Hibernate Architecture

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

The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand api layer
* 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 such as 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).

## **Elements of Hibernate Architecture**

|  |
| --- |
| For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows: |

#### **SessionFactory**

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

#### **Session**

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

#### **Transaction**

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

#### **ConnectionProvider**

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

#### **TransactionFactory**

It is a factory of Transaction. It is optional.

First Hibernate Example without IDE

Here, we are going to create the first hibernate application without IDE. For creating the first hibernate application, we need to follow the following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object
5. Load the jar file
6. Run the first hibernate application by using command prompt

Let's create the simple Persistent class:

#### **Employee.java**

**package** com.javatpoint.mypackage;

**public** **class** Employee {

**private** **int** id;

**private** String firstName,lastName;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

### **2) Create the mapping file for Persistent class**

The mapping file name conventionally, should be class\_name.hbm.xml. There are many elements of the mapping file.

#### **employee.hbm.xml**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-mapping PUBLIC

 "-//Hibernate/Hibernate Mapping DTD 5.3//EN"

 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

 <hibernate-mapping>

  <**class** name="com.javatpoint.mypackage.Employee" table="emp1000">

    <id name="id">

     <generator **class**="assigned"></generator>

    </id>

    <property name="firstName"></property>

    <property name="lastName"></property>

  </**class**>

 </hibernate-mapping>

### **3) Create the Configuration file**

The configuration file contains information about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml .

#### **hibernate.cfg.xml**

<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"

          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>

        <property name="hbm2ddl.auto">update</property>

        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

        <property name="connection.username">system</property>

        <property name="connection.password">jtp</property>

        <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

    <mapping resource="employee.hbm.xml"/>

    </session-factory>

</hibernate-configuration>

### **4) Create the class that retrieves or stores the object**

In this class, we are simply storing the employee object to the database.

**package** com.javatpoint.mypackage;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** StoreData {

**public** **static** **void** main(String[] args) {

    //Create typesafe ServiceRegistry object

    StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

   Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

    Employee e1=**new** Employee();

    e1.setId(101);

    e1.setFirstName("Gaurav");

    e1.setLastName("Chawla");

    session.save(e1);

    t.commit();

    System.out.println("successfully saved");

    factory.close();

    session.close();

}

}

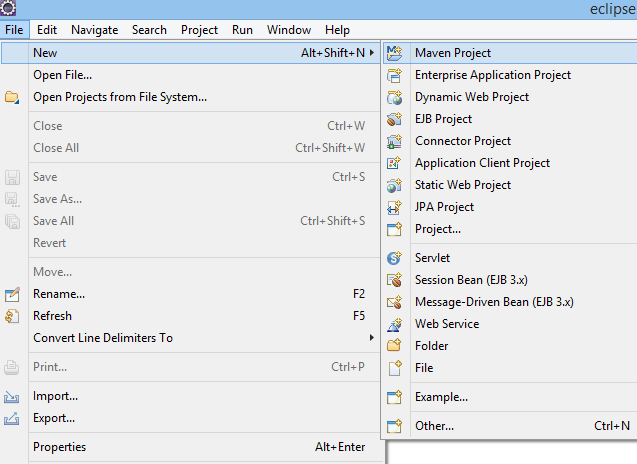
### **5) Load the jar file**

### **6) Run the first hibernate application without IDE**

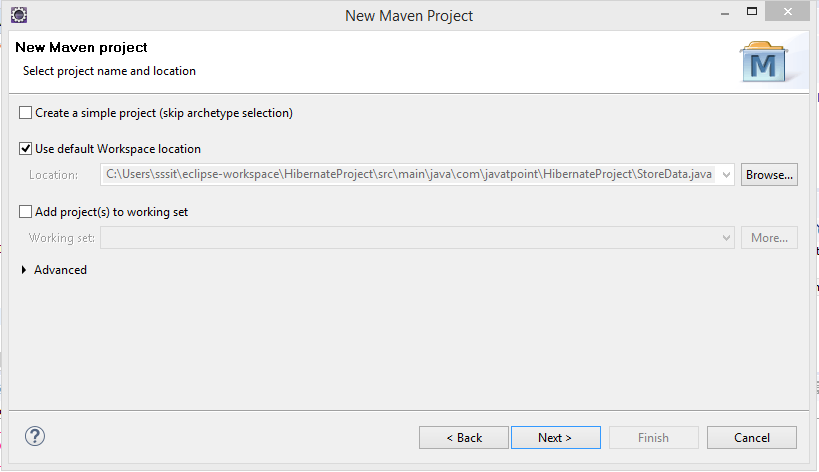
# Hibernate Example using Annotation in Eclipse

