explorer | Snippets | Progress | Package Explorer

```
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 11:03:35 PM)
INFO: HHH000037: Columns: [firstname, birth_date, id, lastname, cell_phone]
Hibernate: select employee0_.id as id0_0_, employee0_.birth_date as birth2_0_0_, employee0_.cell_phone
Hibernate: delete from employee where id=?
```

Hibernate Criteria:

List:

```
EmployeeDao.java
15    public static void main(String[] args) {
16        for (Employee emp : getAllEmployees()) {
17            System.out.println(emp);
18        }
19    }
20
21    private static List<Employee> getAllEmployees() {
22        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
23        Session session = sessionFactory.openSession();
24
25        Criteria criteria = session.createCriteria(Employee.class);
26        List<Employee> empList = (List<Employee>) criteria.list();
27
28        return empList;
29    }
```

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Progress  Package Explorer
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 11:13:32 PM)
INFO: HHH000037: Columns: [firstname, birth_date, id, lastname, cell_phone]
Hibernate: select this_.id as id0_0_, this_.birth_date as birth2_0_0_, this_.cell_phone as cell3_0_0_, this_.fir
Id: 12, firstname: Suraj, lastname: Shrestha, birthDate: 2001-01-19, cellphone: 977-8941456790
Id: 13, firstname: Rajan, lastname: Budhathoki, birthDate: 2010-01-10, cellphone: 977-8941986423
Id: 14, firstname: Saurav, lastname: Joshi, birthDate: 1996-01-23, cellphone: 977-9806142356
```

Restrictions (where clause), UniqueResult:

```
20    private static Employee getEmp() {
21        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
22        Session session = sessionFactory.openSession();
23
24        Criteria criteria = session.createCriteria(Employee.class);
25        criteria.add(Restrictions.eq("id", 13L));
26
27        Employee emp = (Employee) criteria.uniqueResult();
28
29        System.out.println(emp);
30        return emp;
31    }
```

```
Console  Markers  Properties  Servers  Data Source Explorer  Snippets  Progress  Package Explorer
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Jan 1, 2016, 11:26:42 PM)
Hibernate: select this_.id as id0_0_, this_.birth_date as birth2_0_0_, this_.cell_phone as cell3_0_0_, th
Id: 13, firstname: Rajan, lastname: Budhathoki, birthDate: 2010-01-10, cellphone: 977-8941986423
```

Relationship:

- **One to one**: This denotes a simple relationship in which one entity of type A belongs exactly to one entity of type B.

- **Many to one/One To Many**: As the name indicates, this relationship encompasses the case that an entity of type A has many child entities of type B.

- **Many to many**: In this case there can be many entities of type A that belong to many entities of type B.

One to One Relationship:

We can perform one to one mapping in hibernate by two ways:

- By many-to-one element
- By one-to-one element

Employee.java | EmployeeDao....

```java
@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue
    private Long id;

    @Column(name = "firstname")
    private String firstname;

    @Column(name = "lastname")
    private String lastname;

    @Column(name = "birth_date")
    private Date birthDate;

    @Column(name = "cell_phone")
    private String cellphone;

    // One To One Mapping. Employee has a Add
    @OneToOne
    @JoinColumn(name = "address_id")
    private Address address;

    public Employee() {
```

Address.java

```java
package yrojha.JavaStudHibern

import javax.persistence.Colu

@Entity
@Table(name = "address")
public class Address {

    @Id
    @GeneratedValue
    private Long id;

    @Column(name = "city")
    private String city;

    @Column(name = "state")
    private String state;

    @Column(name = "zip")
    private String zip;

    @Column(name = "country")
    private String country;

    public Address() {

    }
```

```
Employee.java   EmployeeDao.java
17    public static void main(String[] args) {
18        oneToOneMapping();
19    }
20
21    private static Employee oneToOneMapping() {
22        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
23        Session session = sessionFactory.openSession();
24        session.beginTransaction();
25
26        Address address = (Address) session.get(Address.class, 1L);
27        Employee emp = new Employee("Anjan", "Pant", new Date(System.currentTimeMillis()), "977-9841661234", address);
28
29        session.merge(emp);        OR        session.save(emp);
30
31        session.getTransaction().commit();
32        session.close();
33
34        return emp;
35    }
```

16	Anjan	Pant	2016-01-02	977-9841661234	1

1	Kathmandu	Bagmati	0977	Nepal

