



# Back End Development 1 ○

---

+

## Sesi 13

The background is a solid dark blue. It features several abstract geometric elements: a light blue triangle in the upper left, a dark blue circle in the upper right, a light blue rectangle in the top right corner, a white circle in the lower left, and a white arc in the lower right. The text is centered in the middle of the image.

**ORM -  
HIBERNATE** +

## ORM HIBERNATE - Sesi 13

# Introduction

Apa itu ORM ?

ORM atau dikenal juga dengan Object Relationship Mapping adalah Jembatan antara OOP dengan Database Relational.

Jika di 2 pertemuan sebelumnya kita belajar terkait Database ( MySQL ) dan cara koneksi database ke Java, Coba Perhatikan :

Semakin banyaknya query SQL didalam source code Java akan merepotkan developer karena penggunaan SQL sangat berbeda dengan Java, contoh :

Pada saat kita melakukan parsing variabel mulai dari database, set up koneksi dan Operasi CRUD tentu hal ini ber-impact pada Performance & Maintainability app Java yang kita buat.

Nah Insiden ini dikenal dengan Impedance Mismatch.

Maka Dari itu kita butuh ORM untuk melakukan pemetaan database ke dalam OOP.

## Contoh source non ORM

```
String host = "localhost";
String db = "hacktiv8";
String user = "root";
String pass = "";
String driver = "com.mysql.jdbc.Driver";
String koneksi = "";

koneksi = "jdbc:mysql://" + host + ":3306/" + db + "?user=" + user + "&password=" + pass;

Class.forName(driver).newInstance();
Connection con = DriverManager.getConnection(koneksi);
Statement stmt = con.createStatement();

String ReqUser = request.getParameter("username");
String ReqPass = request.getParameter("password");
String QUERY_INSERT = "INSERT INTO Mahasiswa VALUES (" + ReqUser + "," + ReqPass + ")";
```

## Contoh source with ORM

```
public void tambahMhs (){

    Mahasiswa baru = new Mahasiswa();
    baru.setNRP(1);
    baru.setNama("fox");
    baru.setPassword("BootcampH8");

    HibernateUtil
        .getSessionFactory()
        .getCurrentSession().save(baru);

}
```



Keterangan :

1. Kedua Source diatas sama-sama melakukan operasi Insert data,

Lalu apa bedanya ?

Ketika kita menggunakan ORM Operasi Koneksi Database dapat lebih mudah dimanage sehingga code jadi jauh lebih mudah untuk dibaca.

Metode yang paling umum dari ORM adalah memetakan table atau query ke dalam class JAVA atau sering disebut dengan POJO atau Java Beans.

Apa itu POJO ? Plain Old Java Object

Class pada Java yang hanya berisi Variabel dan setter getter TANPA ada method proses lainnya

```
public class Mahasiswa {  
    private int NRP;  
    private String nama;  
    private String password;  
  
    public String getNama() {  
        return nama;  
    }  
    public int getNRP() {  
        return NRP;  
    }  
    public String getPassword() {  
        return password;  
    }  
    public void setNama(String nama) {  
        nama = nama;  
    }  
    public void setNRP(int nrp) {  
        NRP = nrp;  
    }  
    public void setPassword(String password) {  
        this.password = password;  
    }  
}
```

Mahasiswa	
NRP	int
Nama	varchar
Password	varchar

### DAO

Nah setelah kita tau tentang POJO, selanjutnya akan ada yang dinamakan DAO ( Data Access Object ), apa itu?

Class Java yang berisi operasi CRUD ( Create Retrieve Update Delete ),

Lalu Apa hubungannya dengan POJO?

POJO hasil pemetaan database menjadi Parameter DAO yang dibuat.

Database di mapping menggunakan POJO

Lalu Proses CRUD dikerjakan menggunakan DAO.



## Hibernate

Apa itu Hibernate ORM?

*library object-relational mapping* untuk bahasa pemrograman Java, yang menyediakan kerangka kerja (*framework*) untuk memetakan *model object-oriented domain* ke database relasional tradisional.

Apa itu bedanya *Hibernate* & *Maven* ?

Hibernate ini menyediakan *framework* untuk Java untuk koneksi ke database relasional seperti MySQL, SQL Server, Oracle, DB2, dan lain-lain.

Sedangkan Maven sendiri, itu adalah plugin Eclipse yang digunakan untuk mengatur dependency library.

