Hibernate

Eng. Ayah Alrifai' ^^

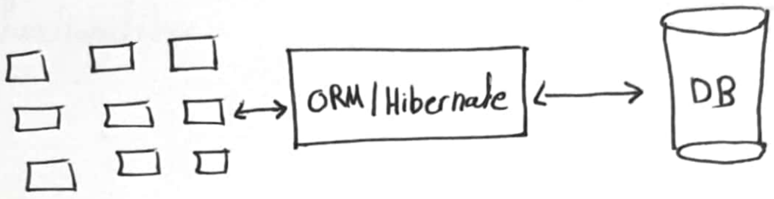
Create Hibernate file

src/hibernate.cfg.xml   this file used to add DB Configurations

should add Dependancy for JDBC & Hibernate

what ORM ?

stands for Object Relation Mapping is a Programming technique for convert data between relation databases and object oriented programming language.



Configuration Object

The Configuration object is the first Hibernate object you create in any Hibernate it is usually created once.

- Database Connection : this is handle throw one or more configuration file supported by Hibernate, these file are hibernate.proprties , hibernate.cfg.xml.

- class Mapping setup : This component create the connection between the java classes and database tables

Session Factory object

turn Configurs Hibernate for the application using the supplied configuration file it is safe thread, and you create it when run application and kept it for later use , each database should has session Factory

Session Object

a session is used to get Physical connection with database, the session objects should not be kep open for long time because usually thread safe and they should be created and destroy them as needed.

Transaction Object

Represents a unit of work with a database and most of the RDBMS supports transaction functionality

Query object

use SQL or Hibarnate Query Language (HQL) string to retrive data from the database and create objects

Criteria Object

used to create and execute object orinted criteria queries to retrive objects

Hibernate Proprties

File -> hibernate.proprties / hibernate.cfg.xml

1 hibernate.dialect = org.hibernate.dialect.DB2Dialect   تتغير حسب ال DB يلي شغالة عليها

This proprty makes Hibernate generate the appropriate SQL for the chosen DB.

2 hibernate.connection.driver-class = orcle.jdbc.driver.OracleDriver

Mongo , Oracle , MySQL   حسب أيش ال DB يلي أنا عم بستخدمها كل وحدة إلها Driver خاص فيها

3 hibernate.connection.url = jdbc : mysql : ///localhost : 3306 / database-Name

حسب نوع الـ DB الشغالة عليها

4 hibernate.connection.username = root

5 hibernate.connection.password = 123456

6 hibernate.connection.Pool_size = 1

limit the number of connections waiting in the Hibernate database connection Pool

DB and Dialect prop

org.hibernate.dialect. ___

DB2Dialect, HSQLDialect, InformixDialect, MySQLDialect, ...

Session Interface Methods

1 beginTransaction ()

Begin a unit of work and return associated Transaction object

2 Void cancleQuery()

3 Connection close()

4 CreateCriteria() return Criteria

5 Query createFilter(object collection, string querystring)

6 Query CreateQuery (String q)

7 sqlQuery CreateSQLQuery (String q)

8 Void delete(Object o)

9 Void delete(String entityName, Object o)

10 Session get(String entityName, Serializble id)

11 SessionFactory getSessionFactory()

12 Transaction getTransaction()

13 boolean isOpen()

14 Serializable save(Object o)

15 Void SaveorUpdate(Object o)

16 Void update(Object o)

17 Void update(String EntityName, Object o)

Pirsestent class

The entire concept on Hibernate to take the values from java class attributes and Persist them to a DB table

Persistent class or Plain Old Java Object (POJO)

لازم ال class يكون فيها default Const ، لازم يكون فيها ID ، كل ال Instance Var تكون Private ، إلهن setter , getter ، ما تكون ال class هاي Implement , extend ل serializable class

Mapping File

1 POJO Class

```
Public class Student {
        Private int id;
        Private String name;
        Private String birthDate;

        Public Studen() {

        }
        + setters
        + getters
}
```

2 RDBMs table

```
Create table STUDENT (
    id INT NOT NULL auto-increment,
    name VARCHAR(30) ,
    birthdate VARCHAR(8) ,
    PRYMARY_KEY (id));
```

3 student.hbm.xml

<Pojo-class-name>.hbm.xml

```
<hibernate-mapping>
        <class name="Student" table="STUDENT" >
                <meta attribute="class-description" >
                        student details
                </meta>

                <id name="id" type="int" column="id" >
                    <generator class="native">
                </id>
                <property name="name" column="name" type="String" />
                <property name="birthdate" column="birth_date" type="String" />

        </class>
</hibernate-mapping>
```