### **1) Create the Maven Project**

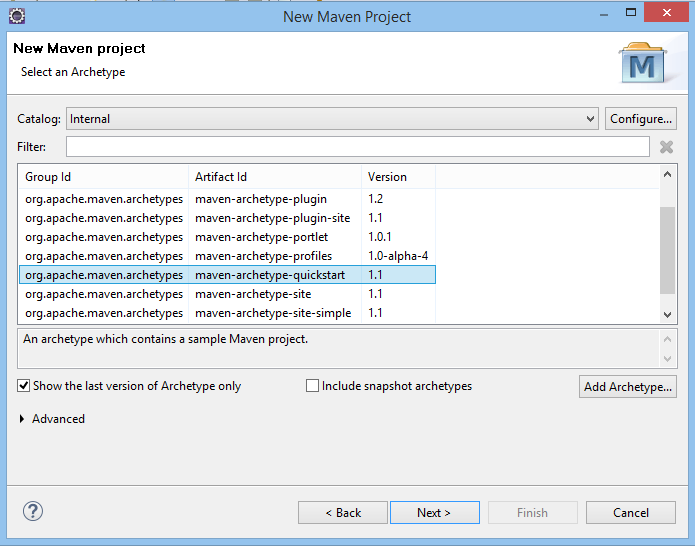
* To create the maven project left click on **File Menu** -**New**-**Maven Project**.



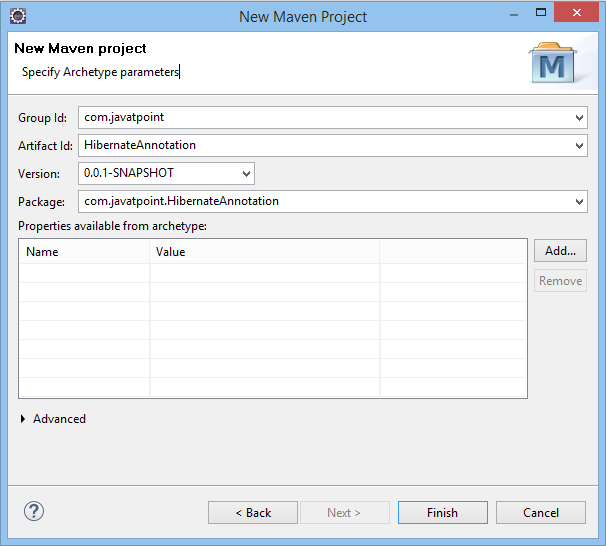
* The new maven project opens in your eclipse. **Click Next**.



* Now, select catalog type: internal and maven archetype - **quickstart** of 1.1 version. Then, **click next**.



* Now, specify the name of Group Id and Artifact Id. The Group Id contains package name (e.g. com.javatpoint) and Artifact Id contains project name (e.g. HibernateAnnotation). Then **click Finish**.



### **2) Add project information and configuration in pom.xml file.**

Open pom.xml file and click source. Now, add the below dependencies between <dependencies>....</dependencies> tag. These dependencies are used to add the jar files in Maven project.

<dependency>

    <groupId>org.hibernate</groupId>

    <artifactId>hibernate-core</artifactId>

    <version>5.3.1.Final</version>

</dependency>

<dependency>

    <groupId>com.oracle</groupId>

    <artifactId>ojdbc14</artifactId>

    <version>10.2.0.4.0</version>

</dependency>

### **3) Create the Persistence class.**

Here, we are creating the same persistent class which we have created in the previous topic. But here, we are using annotation.

**@Entity** annotation marks this class as an entity.

**@Table** annotation specifies the table name where data of this entity is to be persisted. If you don't use @Table annotation, hibernate will use the class name as the table name by default.

**@Id** annotation marks the identifier for this entity.

**@Column** annotation specifies the details of the column for this property or field. If @Column annotation is not specified, property name will be used as the column name by default.

To create the Persistence class, right click on **src/main/java - New - Class -** specify the class name with package - **finish**.

**Employee.java**

**package** com.javatpoint.mypackage;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name= "emp500")

**public** **class** Employee {

@Id

**private** **int** id;

**private** String firstName,lastName;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

### **4) Create the Configuration file**

To create the configuration file, right click on **src/main/java - new - file -** specify the file name (e.g. hibernate.cfg.xml) - **Finish**.

**hibernate.cfg.xml**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

        "-//Hibernate/Hibernate Configuration DTD 5.3//EN"

        "http://www.hibernate.org/dtd/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>

    <property name="hbm2ddl.auto">update</property>

        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>

        <property name="connection.username">system</property>

        <property name="connection.password">jtp</property>

        <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>

        <mapping **class**="com.javatpoint.mypackage.Employee"/>

    </session-factory>

</hibernate-configuration>

### **5) Create the class that retrieves or stores the persistent object.**

**StoreData.java**

**package** com.javatpoint.mypackage;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** StoreData {

**public** **static** **void** main(String[] args) {

    StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

    Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

    Employee e1=**new** Employee();

    e1.setId(101);

    e1.setFirstName("Gaurav");

    e1.setLastName("Chawla");

    session.save(e1);

    t.commit();

    System.out.println("successfully saved");

    factory.close();

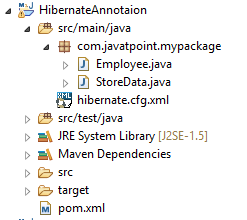
    session.close();

}

}

### **Run the application**

Before running the application, determine that the directory structure is like this.



To run the hibernate application, right click on the **StoreData - Run As - Java Application**.

Session Methods: https://www.tutorialspoint.com/hibernate/hibernate\_sessions.htm