Hibernate in tutorialspoint.com

# Hibernate in tutorialspoint.com

## ORM overview

What is **persistence**?

Persistence refers to the characteristic of state that outlives the process created it. Persistence means that we would like our application’s data to outlive the application process. In Java, we’d like the state of our objects to live beyond the scope of JVM so the same state is available later.

### What is JDBC?

JDBC: **J**ava **D**ata**B**ase **C**onnectivity. It provides Java APIs (Application Programming Interface) for accessing RDBMS from Java program. These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database. JDBC provides a flexible architecture to write a database independent application that can run on different platforms and interact with different DBMS without any modification.

JDBC Pros and Cons

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| - Clean, simple SQL processing  - Good performance with large data  - Very good for small applications  - Simple syntax | - Complex with large projects  - No encapsulation  - Hard to implement MVC concept  - Query is DBMS specific |

### Why Object Relational Mapping (ORM)?

RDMBS uses relational models. OOP system uses object models. Object-Relational mismatch: object models and relational models do not work well together.

5 mismatch problems:

|  |  |
| --- | --- |
| **Mismatch** | **Description** |
| Granularity | Đôi khi một object model có nhiều classes hơn số bảng phù hợp trong RDBMS. |
| Inheritance | RDBMS không định nghĩa Inheritance. |
| Identity | Java định nghĩa cả object identity (a==b) và object equality (a.equals(b)). |
| Associations | OO language dùng object references để định nghĩa quan hệ cha - con giữa các đối tượng trong khi RDBMS dùng foreign-key. |
| Navigation | Về cơ bản, cách truy cập đối tượng trong Java và RDBMS khác nhau. |

**O**bject-**R**elational **M**apping (ORM) giải quyết các mismatches trên.

### What is ORM?

ORM là **O**bject-**R**elational **M**apping (ORM) dùng converting data giữa RDBMS và OOP như Java, C#. Một ORM system có các lợi thế so với JDBC:

|  |
| --- |
| **Advantage** |
| Business code truy cập objects rather than DB tables. |
| Hides details of SQL queries from OO logic. |
| Entities based on business concepts rather than database structure. |
| Transaction management and automatic key generation. |
| Fast development of application. |

Một ORM solution bao gồm 4 entities sau:

|  |
| --- |
| **Entities** |
| An API to perform basic CRUD (**C**reate **R**ead **U**pdate **D**elete) operations on objects of persistent classes. |
| A language or API to specify queries that refer to classes and properties of classes. |
| A configurable facility for specifying mapping metadata. |
| A technique to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions. |

### Java ORM frameworks

Một số ORM frameworks: Hibernate, Spring DAO, Enterprise JavaBeans Entity Beans, ...

## Hibernate overview

Hibernate là một ORM solution cho JAVA.

Hibernate maps các Java classes với database tables, Java data types với SQL data types.

Hibernate đặt giữa Java objects và database server để handle tất cả việc về persisting của các objects đó dựa trên các O/R mechanisms and patterns phù hợp.



### Hibernate Advantages

Các lợi thế của Hibernate:

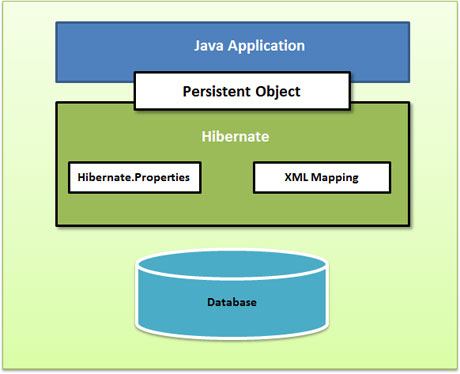
* Mapping Java classes với database tables dùng XML files mà không coding.
* Cung cấp simple APIs để storing và retrieving thông qua các Java objects để liên kết với database.
* Khi thay đổi table thì chỉ cần thay đổi file XML tương ứng.
* Chỉ cần thành thạo Java Objects, không cần thành thạo SQL objects.
* Không cần application server để operate.
* Thao tác được các complex associations của các objects trong database.
* Giảm thiểu truy cập database
* Cung cấp truy vấn dữ liệu một cách đơn giản.

