http://sinauacademy.com/images/qqq.png



Java Hibernate

Version 0.3

# Table Revision

|  |  |  |
| --- | --- | --- |
| Version | Description | Author |
| 0.1 | Initial | Fajri Rahmat |
| 0.2 | Fix snippet code | Fajri Rahmat |
| 0.3 | Add one-to-one replation | Fajri Rahmat |

# Chapter 1

# Setting Up Hibernate

**Hibernate** merupakan suatu ***ORM(Object Relational Mapping)* *framework*** yang berguna melakukan operasi untuk memanipulasi *database* dengan memetakan tabel yang ada di *database* dengan JAVA *class*.

## Getting the required libraries for hibernate

Agar dapat menggunakan Hibernate, kita membutuhkan *library-library* Hibernate. Ada beberapa cara untuk mendapatkan *library* ini. Dua diantara nya adalah dengan secara manual *download* dari *website* Hibernate (<http://hibernate.org/orm/>) atau dengan dengan menggunakan **maven**.

Jika dengan cara men-*download* secara manual, harus menambahan *library-library* yang sudah ditambahkan tadi ke *classpath*. Pada modul ini, hanya akan dijelaskan dengan menggunakan **maven**.

## Try it out now

1. Buat project *maven* dengan **File > New > Maven Project**.
2. Pada window **New Maven Project,** centang pada **Create a simple project (skip archetype selection )**, lalu klik **Next > .**
3. Silakan **Group Id** dan **Artifact Id**, abaikan pada **Version** dan pastikan untuk saat ini **Packaging** nya ada **jar**.
4. Buka **pom.xml**, dan tambahkan penggalan *xml* berikut:

<dependencies>

<!-- MySQL connector -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.37</version>

</dependency>

<!-- Hibernate framework -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.1.0.Final</version>

</dependency>

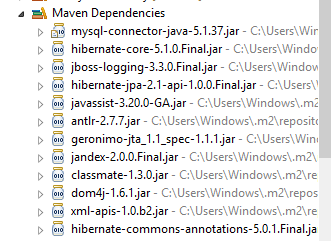
</dependencies>

<build>

<finalName>SinauHibernate</finalName>

</build>

1. Jalankan perintah **mvn clean install –U** untuk mendownload semua *dependency*.
2. Jika **BUILD SUCCESS**, maka semua *dependency-dependency* akan di-*download* ke workspace. Berhubung belum ada Java Class nya, berarti belum ada yang akan di-*compile*.



## Membuat Hibernate Persistent Class

Seperti yang sudah dijelaskan pada bagian sebelumnya, **ORM** akan memetakan object(POJO) di Java dengan tabel di *database*. Berikut langkah-langkahnya:

1. Buatlah *package* baru pada **src/main/java**, dengan nama **sinau.module.hibernate.pojo**.
2. Di dalam *package* tersebut, buatlah *class* baru dengan nama **Employee**.

**package** sinau.module.hibernate.pojo;

**public** **class** Employee {

**private** **long** id;

**private** String firstName;

**private** **double** salary;

//setter and getter here

}

## Database

Pastikan MySQL Server sudah *on*, dan *execute query* berikut pada MySQL *Client*.

**CREATE** DATABASE sinau\_hibernate;

use sinau\_hibernate;

**create** **table** EMPLOYEE (

id **INT** **NOT** **NULL** auto\_increment,

first\_name **VARCHAR**(20) **default** **NULL**,

salary **DECIMAL** **default** **NULL**,

**PRIMARY** **KEY** (id)

);

## Hibernate Mapping File

Sebelumnya kita sudah membuat *class* yang nantinya akan di-*mapping* dengan tabel yang ada di *database*. Selanjutnya kita akan melakukan *mapping* antara tabel EMPLOYEE yang ada di *database sinau\_hibernate* dengan *class* EMPLOYEE.

1. Buat folder baru didalam **src/main/resources** dengan nama **config**. Di dalam folder ini kita akan simpan semua file konfigurasi.
2. Di dalam folder **config**, buatlah file xml yang baru dengan nama **Employee.hbm.xml**.

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

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name=*"sinau.module.hibernate.pojo.Employee"* table=*"EMPLOYEE"*>

<id name=*"id"* type=*"long"*>

<column name=*"id"* />

<generator class=*"identity"* />

</id>

<property name=*"firstName"* type=*"string"* >

<column name=*"first\_name"* length=*"20"* />

</property>

<property name=*"salary"* type=*"double"*>

<column name=*"salary"* />

</property>

</class>

</hibernate-mapping>

1. Masih didalam folder yang sama, buatlah file xml baru dengan nama **hibernate.cfg.xml**. File ini adalah file konfigurasi hibernate. Pada file ini, akan dikonfigurasi koneksi *database* seperti *username, password, database,* alamat *server* dan juga konektor yang digunakan.