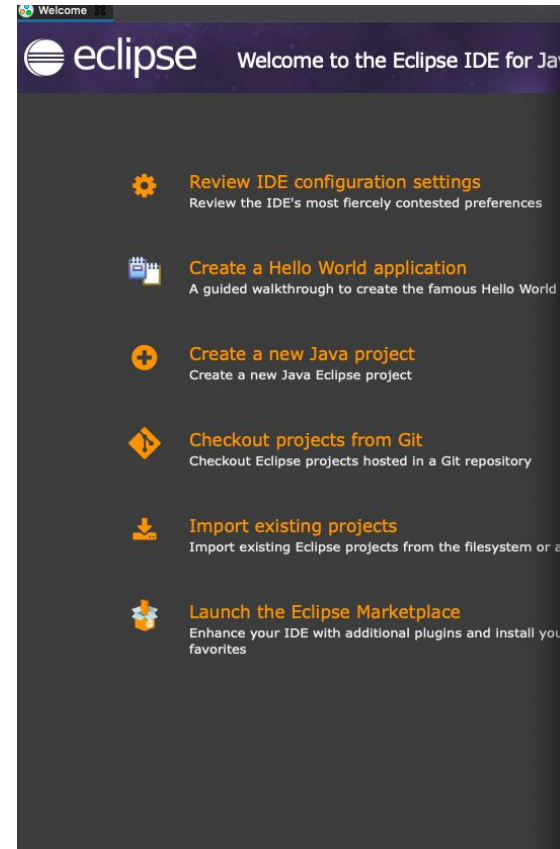
Ok supaya lebih mudah implementasinya kita akan coba implementasikan langsung dengan membuat CRUD Sederhana :

untuk project kali ini kita membutuhkan :

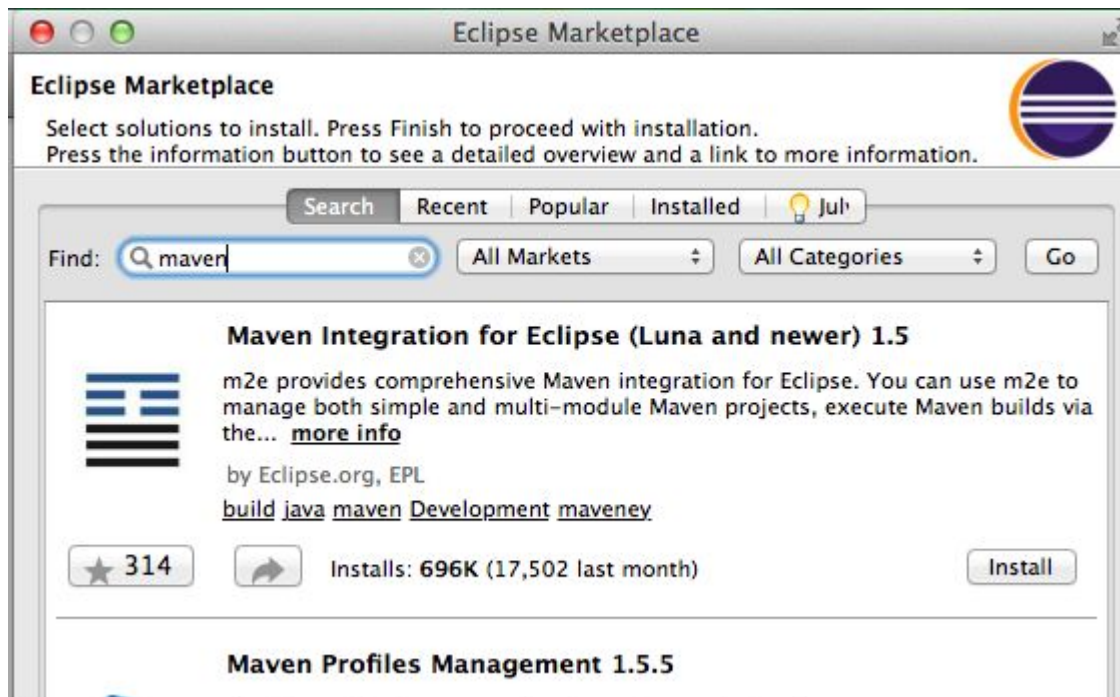
1. Database mySQL , jadi pastikan sudah terinstall.