### Các DBMS mà Hibernate hỗ trợ

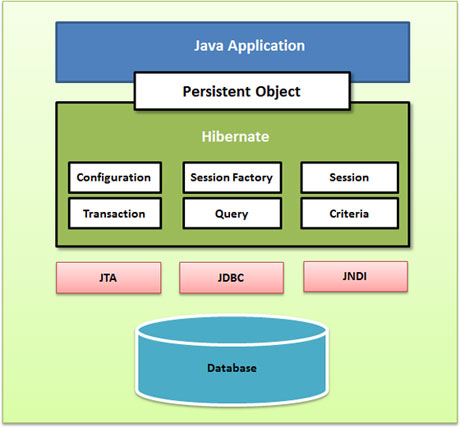
Hỗ trợ hầu hết các RDBMS thông dụng: MySQL, PostgreSQL, MSSQL Server, Oracle, DB2, Sybase, ...

## Hibernate Architecture

Hibernate đặt giữa Java application và Database để cung cấp các persistence services (và persistent objects) cho application. Very high level view của Hibernate Application Architecture:



Detailed view của Hibernate Application Architecture:



Hibernate dùng nhiều Java APIs có sẵn như: JDBC (Java DataBase Connectivity), Java Transaction API (JTA), và Java Naming & Directory Interface (JNDI).

JDBC cung cấp chức năng để làm việc với RDBMS, cho phép Hibernate hỗ trợ bất kỳ RDBMS nào thông qua JDBC driver. JNDI và JTA cho phép Hibernate tương tác với J2EE application servers.

Phần dưới đề cập chi tiết các thành phần.

**Configuration Object**

Object này thường tạo đầu tiên và một lần trong khi khởi tạo ứng dụng. Nó đại diện qua một file configuration hay properties mà Hibernate yêu cầu. Configuration object cung cấp 2 keys components:

* Database Connection: thông qua một hoặc nhiều configuration files do Hibernate hỗ trợ. Các files này là hibernate.properties và hibernate.cfg.xml.
* Class Mapping Setup: component này tạo connection giữa các Java classes và database tables.

**SessionFactory Object**

Configuration object được dùng để tạo một SessionFactory object mà giữ cấu hình Hibernate cho ứng dụng sử dụng các supplied configuration file và cho phép một Session object được instantiated. SessionFactory là một thread safe object và được dùng bởi tất cả threads của ứng dụng.

SessionFactory được tạo ra trong khi application start up và được giữ lại sử dụng về sau. Cần một SessionFactory object cho một database sử dụng một separate configuration file. Nếu dùng nhiều databases thì phải tạo nhiều SessionFactory objects.

**Session Object**

Session được dùng để get một physical connection với một database. Session object được instantiated mỗi lần có tương tác với database. Persistent objects được saved và retrieved xuyên qua một Session object.

Session objects thường không được mở lâu dài. Chúng được created và destroyed khi cần thiết.

**Transaction Object**

Hầu hết RDBMS có hỗ trợ transaction. Transactions trong Hibernate được handled bởi một underlying transaction manager và transaction (từ JDBC hoặc JTA).

Đây là một optional object. Hibernate applications có thể không dùng interface này mà dùng application code.

**Query Object**

Query objects dùng SQL hoặc Hibernate Query Language (HQL) string để lấy dữ liệu từ RDBMS và tạo objects. A Query instance được dùng để ràng buộc query parameters, giới hạn số kết quả trả về và thực thi query.

**Criteria Object**

Criteria object dùng để tạo và thực thi criteria queries hướng đối tượng để lấy về objects.

## Hibernate Configuration

Hibernate cần biết nơi để tìm thông tin mapping mà định nghĩa mối liên quan giữa Java classes và database tables. Nó cũng cần một tập các thiết lập cấu hình liên quan đến database và các tham số khác. Tất cả thông tin này được cung cấp qua một file properties theo chuẩn Java (hibernate.properties) hoặc dưới dạng XML (hibernate.cfg.xml).