<?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>

<property name=*"hibernate.bytecode.use\_reflection\_optimizer"*>false</property>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.password"*>yourpassword</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/sinau\_hibernate</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"show\_sql"*>true</property>

<!-- Mapping configuration -->

<mapping resource=*"/config/Employee.hbm.xml"*></mapping>

</session-factory>

</hibernate-configuration>

## Main Class

1. Sekarang kita akan buat *Main class* nya.

**package** sinau.module.hibernate.app;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**import** sinau.module.hibernate.pojo.Employee;

**public** **class** App {

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

SessionFactory sessionFactory = **new** Configuration().configure("/config/hibernate.cfg.xml").buildSessionFactory();

Session session = sessionFactory.openSession();

session.beginTransaction();

Employee employee = **new** Employee();

employee.setFirstName("Fajri Rahmat");

employee.setSalary(13000000);

session.save(employee);

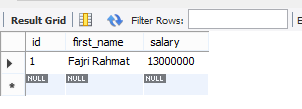
session.getTransaction().commit();

sessionFactory.close();

}

}

1. Dan berikut hasilnya setelah di-*run*.



## Exercise

1. Dengan perintah yang sama, tambahkan beberapa data lagi untuk ditambahkan ke *database*.
2. Buatlah method atau fungsi untuk mengambil data dari database.

***Hint:***

List employees = session.createQuery("FROM Employee").list();

1. Buatlah fungsi untuk melakukan *delete* dan *update* data dari *database*. (**Hint: session.delete(obj)** dan **session.update(obj)** ).

# Chapter 2

# Associations

Pada sesi sebelumnya, sudah dilakukan bagaimana memetakan tabel yang ada pada *database* dengan *class* JAVA. Di dalam *real project*, biasanya akan terdapat lebih dari 1 tabel dan diantara tabel tersebut akan memiliki relasi seperti **one-to-one, one-to-many, many-to-many.** Pada tutorial ini tidak dijelaskan secara rinci maksud dari relasi tersebut. Pada tutorial ini hanya akan dijelaskan bagaimana mengimplementasikan relasi-relasi pada Hibernate.

## One-to-One

1. Jalankan *script ddl* berikut pada aplikasi MySQL *client* yang digunakan:

**create** **table** ADDRESS (

employeeId **INT** **NOT** **NULL**,

street\_name **VARCHAR**(40) **default** **NULL**,

city\_name **VARCHAR**(40) **default** **NULL**,

state\_name **VARCHAR**(40) **default** **NULL**,

zipcode **VARCHAR**(10) **default** **NULL**,

**FOREIGN KEY(employeeId)**

);

1. Buatlah ***POJO Class***baru dengan nama ***Address***.

**package** sinau.module.hibernate.pojo;

**public** **class** Address {

**private** **long** employeeId;

**private** String street;

**private** String city;

**private** String state;

**private** String zipcode;

**private** Employee employee;

//setter and getter here

}

1. Tambahkan baris ini pada **Employee.hbm.xml** (didalam *tag <class>*)

<one-to-one name=*"address"* class=*"sinau.module.hibernate.pojo.Address"* cascade=*"save-update"* />

1. Buat *mapping* baru untuk *class* **Address** dengan tabel **Address**.

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

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name=*"sinau.module.hibernate.pojo.Address"* table=*"address"*>

<id name=*"employeeId"* type=*"int"* column=*"id"*>

<generator class=*"foreign"*>

<param name=*"property"*>employee</param>

</generator>

</id>

<one-to-one name=*"employee"* class=*"sinau.module.hibernate.pojo.Employee"* constrained=*"true"* /> <property name=*"street"* column=*"street\_name"* type=*"string"* />

<property name=*"city"* column=*"city\_name"* type=*"string"* />

<property name=*"state"* column=*"state\_name"* type=*"string"* />

<property name=*"zipcode"* column=*"zipcode"* type=*"string"* />

</class>

</hibernate-mapping>

1. Tambahkan konfigurasi *mapping* untuk tabel Address di **hibernate.cfg.xml**.

<mapping resource=*"/config/Address.hbm.xml"*></mapping>

1. Ganti **App.java** menjadi seperti dibawah ini:

**package** sinau.module.hibernate.app;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.cfg.Configuration;

**import** sinau.module.hibernate.pojo.Address;

**import** sinau.module.hibernate.pojo.Employee;

**public** **class** App {

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

SessionFactory sessionFactory = **new** Configuration().configure("/config/hibernate.cfg.xml").buildSessionFactory();

Session session = sessionFactory.openSession();

session.beginTransaction();

Address empAddress = **new** Address();

empAddress.setCity("Kuningan");

empAddress.setState("Jakarta Selatan");

empAddress.setZipcode("12940");

empAddress.setStreet("Jln. Bahagia Selalu");

Employee employee = **new** Employee();

employee.setFirstName("Fajri Fulan");

employee.setSalary(15000000);

employee.setAddress(empAddress);

empAddress.setEmployee(employee);

session.save(employee);

session.getTransaction().commit();

sessionFactory.close();

}

}