2. Maven

Klik Launch to Eclipse Marketplace => Search Maven

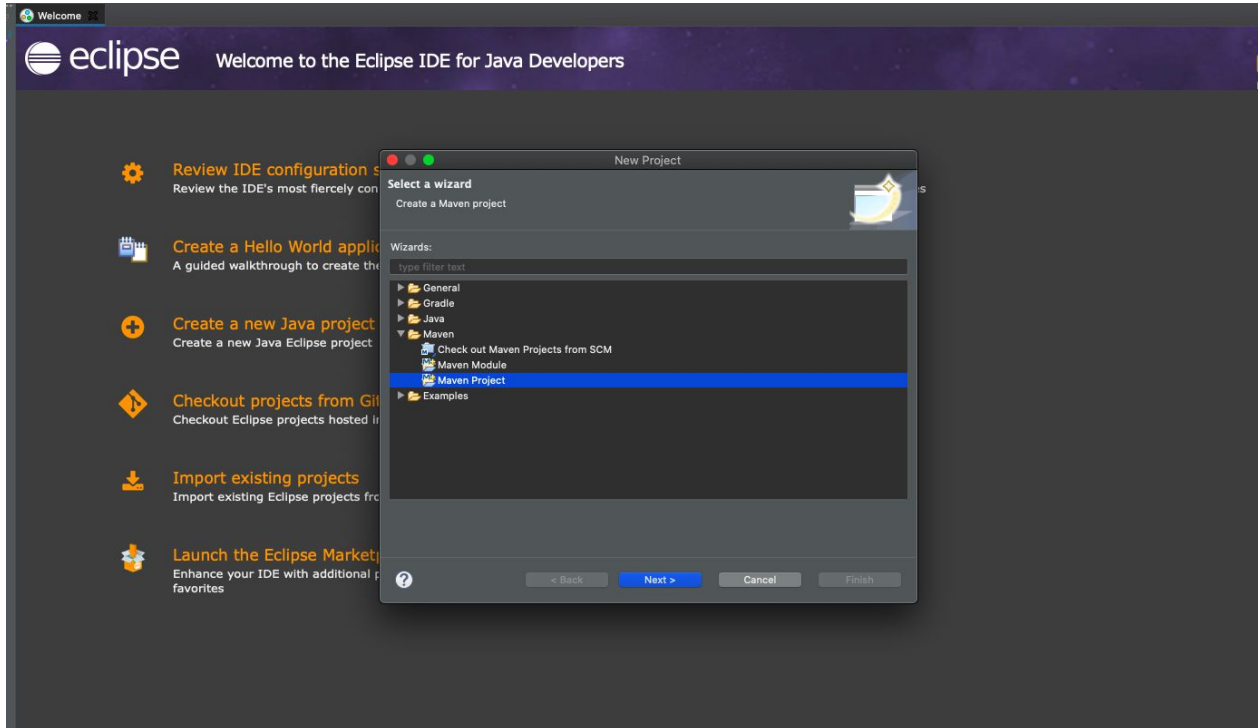






## Buat Maven Project di Eclipse

1. Klik kanan di *Project Explorer*. *New Project*, cari Maven Project.
2. Pada *New Maven Project*. Centang “*Create a simple project*”. Kemudian klik Next.
3. Isi Group Id, misalnya “org.coba”, Artifact Id misalnya “testhibernate1” dan Name misalnya Test Hibernate1.