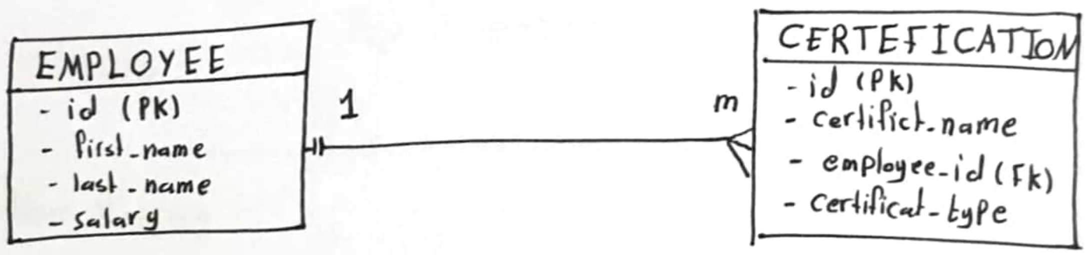
that is mean make id auto increment

ibernate mapping types

Mapping type	Java type	SQL type
integer	Integer	INTEGER
long	Long	BIGINT
short	Short	SMALLINT
float	Float	FLOAT
double	Double	DOUBLE
character	String	CHAR(1)
string	String	VARCHAR
byte	Byte	TINYINT
boolean	Boolean	BIT
yes/no	Boolean	CHAR(1) ('Y' or 'N')
true/false	Boolean	CHAR(1) ('T' or 'F')
date	java.sql.Date	DATE
time	java.sql.Time	TIME
~~date~~ timestamp	java.sql.Timestamp	TIMESTAMP
binary	byte[]	BLOB
text	String	CLOB
blob	Blob	BLOB

ollections Mapping

1	java.util.Set	<set>	initialized with	java.util.HashSet
2	java.util.SortedSet	<set>	initialized with	java.util.TreeSet
3	Java.util.List	<list>	initialized with	java.util.ArrayList
4	Java.util.Collection	<bag>,<ibag>	initialized with	java.util.ArrayList
5	Java.util.Map	<map>	initialized with	java.util.HashMap
6	java.util.SortedMap	<map>	initialized with	java.util.TreeMap



[1, 2, 3, 4]

in Employee class there is instance variable

[1] Private Set certificates ;

[2] Private SortedSet certificates;

[3] Private List certificates;

[4] Private Collection certificates;

in mapping for Employee

<set name="certificates" cascade="all">

الإسم بال emp class

<key column="employee_id" />

DB اسم ال col بال

<one-to-many class="Certifical" />

بإسم ال Pojo

<ISet>

حسب نوع ال Collection

ممكن تكون <list> , <bag> ,

<ibag>

[tutorialspoint] form more Info and Examples

5,6]

in Employee Pojo add Instance Variable

[5] Private Map certificates;

[6] Private SortedMap certificates;

in mapping Employee

```
<map name="certificates" cascade="all" >
```
Employee Class الإسم بال ←

```
    <key column="employee_id" />
```
DB إسم الـ Col بال ←

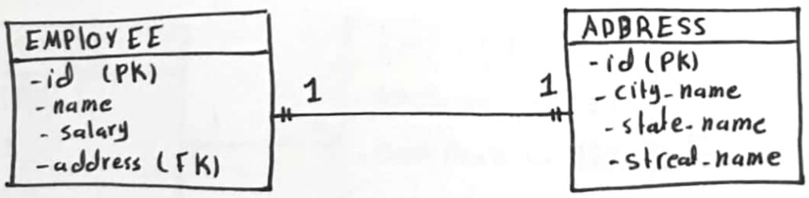
```
    <index column="certifical_type" type="string" />
```
هو عبارة عن الـ Key الخاص بال map ←
وهاد إسمه حسب الـ DB

```
    <one-to-many class="Certificates" />
```
إسم الـ Pojo ←

```
</map>
```

Association Mapping

[1] One-to-One



Employee class → Private Address address;

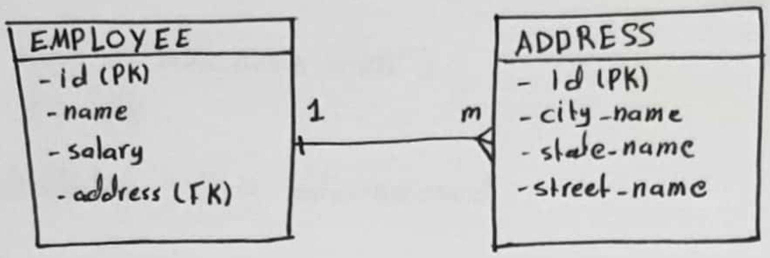
Employee mapping → `<many-to-one name="address" column="address" unique="true" class="Address" not-null="true" />`

- name="address": الإسم بال Employee Pojo ←
- column="address": الإسم بال DB ←
- unique="true": لأنه العلاقة one-to-one ←
- class="Address": Address Pojo ←

[2] One-to-Many

Can be implemented using a Set java collection that does not contain any duplicat element , We already have seen how to map Set Collection.