### Hibernate Properties

Các properties quan trọng để cấu hình database:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| hibernate.dialect | Để phát sinh các SQL phù hợp cho từng loại database. |
| hibernate.connection.driver\_class | JDBC driver class. |
| hibernate.connection.url | JDBC url đến database instance. |
| hibernate.connection.username | Database username. |
| hibernate.connection.password | Database password. |
| hibernate.connection.pool\_size | Số connections chờ trong Hibernate database connection pool. |
| hibernate.connection.autocommit | Có dùng autocommit mode trong JDBC connection không. |

Nếu dùng database với application server và JNDI thì phải cấu hình các properties sau:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| hibernate.connection.datasource | JNDI name định nghĩa trong application server context. |
| hibernate.jndi.class | InitialContext class cho JNDI. |
| hibernate.jndi.  <JNDIpropertyname> | Đặt JNDI property cho JNDI InitialContext. |
| hibernate.jndi.url | Cung cấp URL for JNDI. |
| hibernate.connection.username | Database username. |
| hibernate.connection.password | Database password. |

### Hibernate với MySQL Database

Giả sử dùng Hibernate kết nối MySQL. Tạo file cấu hình hibernate.cfg.xml và đặt nó trong root của application's classpath.

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration SYSTEM

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">

org.hibernate.dialect.MySQLDialect

</property>

<property name="hibernate.connection.driver\_class">

com.mysql.jdbc.Driver

</property>

<!-- Assume test is the database name -->

<property name="hibernate.connection.url">

jdbc:mysql://localhost/test

</property>

<property name="hibernate.connection.username">

root

</property>

<property name="hibernate.connection.password">

root123

</property>

<!-- List of XML mapping files -->

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

</session-factory>

</hibernate-configuration>

Thẻ <mapping> dùng include file hibernate-mapping.

## Hibernate Sessions

Chức năng chính của Session là để cung cấp các thao tác create, read và delete cho instances của các mapped entity classes. Instances có thể là một trong 3 states:

* transient: là một new instance của một persistent class mà không liên kết với một Session và không có đại diện trong database và không có giá trị identifier.
* persistent: một persistent instance có một đại diện trong database, một giá trị identifier và liên kết với một Session.
* detached: khi đóng Hibernate Session, persistent instance sẽ trở thành detached instance.

VD về một transaction và session:

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// do some work

...

tx.commit();

}

catch (Exception e) {

if (tx!=null) tx.rollback();

e.printStackTrace();

}

finally {

session.close();

}

Trong VD trên, factory là một SessionFactory instance.

**Session Interface Methods**

Session interface cung cấp một số methods. Bạn tham khảo trong Hibernate documentation.

## Hibernate Persistent Class

Hibernate lấy giá trị từ các thuộc tính của Java class và persist chúng vào một database table. Một mapping document giúp Hibernate xác định cách lấy từng giá trị trong classes và map chính xác vào từng fields trong table.

Persistent classes là các Java classes mà được mapped với các database tables. Các classes này nên tuân theo một số quy tắc đơn giản - mẫu Plain Old Java Object (POJO). Một số quy tắc chính như sau:

* Mọi Java classes mà sẽ được persisted thì cần một constructor mặc định.
* Mọi classes nên chứa một ID để dễ xác định trong Hibernate và database. Thuộc tính này maps tới cột khóa chính của bảng.
* Mọi thuộc tính mà sẽ được persisted nên được khai báo private và có các methods get/set định nghĩa theo kiểu JavaBean.

POJO có nghĩa object là một Java Object nguyên thủy, không phải Enterprise JavaBean (EJB).