```
Console   Tasks
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Jan 2, 2016, 4:38:29 PM)
Jan 02, 2016 4:38:30 PM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000037: Columns: [cell_phone, id, birth_date, address_id, lastname, firstname]
Hibernate: select address0_.id as id1_0_, address0_.city as city1_0_, address0_.country as country1_0_, address0_.state as state
Hibernate: insert into employee (address_id, birth_date, cell_phone, firstname, lastname) values (?, ?, ?, ?, ?)
```

Section B: Relationships and auto create tables.

NOTE: hbm2ddl.auto= create create all the table automatically.

```
hibernate.cfg.xml   EmployeeDao.java   AddressDao.java
24          <!-- Disable the second-level cache -->
25          <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
26
27          <!-- Echo all executed SQL to stdout -->
28          <property name="show_sql">true</property>
29          <property name="hbm2ddl.auto">create</property> <!-- validate, create,update -->
30
31          <!-- Mapping to Entity -->
32          <mapping class="yrojha.JavaStudHibernate.table.Employee" />
33          <mapping class="yrojha.JavaStudHibernate.table.Address" />
34          <mapping class="yrojha.JavaStudHibernate.table.PhoneNo" />
35          <mapping class="yrojha.JavaStudHibernate.table.Department" />
36      </session-factory>
37
38  </hibernate-configuration>
```

create param create all the table automatically.

```
Design   Source
Console   Markers   Progress
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Jan 2, 2016, 6:03:50 PM)
Hibernate: alter table employee drop foreign key FK4722E6AE5880F8AB
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: HHH000389: Unsuccessful: alter table employee drop foreign key FK4722E6AE5880F8AB
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: Table 'hibernatedb.employee' doesn't exist
Hibernate: alter table employee_dept drop foreign key FK8711CD36A01A5796
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: HHH000389: Unsuccessful: alter table employee_dept drop foreign key FK8711CD36A01A5796
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: Table 'hibernatedb.employee_dept' doesn't exist
Hibernate: alter table employee_dept drop foreign key FK8711CD364C05AD4F
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: HHH000389: Unsuccessful: alter table employee_dept drop foreign key FK8711CD364C05AD4F
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: Table 'hibernatedb.employee_dept' doesn't exist
Hibernate: alter table phone_no drop foreign key FKB3B31632CFF3BDA9
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: HHH000389: Unsuccessful: alter table phone_no drop foreign key FKB3B31632CFF3BDA9
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport perform
ERROR: Table 'hibernatedb.phone_no' doesn't exist
Hibernate: drop table if exists address
Hibernate: drop table if exists dept
Hibernate: drop table if exists employee
Hibernate: drop table if exists employee_dept
Hibernate: drop table if exists phone_no
Hibernate: create table address (id bigint not null auto_increment, city varchar(255), country varchar(255), state varchar(255), zip varchar(255), primary key (id))
Hibernate: create table dept (id bigint not null auto_increment, name varchar(255), primary key (id))
Hibernate: create table employee (id bigint not null auto_increment, birth_date date, cell_phone varchar(255), firstname varchar(255), lastname varchar(255), address
Hibernate: create table employee_dept (emp_id bigint not null, dept_id bigint not null)
Hibernate: create table phone_no (id bigint not null auto_increment, number varchar(255), employee_id bigint, primary key (id))
Hibernate: alter table employee add index FK4722E6AE5880F8AB (address_id), add constraint FK4722E6AE5880F8AB foreign key (address_id) references address (id)
Hibernate: alter table employee_dept add index FK8711CD36A01A5796 (dept_id), add constraint FK8711CD36A01A5796 foreign key (dept_id) references dept (id)
Hibernate: alter table employee_dept add index FK8711CD364C05AD4F (emp_id), add constraint FK8711CD364C05AD4F foreign key (emp_id) references employee (id)
Hibernate: alter table phone_no add index FKB3B31632CFF3BDA9 (employee_id), add constraint FKB3B31632CFF3BDA9 foreign key (employee_id) references employee (id)
Jan 02, 2016 6:03:52 PM org.hibernate.tool.hbm2ddl.SchemaExport execute
INFO: HHH000230: Schema export complete
Hibernate: insert into employee (address_id, birth_date, cell_phone, firstname, lastname) values (?, ?, ?, ?, ?)
Data Inserted. Id: 1
```

