# Hibernate One to One Example using Annotation

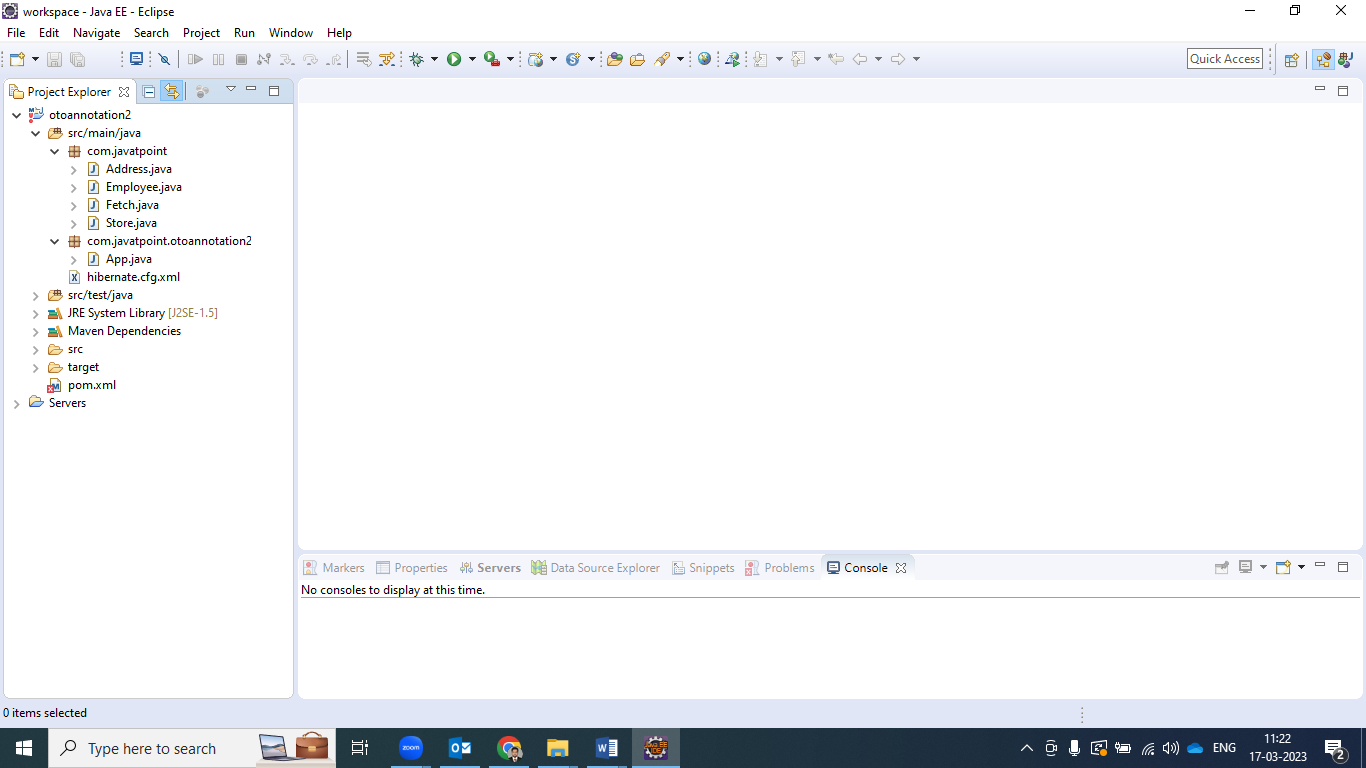
Here, we are going to perform one to one mapping by one-to-one element using annotation. In such case, no foreign key is created in the primary table.

In this example, one employee can have one address and one address belongs to one employee only. Here, we are using bidirectional association. Let's look at the persistent classes.

### Persistent classes for one to one mapping

There are two persistent classes Employee.java and Address.java. Employee class contains Address class reference and vice versa.