VD về POJO class Employee:

public class Employee {

private int id;

private String firstName;

private String lastName;

private int salary;

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.firstName = fname;

this.lastName = lname;

this.salary = salary;

}

public int getId() {

return id;

}

public void setId( int id ) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName( String first\_name ) {

this.firstName = first\_name;

}

public String getLastName() {

return lastName;

}

public void setLastName( String last\_name ) {

this.lastName = last\_name;

}

public int getSalary() {

return salary;

}

public void setSalary( int salary ) {

this.salary = salary;

}

}

## Hibernate Mapping Files

Một ORM thường định nghĩa trong XML document. File mapping này chỉ thị cho Hibernate cách để map các defined classes với database tables.

Dựa vào VD ở trên, POJO Employee.

Có một bảng phù hợp với một persistence class. Persistent Employee tương ứng với table:

create table EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

Dựa trên class và table, ta định nghĩa mapping cho Hibernate:

<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

<meta attribute="class-description">

This class contains the employee detail.

</meta>

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

<generator class="native"/>

</id>

<property name="firstName" column="first\_name" type="string"/>

<property name="lastName" column="last\_name" type="string"/>

<property name="salary" column="salary" type="int"/>

</class>

</hibernate-mapping>

Có thể lưu file trên thành Employee.hbm.xml. Ý nghĩa các thành phần như sau:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| <hibernate-mapping> | Là root tag, chứa tất cả <class> elements. |
| <class> elements | Chỉ định mapping Java classes với database tables. Thuộc tính name chỉ định tên Java class. Thuộc tính table chỉ định tên bảng. |
| <meta> element | Optional. Dùng tạo mô tả class. |
| <id> element | maps thuộc tính unique ID trong class với khóa chính trong database table. name tham chiếu đến property trong class. column tham chiếu tên field trong table. type là hibernate mapping type. |
| <generator> element | Dùng tự phát sinh giá trị khóa chính. class=”native” cho phép Hibernate dùng identity, sequence hoặc hilo algorithm để tạo khóa chính. |
| <property> element | Dùng maps thuộc tính Java class với cột của database table. name tham chiếu đến property trong class. column tham chiếu tên field trong table. type là hibernate mapping type. |

Ngoài ra, còn nhiều attributes và elements khác bạn tham khảo trong documentation.

Trường hợp khóa chính composite, ta dùng composite-id như VD sau:

<class name="com.liferay.portal.model.impl.UserGroupGroupRoleImpl" table="UserGroupGroupRole">

<cache usage="read-write" />

<composite-id name="primaryKey" class="com.liferay.portal.service.persistence.UserGroupGroupRolePK">

<key-property name="userGroupId" type="com.liferay.portal.dao.orm.hibernate.LongType" />

<key-property name="groupId" type="com.liferay.portal.dao.orm.hibernate.LongType" />

<key-property name="roleId" type="com.liferay.portal.dao.orm.hibernate.LongType" />

</composite-id>

</class>

class của <composite-id> element chỉ đến Java class tương ứng với khóa chính kết hợp.

## Hibernate Mapping Types

Type được dùng trong mapping document không phải Java data types hay một loại SQL database type mà là Hibernate mapping types. Nó dùng để translate từ Java sang SQL data types và ngược lại.

Danh sách Hibernate mapping types.

|  |  |  |
| --- | --- | --- |
| **Mapping type** | **Java type** | **ANSI SQL type** |
| **Primitive types** | | |
| integer | int or java.lang.Integer | INTEGER |
| long | long or java.lang.Long | BIGINT |
| short | short or java.lang.Short | SMALLINT |
| float | float or java.lang.Float | FLOAT |
| double | double or java.lang.Double | DOUBLE |
| big\_decimal | java.math.BigDecimal | NUMERIC |
| character | java.lang.String | CHAR(1) |
| string | java.lang.String | VARCHAR |
| byte | byte or java.lang.Byte | TINYINT |
| boolean | boolean or java.lang.Boolean | BIT |
| yes/no | boolean or java.lang.Boolean | CHAR(1) ('Y' or 'N') |
| true/false | boolean or java.lang.Boolean | CHAR(1) ('T' or 'F') |
| **Date and time types** | | |
| date | java.util.Date or java.sql.Date | DATE |
| time | java.util.Date or java.sql.Time | TIME |
| timestamp | java.util.Date or java.sql.Timestamp | TIMESTAMP |
| calendar | java.util.Calendar | TIMESTAMP |
| calendar\_date | java.util.Calendar | DATE |
| **Binary and large object types** | | |
| binary | byte[] | VARBINARY (or BLOB) |
| text | java.lang.String | CLOB |
| serializable | java.io.Serializable implementation | VARBINARY (or BLOB) |
| clob | java.sql.Clob | CLOB |
| blob | java.sql.Blob | BLOB |
| **JDK-related types** | | |
| class | java.lang.Class | VARCHAR |
| locale | java.util.Locale | VARCHAR |
| timezone | java.util.TimeZone | VARCHAR |
| currency | java.util.Currency | VARCHAR |

