**Mapping List**

1 List trong java collection lưu trữ các thành phần cho phép trùng nhau giống như e1.equal(e2) và chấp nhận có nhiều null value trong nó.

**Tạo bảng db :**

CREATE TABLE productlines(

productLine VARCHAR(50) NOT NULL,

textDescription VARCHAR(400),

htmlDescription TEXT,

image VARCHAR(500),

PRIMARY KEY(productLine)

) ENGINE = INNODB DEFAULT CHARSET = utf8;

CREATE TABLE products(

productCode VARCHAR(15) NOT NULL,

productName VARCHAR(70) NOT NULL,

productLine VARCHAR(50) DEFAULT NULL,

productScale VARCHAR(10) NOT NULL,

productVendor VARCHAR(50) NOT NULL,

productDescription TEXT NOT NULL,

quantityInStock SMALLINT(6) NOT NULL,

buyPrice DOUBLE NOT NULL,

MSRP DOUBLE NOT NULL,

PRIMARY KEY (productCode),

CONSTRAINT product\_productline\_fk FOREIGN KEY (productLine) REFERENCES productlines(productLine)

) ENGINE = INNODB DEFAULT CHARSET = utf8;

Giả sử có 2 bảng db productlines và products có quan hệ 1 nhiều , trong đó 1 productline có thể có nhiều products liên kết với nó.

public class ProductLines {

private String producLine;

private String textDescription;

private String htmlDescription;

private String image;

private List product;

**Định nghĩa các POJO class :**

Các POJO class định nghĩa bình thường với các getter, setter và constructer ứng với class đó. Trong class ProductLines sẽ định nghĩa 1 List products như trên ứng với 1 ProductLine có thể có nhiều products.

**Cấu hình Hibernate mapping xml file :**

*<?***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="model.ProductLines" table="productlines"**>  
 <**meta attribute="class-description"**>  
 This class contains the employee detail.  
 </**meta**>  
 <**id name="productLine" type="string" column="productLine"**>  
 </**id**>  
 <**list name="product" cascade="all"**>  
 <**key column="productLine"**/>  
 <**one-to-many class="model.Products"**/>  
 </**list**>  
 <**property name="textDescription" column="textDescription" type="string"**/>  
 <**property name="htmlDescription" column="htmlDescription" type="string"**/>  
 <**property name="image" column="image" type="string"**/>  
 </**class**>  
</**hibernate-mapping**>

**Cấu hình bằng annotation :**

@OneToMany(fetch = FetchType.LAZY, mappedBy = "productLine", cascade = "all")

public List <Products> products;

Trong cấu hình trên, <list> để khai báo List và đặt quan hệ giữa Products và ProductLines. Ta dùng thuộc tính **casade** trong <list> để nói với Hibernate rằng lưu các đối tượng Product cùng lúc với đối tượng ProductLine. Thuộc tính **name** để định nghĩa biến của List trong class cha, trong trường hợp này là *product*.

<key> là 1 column trong bảng products, column đó chứa khóa ngoại của bảng cha productLines.

<one-to-many> chỉ ra 1 đối tượng ProductLines quan hệ tới nhiều đối tượng Products.

Đối với annotation, chức năng của mappedBy sẽ giống như <key>.

**Mapping Bag : one-to-many by Bag**

Tương tự như **List** , nhưng **one-to-many by bag** trong file xml sẽ thay thẻ <list> bằng thẻ <bag> và các thuộc tính giống hệt <list>. Đối với annotation thì không có gì thay đổi.

**Mapping Set : one-to-many by Set**

Tương tự như **List** , nhưng **one-to-many by set** trong file xml sẽ thay thẻ <list> bằng thẻ <set> và các thuộc tính giống hệt <list>. Đối với annotation thì không có gì thay đổi.

**Many-to-many : mapping by map**

**Tạo bảng db :**

CREATE TABLE employee (

id INT NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(20) DEFAULT NULL,

last\_name VARCHAR(20) DEFAULT NULL,

salary INT,

PRIMARY KEY (id)

) ENGINE = INNODB DEFAULT CHARSET = utf8;

CREATE TABLE certificate (

id INT NOT NULL AUTO\_INCREMENT,

certificate\_name VARCHAR(30) DEFAULT NULL,

PRIMARY KEY (id)

) ENGINE = INNODB DEFAULT CHARSET = utf8;

CREATE TABLE emp\_cert (

employee\_id INT NOT NULL,

certificate\_id INT NOT NULL,

PRIMARY KEY (employee\_id, certificate\_id),

CONSTRAINT emp\_cert\_emp\_fk FOREIGN KEY (employee\_id) REFERENCES employee(id),

CONSTRAINT emp\_cert\_cert\_fk FOREIGN KEY (certificate\_id) REFERENCES certificate(id)

) ENGINE = INNODB DEFAULT CHARSET = utf8;

**Định nghĩa lớp POJO :**

private int id;

private String firstName;

private String lastName;

private int salary;

private Map certificates;

private int id;

private String name;

private Map employees;

Tạo các getter, setter và constructer như bình thường.

**Cấu hình Hibernate mapping xml file :**

<**map name="certificates" cascade="all" table="emp\_cert"**>  
 <**key column="employee\_id"**></**key**>  
 <**index column="certificate\_type" type="string"**></**index**>  
 <**many-to-many column="certificate\_id" class="model.Certificate"**/>  
</**map**>

**Cấu hình bằng annotation :**

@ManyToMany(cascade = CascadeType.***ALL***)  
@JoinTable(name = **"emp\_cert"**, joinColumns = @JoinColumn(name = **"employee\_id"**),  
 inverseJoinColumns = @JoinColumn(name = **"certificate\_id"**))  
@MapKeyJoinColumn(name = **"certificate\_type"**)  
**private** Map<String, Certificate> **certificates**;