Project Structure:



**Employee.java**

**package** com.javatpoint;

**import** javax.persistence.\*;

@Entity

@Table(name="emp220")

**public** **class** Employee {

    @Id

    @GeneratedValue(strategy=GenerationType.AUTO)

    @PrimaryKeyJoinColumn

**private** **int** employeeId;

**private** String name,email;

@OneToOne(targetEntity=Address.**class**,cascade=CascadeType.ALL)

**private** Address address;

**public** **int** getEmployeeId() {

**return** employeeId;

}

**public** **void** setEmployeeId(**int** employeeId) {

**this**.employeeId = employeeId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** Address getAddress() {

**return** address;

}

**public** **void** setAddress(Address address) {

**this**.address = address;

}

}

**Address.java**

**package** com.javatpoint;

**import** javax.persistence.\*;

@Entity

@Table(name="address220")

**public** **class** Address {

    @Id

    @GeneratedValue(strategy=GenerationType.AUTO)

**private** **int** addressId;

**private** String addressLine1,city,state,country;

**private** **int** pincode;

@OneToOne(targetEntity=Employee.**class**)

**private** Employee employee;  //simple dependency Injection

**public** **int** getAddressId() {

**return** addressId;

}

**public** **void** setAddressId(**int** addressId) {

**this**.addressId = addressId;

}

**public** String getAddressLine1() {

**return** addressLine1;

}

**public** **void** setAddressLine1(String addressLine1) {

**this**.addressLine1 = addressLine1;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

**public** Employee getEmployee() {

**return** employee;

}

**public** **void** setEmployee(Employee employee) {

**this**.employee = employee;

}

}

### 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>Your MySQL Name</groupId>

    <artifactId>MySQL</artifactId>

    <version>MySQL version</version>

</dependency>

### 3) Configuration file

This file contains information about the database and mapping file.

**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 **class**="com.javatpoint.Address"/>

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

    </session-factory>

</hibernate-configuration>

### 4) User classes to store and fetch the data

**Store.java**

**package** com.javatpoint;

**import** org.hibernate.\*;

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

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

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

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

**public** **class** Store {

**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.setName("Ravi Malik");

    e1.setEmail("ravi@gmail.com");

    Address address1=**new** Address();

    address1.setAddressLine1("G-21,Lohia nagar");

    address1.setCity("Ghaziabad");

    address1.setState("UP");

    address1.setCountry("India");

    address1.setPincode(201301);

    e1.setAddress(address1);

    address1.setEmployee(e1);

    session.persist(e1);

    t.commit();

    session.close();

    System.out.println("success");

}

}

### Output

Hibernate One to One Example using Annotation 1 Hibernate One to One Example using Annotation 2

**Fetch.java**

**package** com.javatpoint;

**import** java.util.Iterator;

**import** java.util.List;

**import** javax.persistence.TypedQuery;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

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

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

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

**public** **class** Fetch {

**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();

    TypedQuery query=session.createQuery("from Employee");

    List<Employee> list=query.getResultList();

    Iterator<Employee> itr=list.iterator();

**while**(itr.hasNext()){

     Employee emp=itr.next();

     System.out.println(emp.getEmployeeId()+" "+emp.getName()+" "+emp.getEmail());

     Address address=emp.getAddress();

     System.out.println(address.getAddressLine1()+" "+address.getCity()+" "+

        address.getState()+" "+address.getCountry()+" "+address.getPincode());

    }

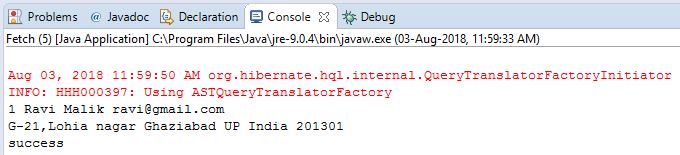
    session.close();

    System.out.println("success");

}

}

### Output



### Download

[Download this Hibernate Example](https://www.javatpoint.com/src/hb/otoannotation2.zip)