New Maven Project

**New Maven project**  
Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

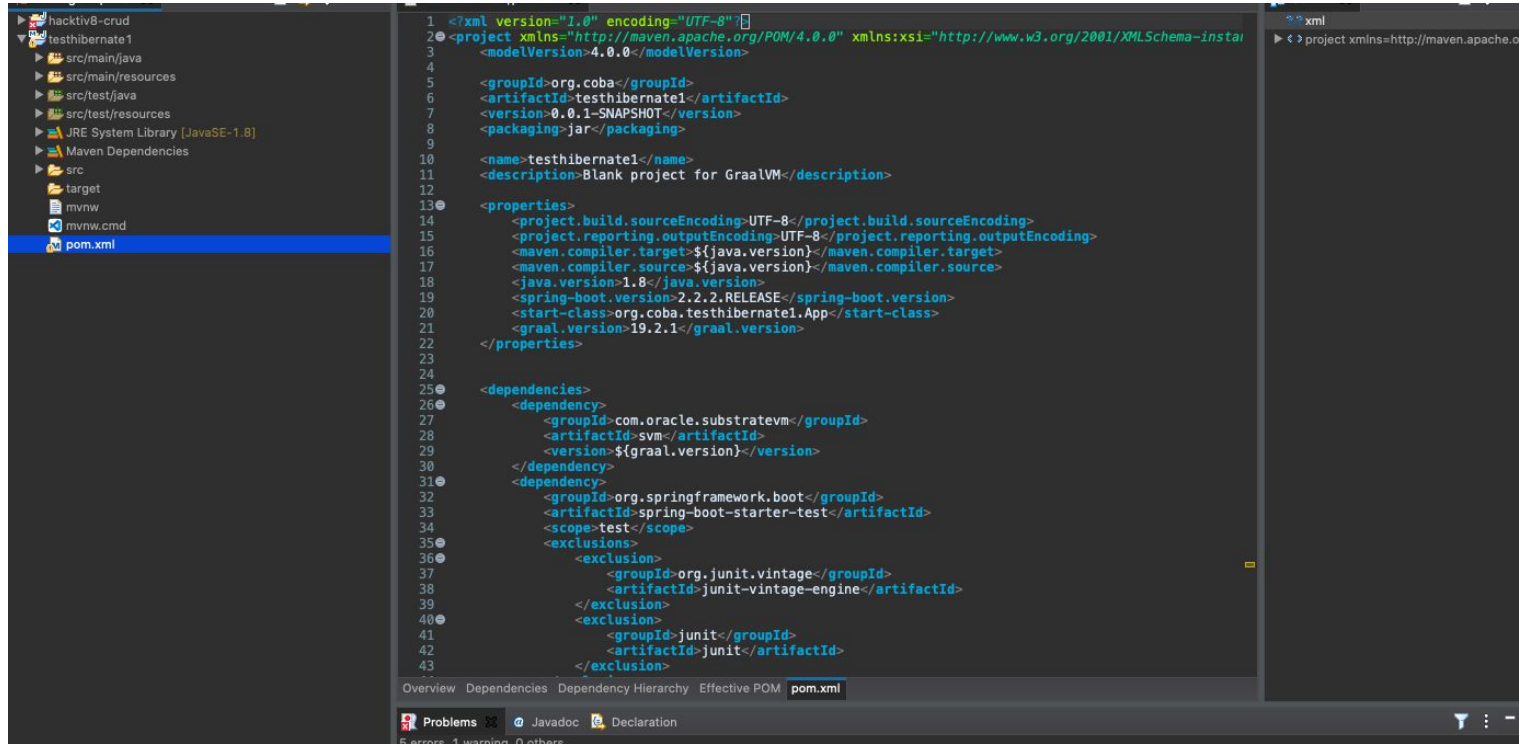
Description:

Parent Project



## Definisikan *dependency* di pom.xml

Buka src/pom.xml, definisikan dependency hibernate, hibernate-annotation, jta dan mysql-connector-java seperti contoh di bawah.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>org.coba</groupId>
6   <artifactId>testhibernate1</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>jar</packaging>
9
10  <name>testhibernate1</name>
11  <description>Blank project for GraalVM</description>
12
13  <properties>
14    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
16    <maven.compiler.target>${java.version}</maven.compiler.target>
17    <maven.compiler.source>${java.version}</maven.compiler.source>
18    <java.version>1.8</java.version>
19    <spring-boot.version>2.2.2.RELEASE</spring-boot.version>
20    <start-class>org.coba.testhibernate1.App</start-class>
21    <graal.version>19.2.1</graal.version>
22  </properties>
23
24  <dependencies>
25    <dependency>
26      <groupId>com.oracle.substratevm</groupId>
27      <artifactId>svm</artifactId>
28      <version>${graal.version}</version>
29    </dependency>
30    <dependency>
31      <groupId>org.springframework.boot</groupId>
32      <artifactId>spring-boot-starter-test</artifactId>
33      <scope>test</scope>
34      <exclusions>
35        <exclusion>
36          <groupId>org.junit.vintage</groupId>
37          <artifactId>junit-vintage-engine</artifactId>
38        </exclusion>
39        <exclusion>
40          <groupId>junit</groupId>
41          <artifactId>junit</artifactId>
42        </exclusion>
43      </exclusions>
44    </dependency>
45  </dependencies>
```



```
<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>org.coba</groupId>
  <artifactId>testhibernate1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Test Hibernate1</name>

  <dependencies>
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate</artifactId>
      <version>3.2.6.ga</version>
    </dependency>
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-annotations</artifactId>
      <version>3.3.1.GA</version>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.9</version>
    </dependency>
    <dependency>
      <groupId>javax.transaction</groupId>
      <artifactId>jta</artifactId>
      <version>1.1</version>
    </dependency>
  </dependencies>
```



```
<build>
  <finalName>Hibernate4</finalName>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Klik kanan pada src/pom.xml Run As > Maven Build. Ketik package pada Goals, kemudian Run. Sambil menunggu sampai Eclipse selesai mendownload dependency, lanjutkan proses selanjutnya.



Buat Database MySQL dan contoh table

Selanjutnya untuk mencoba koneksi ke database test dan table address\_book.