[3] Many-to-One



Employee class → Private Address address;

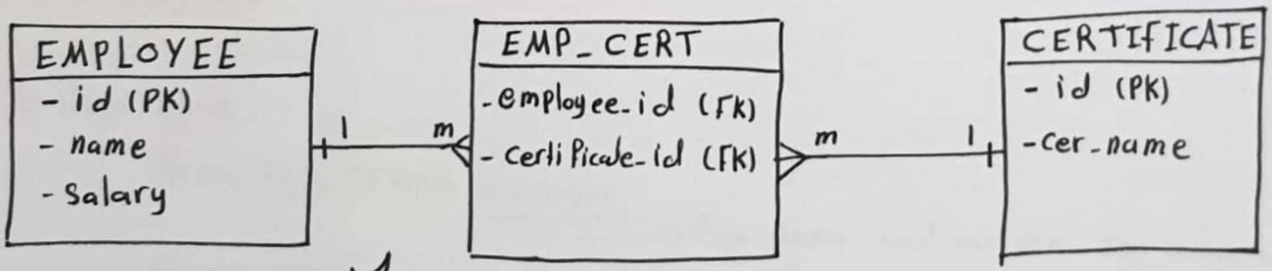
Employee mapping → <many-to-one name="$\underbrace{\text{address}}$" Column="$\underbrace{\text{address}}$"

Employee Pojo الإسم بال ← | DB الإسم بال ←

class = "$\underbrace{\text{Address}}$" not-null = "true" />

Address Pojo ←

[4] Many-to-Many



one-to-many ☑
So we will use Set

Employee Class → Private Set Certificates;

Employee mapping → <Set name="$\underbrace{\text{certificates}}$" cascade="save-update"

Employee Pojo الإسم بال ←

table="$\underbrace{\text{EMP\_CERT}}$" >

(FK) m:m إسم الtable يلي بتربط ←

<Key Column = "$\underbrace{\text{employee-id}}$" /> → column name in EMP-CERT

<many-to-many column="$\underbrace{\text{certificate-id}}$" class="$\underbrace{\text{Certificate}}$"

Column name in EMP-CERT ← | Pojo name ←

</set>

Annotations

Hibernate annotations are the newest way to define mappings without the use of XML file, You can use annotations in additional to or as a replecment of XML mapping metadata.

read about Mapping using annotations ^^

These are some annotations:

```
@Entity
@Table(name = "table name in DB")
@Id            it is PK
@GeneratedValue     it is auto-increment
@Column
@Column(name = "col name in DB")
   ↳ length = [size of string value]
     nullable = true
     unique = false
     updatable = true       //can update it in SQL statement
     insertable = true      // can insert it by SQL update stat
     ColumnDefinition = "....."  //use when generating the DDL
                                   for the column
```

Query Language

1 From cluse

```
String hql = "FROM Employee";
              └──→ Pojo name and can use Pkg name + class name
Query query = session.createQuery(~~query~~ hql);
List result = query.list();
```

2 As cluse

```
From Employee as E
```

3 SELECT Cluse

```
SELECT E.name FROM Employee AS E
```

4 WHERE Cluse

```
FROM Employee E WHERE E.Id = 10
```

5 ORDERBY Cluse

```
FROM Employee E WHERE E.id > 10 ORDER BY E.salary DESC
```

6 GROUB BY Cluse

```
SELECT SUM(E.salary), E.name FROM Employee E
GROUP BY E.name
```

Using named Parameters

```
hql = "FROM Employee E WHERE E.id = :employee-id ";
Query query = session.createQuery (hql);
query.setParameter ("employee-id", 10);
```

7 UPDATE Cluse

```
hql = "UPDATE Employee set salary = :salary WHERE id = :employee-id ";
Query query = Session.createQuery (hql);
query.setParameter ("salary", 1000);
query.setParameter ("employee-id", 10);
int result = query. executeUpdate();
```

8 DELETE Cluse

```
DELETE FROM Employee WHERE id = 10
```

* use executeUpdate()

INSERT Cluse

INSERT INTO Employee(name, Salary) SELECT name, salary FROM
Employee WHERE id=8

HQL supports INSERT INTO cluse only where records can be inserted from one object to another object