## Hibernate O/R Mappings

### Collections Mappings

Nếu một entity/class có biến đặc biệt có giá trị thuộc dạng collection thì ta có thể dùng bất kỳ một collection interfaces nào có sẵn trong Java để map chúng. Hibernate có thể persist các instances của java.util.Map, java.util.Set, java.util.SortedMap, java.util.SortedSet, java.util.List, và mảng.

|  |  |
| --- | --- |
| **Collection type** | **Mapping and Description** |
| java.util.Set | Mapped bằng <set> element và initialized với java.util.HashSet. |
| java.util.SortedSet | Mapped bằng <set> element và initialized với java.util.TreeSet. |
| java.util.List | Mapped bằng <list> element và initialized với java.util.ArrayList. |
| java.util.Collection | Mapped bằng <bag> hoặc <ibag> element và initialized với java.util.ArrayList. |
| java.util.Map | Mapped bằng <map> element và initialized với java.util.HashMap. |
| java.util.SortedMap | Mapped bằng <map> element và initialized với java.util.TreeMap. |

Hibernate hỗ trợ <primitive-array> cho mảng các giá trị primitive types, <array> cho mọi mảng khác.

#### Hibernate Set Mappings

Một Set là Java collection không chứa duplicate element và nhiều nhất là một null element. Objects muốn thêm vào Set phải implement cả 2 methods: equals() và hashCode().

Bên dưới là VD cụ thể.

EMPLOYEE và CERTIFICATE có quan hệ 1-to-n. Định nghĩa 2 bảng như sau:

create table EMPLOYEE (

id INT NOT NULL auto\_increment,

...

PRIMARY KEY (id)

);

create table CERTIFICATE (

id INT NOT NULL auto\_increment,

...

employee\_id INT default NULL,

PRIMARY KEY (id)

);

Định nghĩa POJO của Employee:

import java.util.\*;

public class Employee {

private int id;

...

private Set certificates;

public Employee() {}

public Employee(...) {

...

}

...

public Set getCertificates() {

return certificates;

}

public void setCertificates( Set certificates ) {

this.certificates = certificates;

}

}

Thuộc tính certificates của Employee dùng java.util.Set.

Định nghĩa POJO của Certificate có implement equals() và hasCode():

public class Certificate {

private int id;

...

public boolean equals(Object obj) {

if (obj == null) return false;

if (!this.getClass().equals(obj.getClass())) return false;

Certificate obj2 = (Certificate)obj;

if((this.id == obj2.getId()) && (this.name.equals(obj2.getName())))

{

return true;

}

return false;

}

public int hashCode() {

int tmp = 0;

tmp = ( id + name ).hashCode();

return tmp;

}

}

Định nghĩa hibernate mapping file:

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

...

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

<key column="employee\_id"/>

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

</set>

...

</class>

<class name="Certificate" table="CERTIFICATE">

<meta attribute="class-description">

This class contains the certificate records.

</meta>

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

<generator class="native"/>

</id>

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

</class>

</hibernate-mapping>

<set> element để thiết lập relationship giữa Certificate và Employee. cascade=”all” để Hibernate thực hiện persist cho Certificate cùng số lần với Employee. name attribute được thiết lập cho biến kiểu Set trong parent class. Với mỗi biến Set, ta cần định nghĩa một set element riêng biệt trong mapping file.