```
hibernatedb
Tables
address
  Columns
    id, bigint(20)
    city, varchar(255), Nullable
    country, varchar(255), Nullable
    state, varchar(255), Nullable
    zip, varchar(255), Nullable
  Indexes
    PRIMARY (id)
dept
  Columns
    id, bigint(20)
    name, varchar(255), Nullable
  Indexes
    PRIMARY (id)
employee
  Columns
    id, bigint(20)
    birth_date, date, Nullable
    cell_phone, varchar(255), Nullable
    firstname, varchar(255), Nullable
    lastname, varchar(255), Nullable
    address_id, bigint(20), Nullable
  Indexes
    PRIMARY (id)
    FK4722E6AE5880F8AB (address_id)
employee_dept
  Columns
    emp_id, bigint(20)
    dept_id, bigint(20)
  Indexes
    FK8711CD36A01A5796 (dept_id)
    FK8711CD364C05AD4F (emp_id)
phone_no
  Columns
    id, bigint(20)
    number, varchar(255), Nullable
    employee_id, bigint(20), Nullable
  Indexes
    PRIMARY (id)
    FKB3B31632CFF3BDA9 (employee_id)
```

After creating above tables in ‘hibernatedb’. We can insert data in all the table which have relation with employee as shown in below.

Steps:

1. Create Employee Object.
2. Insert data in Address table.
3. Insert Data in Department Table
4. Save Phone number 1 and set Employee object inside it.
5. Save Phone number 2 and set Employee Object inside it.
6. Set Address, Department and Phone numbers in employee.
7. SaveOrUpdate() it will fire all the query required(including employee_dept table) as shown in console log.

EmployeeDao.java | Employee.java | PhoneNo.java | AddressDao.java | DepartmentDao.java | hibernate.cfg.xml

```java
20      public static void main(String[] args) {
21          insertEmployeeAddressPhoneDept();
22      }
23      private static void insertEmployeeAddressPhoneDept() {
24          SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
25          Session session = sessionFactory.openSession();
26          session.beginTransaction();
27
28          Employee employee = new Employee("Bibek", "Shrestha", new Date(System.currentTimeMillis()));
29
30          Address address = new Address("Kathmandu", "Bagmati", "0977", "Nepal");
31          session.save(address);
32
33          Department department = new Department("IT");
34          session.save(department);
35
36          PhoneNo phoneNo1 = new PhoneNo("977-9841665543");
37          session.save(phoneNo1);
38          phoneNo1.setEmployee(employee);
39
40          PhoneNo phoneNo2 = new PhoneNo("977-9841431122");
41          session.save(phoneNo2);
42          phoneNo2.setEmployee(employee);
43
44          employee.setAddress(address);
45          employee.setDepartments(Arrays.asList(department));
46          employee.setPhoneNos(Arrays.asList(phoneNo1, phoneNo2));
47
48          session.saveOrUpdate(employee);
49
50          session.getTransaction().commit();
51          session.close();
52      }
```

employee

id	birth_date	cell_phone	firstname	lastname	address_id
7	2016-01-03	(NULL)	Bibek	Shrestha	3

address

id	city	country	state	zip
3	Kathmandu	Nepal	Bagmati	0977

phone_no

id	number	employee_id
7	977-9841665543	7
8	977-9841431122	7

employee_dept

emp_id	dept_id
7	3

dept

id	name
3	IT

Console | Markers | Progress

```
<terminated> EmployeeDao [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Jan 3, 2016, 12:54:20 PM)
INFO: HHH000037: Columns: [id, number, employee_id]
Hibernate: insert into address (city, country, state, zip) values (?, ?, ?, ?)
Hibernate: insert into dept (name) values (?)
Hibernate: insert into phone_no (employee_id, number) values (?, ?)
Hibernate: insert into phone_no (employee_id, number) values (?, ?)
Hibernate: insert into employee (address_id, birth_date, firstname, lastname) values (?, ?, ?, ?)
Hibernate: update phone_no set employee_id=?, number=? where id=?
Hibernate: update phone_no set employee_id=?, number=? where id=?
Hibernate: insert into employee_dept (emp_id, dept_id) values (?, ?)
```