Maka harus dibuatkan database dan tablenya.

```
CREATE TABLE IF NOT EXISTS `address_book` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255) NOT NULL,  
  `address` varchar(255) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
```



## Buat file konfigurasi Hibernate

Buat file `src/main/resources/hibernate.cfg.xml` dengan cara klik kanan folder `resources`, `New > Other, XML > XML file`.

Perhatikan bahwa mapping resource mengarah ke file `testhibernate1/AddressBook.hbm.xml`.

File ini akan dibuat dilangkah selanjutnya.



```
<?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="connection.url">jdbc:mysql://localhost:3306/test</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <property name="hbm2ddl.auto">create</property>
        <property name="connection.pool_size">1</property>
        <property name="current_session_context_class">thread</property>
        <mapping resource="testhibernate1/AddressBook.hbm.xml"></mapping>
    </session-factory>
</hibernate-configuration>
```



Project Explorer

- RemoteSystemsTempFiles
- Servers
- test1
- testhibernate1
  - src/main/java
  - src/main/resources
    - hibernate.cfg.xml
  - src/test/java
  - src/test/resources
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - src
    - main
      - java
      - resources
    - test
      - java
      - resources
    - target
    - pom.xml

testhibernate1/pom.xml

hibernate.cfg.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3     "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4     "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd"
5 <hibernate-configuration>
6     <session-factory>
7         <property name="connection.url">jdbc:mysql://localhost:3306/test</property>
8         <property name="connection.username">root</property>
9         <property name="connection.password"></property>
10        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
11        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
12        <property name="show_sql">true</property>
13        <property name="format_sql">true</property>
14        <property name="hbm2ddl.auto">create</property>
15        <property name="connection.pool_size">1</property>
16        <property name="current_session_context_class">thread</property>
17        <mapping resource="hibernate4/contacts.hbm.xml"></mapping>
18    </session-factory>
19 </hibernate-configuration>
20
```

Design Source

Tasks Console



Pada tahap ini berarti, Hibernate dan beberapa dependency telah terpasang dengan menggunakan Maven, dan konfigurasi file telah disambungkan ke MySQL.

## Buat Model Class

Ini adalah model class untuk fungsi-fungsi ke database. `src/main/java/AddressBook.java`.

Klik kanan pada folder java, New > Other > Java > Class, kemudian Next.

Ketikkan AddressBook pada field Name. folder `src/main/java/testhibernate1` dan file `AddressBook.java` akan terbuat secara otomatis. Tekan F5 atau refresh.



```
package testhibernate1;

import javax.persistence.Entity;

@Entity
public class AddressBook {
    int id;
    String name;
    String address;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
}
```



**HACKTIV8**

## Buat Mapping File

Mapping file ini untuk mendefinisikan struktur table yang nanti dihubungkan ke Model Class.

Isinya adalah mapping untuk table address\_book di src/main/java/AddressBook.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="testhibernate1.AddressBook" table="address_book" >
    <id name="id" type="int">
      <column name="id" />
      <generator class="increment" />
    </id>
    <property name="name" type="string">
      <column name="name" length="255" not-null="true" />
    </property>
    <property name="address" type="string">
      <column name="address" length="255" not-null="true" />
    </property>
  </class>
</hibernate-mapping>
```



## Aplikasi Java

Nah aplikasi ini adalah sebagai controllernya yang nanti bisa dikembangkan menjadi CRUD (Create, read, update, delete).

Buat java class, src/main/java/App.java. Klik kanan pada folder testhibernate1, New > Other > Java > Class, kemudian Next. Ketikkan App pada field Name.



**HACKTIV8**

```
package testhibernatel;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.AnnotationConfiguration;

import testhibernatel.AddressBook;

public class App {
    public static void main(String[] args) {
        SessionFactory sessionFactory = new AnnotationConfiguration()
            .configure().buildSessionFactory();
        Session session = sessionFactory.getCurrentSession();

        Transaction tx = session.beginTransaction();

        AddressBook emp = new AddressBook();

        emp.setId(1);
        emp.setName("arif");
        emp.setAddress("depok");

        session.save(emp);

        tx.commit();
    }
}
```



Selesai sudah. Jalankan dengan mengklik kanan App.java, Run As > Java Application.

```
24         session.save(obj);
25
26         tx.commit();
27     }
28 }
29
```

Tasks Console

<terminated> App (1) [Java Application] /Library/Java/Java'

```
select
    max(id)
from
    address_book
Hibernate:
insert
into
    address_book
(name, address, id)
values
    (?, ?, ?)
```

Cek dan perhatikan apabila ada error. Apabila sudah sesuai semua, aplikasi ini akan meng insert data ke dalam table address\_book.