<key> element là cột trong bảng CERTIFICATE là khóa ngoại tham chiếu đến bảng EMPLOYEE. Trong trường hợp này là employee\_id.

<one-to-many> element cho biết một Employee object liên kết với nhiều Certificate objects và mỗi Certificate object phải có một parent.

#### Hibernate SortedSet Mappings

Một SortedSet là một Java collection mà không chứa bất kỳ duplicate element nào và các elements được sắp xếp natural hoặc theo comparator đưa ra.

Một SortedSet được mapped bằng <set> element trong mapping và được initialized với java.util.-TreeSet. sort attribute có thể được thiết lập hoặc là một comparator hoặc là natural.

VD cũng dùng 2 đối tượng Employee và Certificate như Set Mapping.

Định nghĩa 2 bảng cũng như trên, có quan hệ 1-to-n.

Phần định nghĩa POJO của Employee sẽ có variable kiểu SortedSet:

public class Employee {

private int id;

...

private SortedSet certificates;

public Employee() {}

public Employee(...) {

...

}

public SortedSet getCertificates() {

return certificates;

}

public void setCertificates( SortedSet certificates ) {

this.certificates = certificates;

}

}

POJO của Certificate phải implements Comparable và hiện thực method compareTo:

public class Certificate implements Comparable <Certificate> {

...

public Certificate() {}

public Certificate(...) {

...

}

public int compareTo(Certificate that){

final int BEFORE = -1;

final int AFTER = 1;

if (that == null) {

return BEFORE;

}

Comparable thisCertificate = this.getName();

Comparable thatCertificate = that.getName();

if(thisCertificate == null) {

return AFTER;

} else if(thatCertificate == null) {

return BEFORE;

} else {

return thisCertificate.compareTo(thatCertificate);

}

}

}

Định nghĩa hibernate mapping files như sau:

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

...

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

<key column="employee\_id"/>

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

</set>

...

</class>

<class name="Certificate" table="CERTIFICATE">

...

</class>

</hibernate-mapping>

Ta dùng class MyClass để sắp xếp các phần tử trong SortetSet. Class này như sau:

import java.util.Comparator;

public class MyClass implements Comparator<Certificate>{

public int compare(Certificate o1, Certificate o2) {

final int BEFORE = -1;

final int AFTER = 1;

/\* To reverse the sorting order, multiple by -1 \*/

if (o2 == null) {

return BEFORE \* -1;

}

Comparable thisCertificate = o1.getName();

Comparable thatCertificate = o2.getName();

if(thisCertificate == null) {

return AFTER \* 1;

} else if(thatCertificate == null) {

return BEFORE \* -1;

} else {

return thisCertificate.compareTo(thatCertificate) \* -1;

}

}

}

Class này hiện thực Comparator và hiện thực method compare. Thực hiện xong các bước trên, ta có thể dùng chúng để viết code ứng dụng.

#### Hibernate List Mappings

Một List là một Java collection mà chứa các elements theo chuỗi và cho phép duplicate elements. User có thể truy cập các elements qua integer index và tìm kiếm các elements trong list. So sánh các elements dùng method equals.

VD cũng dùng 2 đối tượng Employee và Certificate như Set Mapping.

Định nghĩa 2 bảng cũng như trên, có quan hệ 1-to-n.

Phần định nghĩa POJO của Employee sẽ có variable kiểu List:

import java.util.\*;

public class Employee {

...

private List certificates;

public Employee() {}

public Employee(...) {

...

}

...

public List getCertificates() {

return certificates;

}

public void setCertificates( List certificates ) {

this.certificates = certificates;

}

}

POJO của Certificate là JavaBean bình thường, không implements.

Định nghĩa hibernate mapping file:

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

...

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

<key column="employee\_id"/>

<list-index column="idx"/>

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

</list>

...

