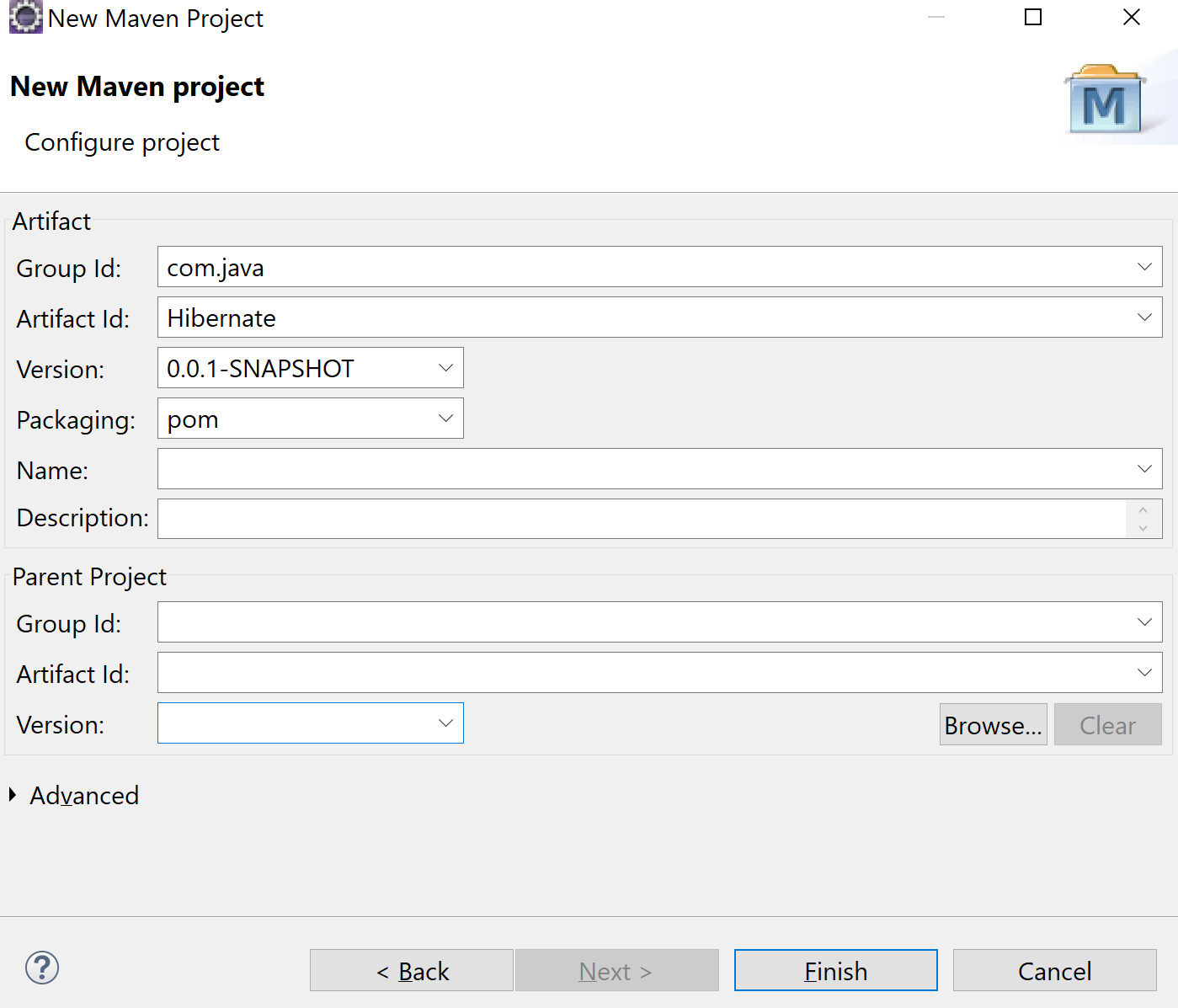
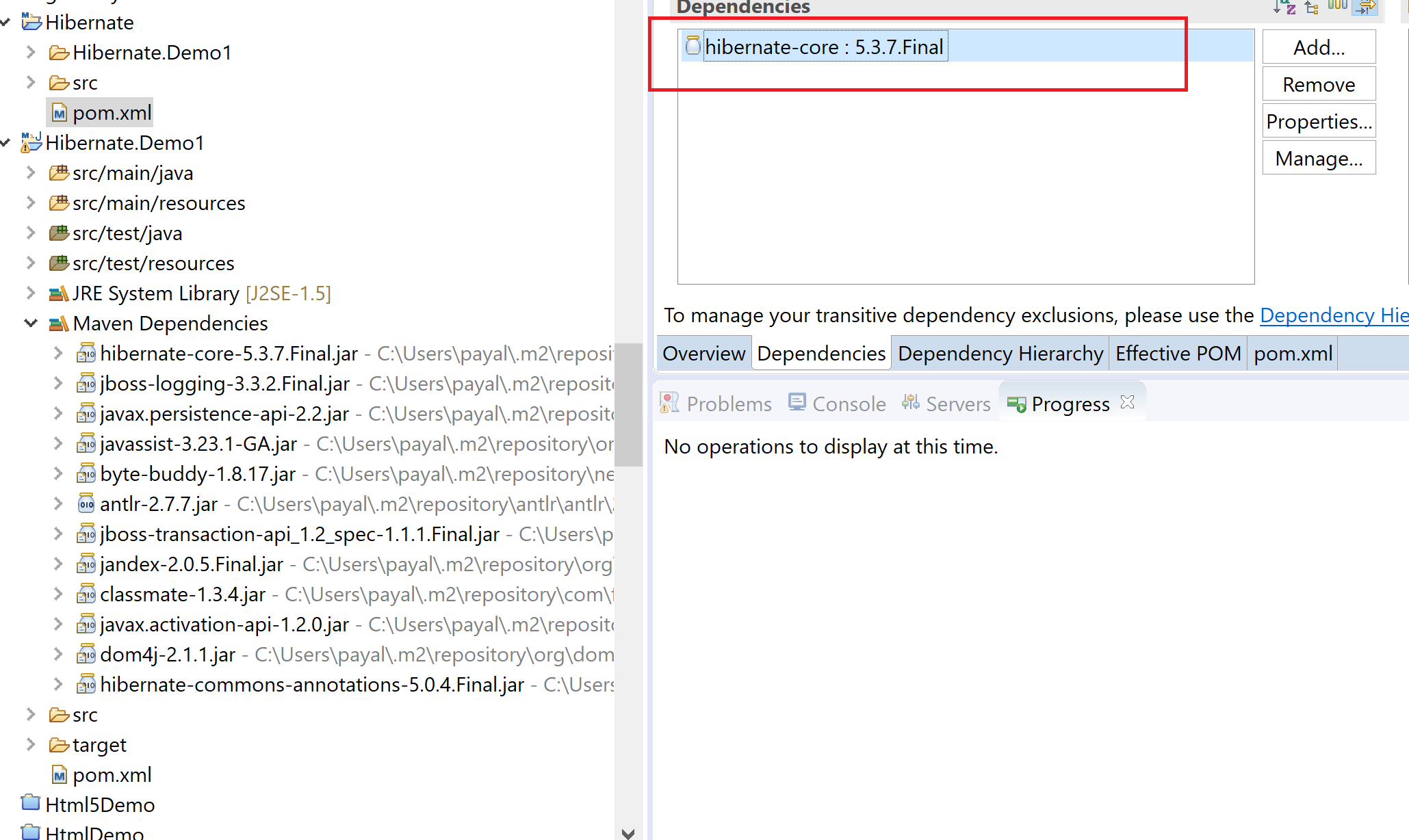
**Hibernate:**