10 Aggregate Method

avg (Prop) , Count (Prop or *) , max(Prop), min(Prop) , sum (Prop)

11 Pagination using Query

```
query.SetFirstResult(1);
query.SetMaxResults(10);     // fetch 10 rows
```

read about joins in HQL ^^

Criteria Queries

allows you to build up a criteria query object programmatically where you can apply filtration rules and logical conditions.

```
Criteria cr = session.createCriteria( Employee.class);
```
Employee ← Pojo class

```
Criterion c1 = Restrictions.gt("Salary", 2000);
```
"Salary" ← Pojo instance var

```
Criterion c2 = Restriction.ilike("name", "Ayah%");
```
ilike ← Case sensitive like

```
Logical Expression orExp = Restrictions.or(c1, c2);

cr.add(orExp);
```
⇒ ممكن أضيف criterion مباشرة أو logExp دائماً بتكون العلاقة بينهن [and].

For BluLogix Ignor It now

```
Public enum Operators {

    EQUAL ("eq", " ${field} = '${Variable}'"),
    :
    Private final string code;
    Private final string template;

    Operators (String code, String template) {
        this.code = code;
        this.template = template;
    }

    + getter
    + setter
}
```

Restrictions

* EQUAL	eq	* LESS_THAN	lt	* IS_EMPTY	em
		* LESS_THAN-EQUAL	lte	* IS_NOT_EMPTY	nem
* NOT_EQUAL	neq	* GREATER_THAN	gt	* BETWEEN	btwn
* LIKE_FIRST	lkf	* GREATER_THAN-EQUAL	gte	* NOT-BETVEEN	nbtwn
* LIKE_LAST	lkl	* IS_NULL	nil	* IN	in
		* IS-NOT-NULL	nnil	* NOT-IN	nin
* LIKE_BOTH	lkb				
* LIKE_BOTH-INSENSTIVE	ilkb			* ORDER_BY_ASC	ordra
* NOT-LIKE-BOTH-INSENSTIVE	nilkb			* ORDER-BY-DESC	ordrd

do same things !!? No

iteria Example

```
Criteria  cr = session.createCriteria(Employee.class);

Criterion  c1 = Restrictions.lt("salary", 3000);

Criterion  c2 = Restrictions.gt ("salary", 1000);

Criterion  c3 = Restrictions.like ("name", "Ayah%");

Criterion  c4 = Restrictions.like ("name", "Sham%");

LogicalExpression  l1 = Restrictions.or (c3, c4);

cr.add (c1);                    // SELECT * FROM Employee WHERE

cr.add (c2);                    // salary < 3000 and salary > 1000

cr.add (l1);                    // and ( name like 'Ayah%' or

List results = cr.list();       //      name like 'Sham%' )
```

$\underbrace{\text{cr.list();}}$ ↳ to get the result

Pagination Using Criteria

```
cr.SetFirstResult(1);

cr.SetMaxResult(100);
```

Sorting Using Criteria

```
Order o1 = Order.desc ("Salary");

Order o2 = Order.asc ("name");

cr.addOrder(o1);

cr.addOrder(o2);
```

Projections & Aggregations

org.hibernate.criteria.Projections

```
1 rowCount()            // get total row count

2 avg ("salary")        // get average of a prop

3 countDistinct ("name") // get distnict count of a prop

4 max("salary")         // get maximum of a prop

5 min("salary")         // get minimum of a prop

6 sum("salary")         // get sum of a prop
```

```
Projection p = Projections.max("salary");

cr.addProjection(p);
```

Native SQL - Scalar Queries

The most basic SQL query is to get a list of scalars (values) from one or more tables

```
String sql = "select name, salary from employee";

SQLQuery query = session.createSQLQuery(sql);

query.setResultTransformer(Criteria.ALIAS_TO_ENTITY_MAP);

List result = query.list();

for (Object r : result) {
    Map row = (Map) r;
     row.get("name");
     row.get("salary");
}
```

هلا شكل ال data يلي رح ترجع ما بتناسب مع أي Pojo عندي فأنا مضطرة أخزن الداتا الراجعة على شكل Map

Native SQL - Entity Queries

```
    String sql = "select * from employee";
SQLQuery query = sesstion.createSQLQuery(sql);
    query.addEntity(Employee.class);
    List results = query.list();
```

بضبوله شكل الداتا يلي رح ترجع رح تكون من نوع Employee بالتالي هو لحاله رح يعمل mapping

Named SQL Queries

```
    String sql = "select * from employee where id = :employee-id";
SQLQuery query = session.createSQLQuery(sql);
    query.addEntity(Employee.class);
    query.setParameter("employee_id", 10);
```