</class>

<class name="Certificate" table="CERTIFICATE">

...

</class>

</hibernate-mapping>

<list> element dùng thiết lập relationship giữa Certificate và Employee. <key> element là cột trong bảng CERTIFICATE là khóa ngoại liên kết với bảng EMPLOYEE. <list-index> element lưu giữ vị trí của element, bắt đầu từ zero.

#### Hibernate Bag Mappings

Một Bag là một Java collection chứa các elements không theo sequence nhưng cho phép duplicate elements. Một Collection được mapped với <bag> element và được initialized với java.util.ArrayList.

VD cũng dùng 2 đối tượng Employee và Certificate như Set Mapping.

Định nghĩa 2 bảng cũng như trên, có quan hệ 1-to-n.

Phần định nghĩa POJO của Employee sẽ có variable kiểu Collection:

import java.util.\*;

public class Employee {

...

private Collection certificates;

public Employee() {}

public Employee(...) {

...

}

...

public Collection getCertificates() {

return certificates;

}

public void setCertificates( Collection certificates ) {

this.certificates = certificates;

}

}

POJO của Certificate là JavaBean bình thường, không implements.

Định nghĩa hibernate mapping file:

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

...

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

<key column="employee\_id"/>

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

</bag>

...

</class>

<class name="Certificate" table="CERTIFICATE">

...

</class>

</hibernate-mapping>

<bag> element được dùng để thiết lập relationship giữa Certificate và Employee.

#### Hibernate Map Mappings

Một Map là một Java collection chứa các phần tử trong các cặp key-value và không cho phép duplicated elements. Map interface cung cấp 3 khung nhìn collection: một tập các keys, collection of values, hoặc tập các key-value mappings.

Một Map được mapped với <map> element và unordered map có thể được initialized với java.util.HashMap.

VD cũng dùng 2 đối tượng Employee và Certificate như Set Mapping.

Định nghĩa 2 bảng cũng như trên, có quan hệ 1-to-n.

Phần định nghĩa POJO của Employee sẽ có variable kiểu Collection:

chuaxong

### Association Mappings

Việc mapping các associations giữa các entity classes và các relationships giữa các tables là cốt lõi của ORM. Có 4 cách mapping cho relationship giữa các objects. Mapping có thể theo một chiều hoặc hai chiều.

|  |  |
| --- | --- |
| **Mapping type** | **Description** |
| Many-to-One | Mapping many-to-one relationship using Hibernate |
| One-to-One | Mapping one-to-one relationship using Hibernate |
| One-to-Many | Mapping one-to-many relationship using Hibernate |
| Many-to-Many | Mapping many-to-many relationship using Hibernate |

#### Many-to-One

#### One-to-One

##### Tieu de 5

##### Tieu de 5

#### One-to-Many

#### Many-to-Many

### Component Mappings

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate annotation

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Query Language (HQL)

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Criteria Queries

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Native SQL

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Caching

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Batch Processing

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Hibernate Interceptors

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

## Heading 1.1 fjsdlas jrlwe flwefjlw flwjfwf ừlwfjewlf ừlwjlefw ưlw

### fdfa

### dfjsalfds fdfsa rưejrlw rưl sfjalf ưlflw fwlf w flwjrlw sdls fwefw

#### ffdslffsfsla fsjdfal lư flw fwjwl flwf lựlspa ầ fjslaf ừljflwefjw flwf ưlfwlfwl

#### fdjfslaf sfjsf fjdsflsfsa fjsdlf sfarw dfla fjdsla ljrwe lfjwl fwwrwl

##### Tieu de 5

##### Tieu de 5

### fjdfka fdjsal djfsla sjflsf álf ậlrjlw lằeopaf fsjdfapruw fajfowf ẻw

#### dslfas fjdsaljflew fwfwf jwfelw

### dfjsal jewlrewrw

fdsfakl fsfda

fsa fs ffsa fds jdfsa fsalf sálf sấl

fsdafsa fsdafsad

fdsffaf sdfad

fdsa

fsa coding titleContent coding example