Framework to connect to our db.

Desktop application.





If the employee leaves a company, I don’t want to delete the record from the table, just set isActive= false; **Soft Delete**

**To configure ur database related details**

1. Xml config
2. Java config

Configuration file

Hibernate-configuration: database details, hibernate properties

Session-factory: 1 db means 1 sf

2 dbs => 2 session factories => 2 configuration files

Hibernate: create ur schema also

<property name=*"hibernate.hbm2ddl.auto"*>create</property>

<mapping class=*"com.java.dto.Employee"*/>

Create: If employee table already exist, it will drop the table and re-create it. If u do not specify this property i.e. hbm2ddl.auto, it will not run any ddl command.

I want hibernate to create a table of name employee and I want hibernate to do the mapping for all the columns.

**@Entity**: If u want to map ur java class to db table, annotate it with @Entity. Employee class -> employee table. Every entity in hibernate should have a primary key column. And we use **@Id** to represent a primary key column,

Every annotation in entity class should be from javax.persistence package.

JPA (java persistence api)

**Hibernate is written on top of it.**

**Hibernate : ORM framework: Object relation mapping**

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

Show us the queries u r generating and executing on the console.

**Dialect:** Corresponding to which database u want to generate sql queries.

<property name=*"hibernate.dialect"*>org.hibernate.dialect.PostgreSQL9Dialect</property>

@Entity

@Id

@ElementCollection: collection

@GeneratedValue: to auto generate primary key values

**Database Normalization**

**Form1:**

In 1 column, u cannot have multiple values

**Load& get**

Load: lazy loading

Get: eager loading

Session.get(Employee.class, 1); //fire the query to the db to get employee details

Employee e=Session.load(Employee.class, 1); //no query fired

**Proxy objects**

Class Employee{

Id…..

}

Class Employee\_Proxy extends Employee{

Id=1;

Name=null;

}

e.getName(); //if I try to access any property on this object, then it will go and fire the query in the db

For improving the performance.

If I make the entity class to be final, it will not be able to create a proxy class, so load will also do eager loading in that case.

1 Employee: can have many projects

**Association Relationship (has-a)**

1. 1 to 1
2. 1 to many
3. Many to 1
4. **Many to many**

1 to many or many to many: It again does lazy loading for the associated object.

Q) Many employees can have 1 manager. But 1 employee cannot have more than 1 manager

Employee, Manager, show relationship

Save the data: 3 emp, 1 manager

Fetch employee :1: should it fetch the manager details also or no? **Eager fetch**