

Hướng Dẫn Chức Năng CRUD Với EJB3 Sử Dụng NetBeans 6.9.1 Server GlassFish 3.1 và JSF 2.0

Bài viết hướng dẫn thao tác bằng EJB 3 để thêm , xóa , sửa dữ liệu tương tác với CSDL.

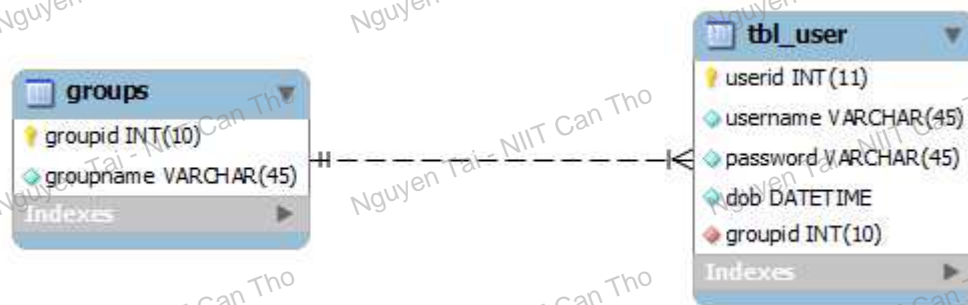
Một số yêu cầu của bài viết:

- Đang tìm hiểu EJB3, những người có khái niệm cơ bản về EJB
- Đang tìm hiểu hoặc đã biết về JSF (1.2 hay 2.0, có thể sử dụng các framework hỗ trợ JSF như richfaces. . .)
- Một số kiến thức cơ bản với NetBeans IDE.

Chuẩn bị:

Trước khi thực hiện các thao tác, chúng ta cần có một cơ sở dữ liệu. Ở đây tôi sử dụng hệ quản trị CSDL MySQL để thiết lập Database.

Tôi đã tạo 1 Database với tên là EJB gồm 2 bảng : Tbl_user và groups. Có quan hệ Nhiều – Một (Tbl_user : Many – Groups: One). Chúng sẽ có mô hình quan hệ như sau:



Hình 1 – Mô quan hệ Một – Nhiều giữa 2 bảng: Groups và tbl_user.

Sau khi đã có CSDL và xác định được mối quan hệ của các bảng, chúng ta sẽ dùng Netbeans IDE để tạo kết nối giữa Netbeans với hệ quản trị CSDL.

Tạo kết nối giữa NetBeans IDE và MySQL:

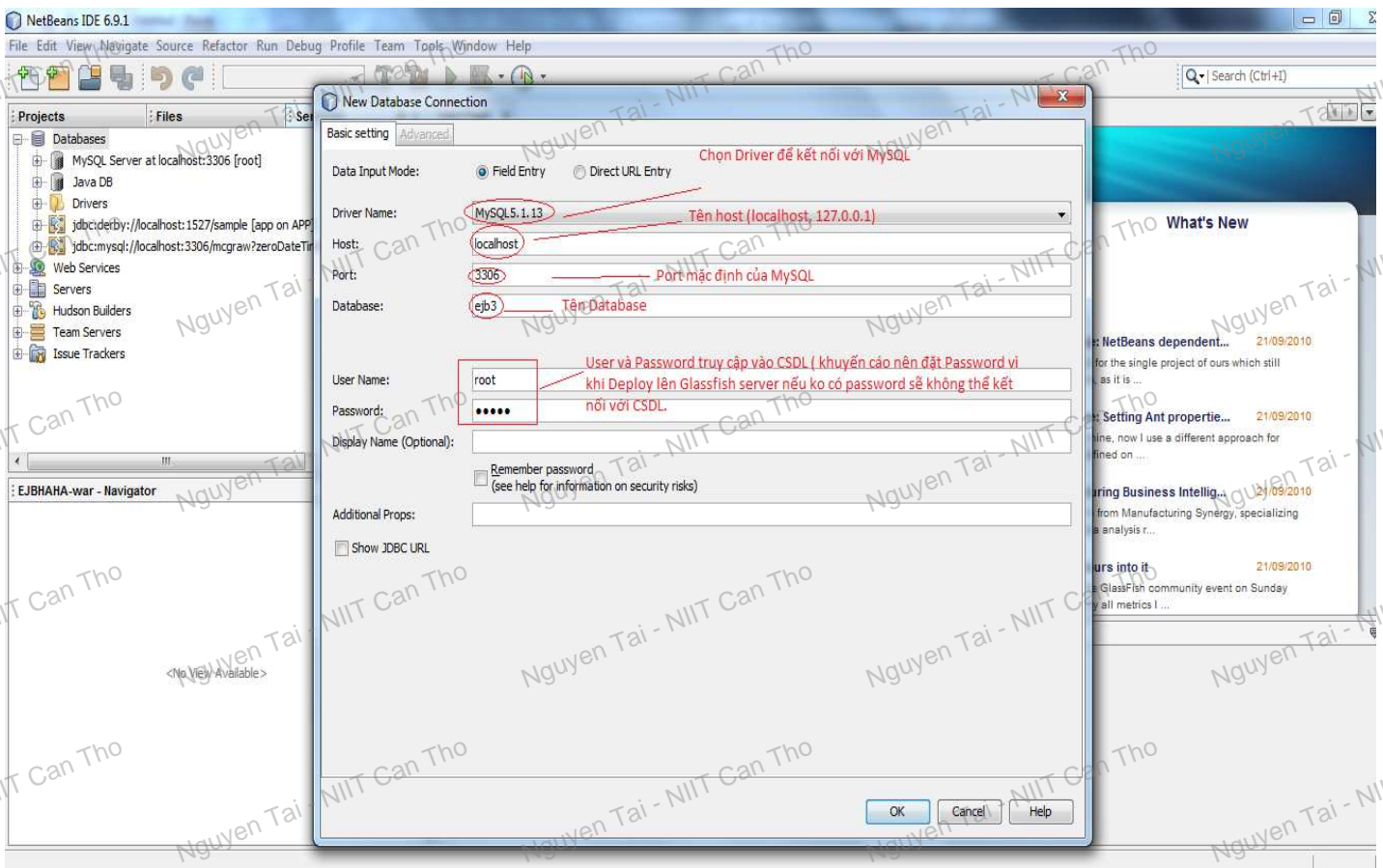
- Khởi động Netbeans IDE
- Chúng ta tiếp tục chọn Tabs Services
- Ở đây tôi đã tạo kết nối giữa Netbeans IDE với MySQL nên tôi sẽ không tạo lại nữa. Nếu bạn chưa tạo bạn có thể tạo 1 kết nối vào MySQL

- Bước tiếp theo là tạo kết nối giữa Netbeans IDE và Database mà ta cần thao tác như hình sau:
- Chuột phải vào Database → New Connection.



Hình 2 – Tạo kết nối mới với Database từ Netbeans

- Sau khi chọn New Connection . . . một cửa sổ hiện thông tin các tham số cần thiết để thiết lập kết nối với Database cần kết nối.

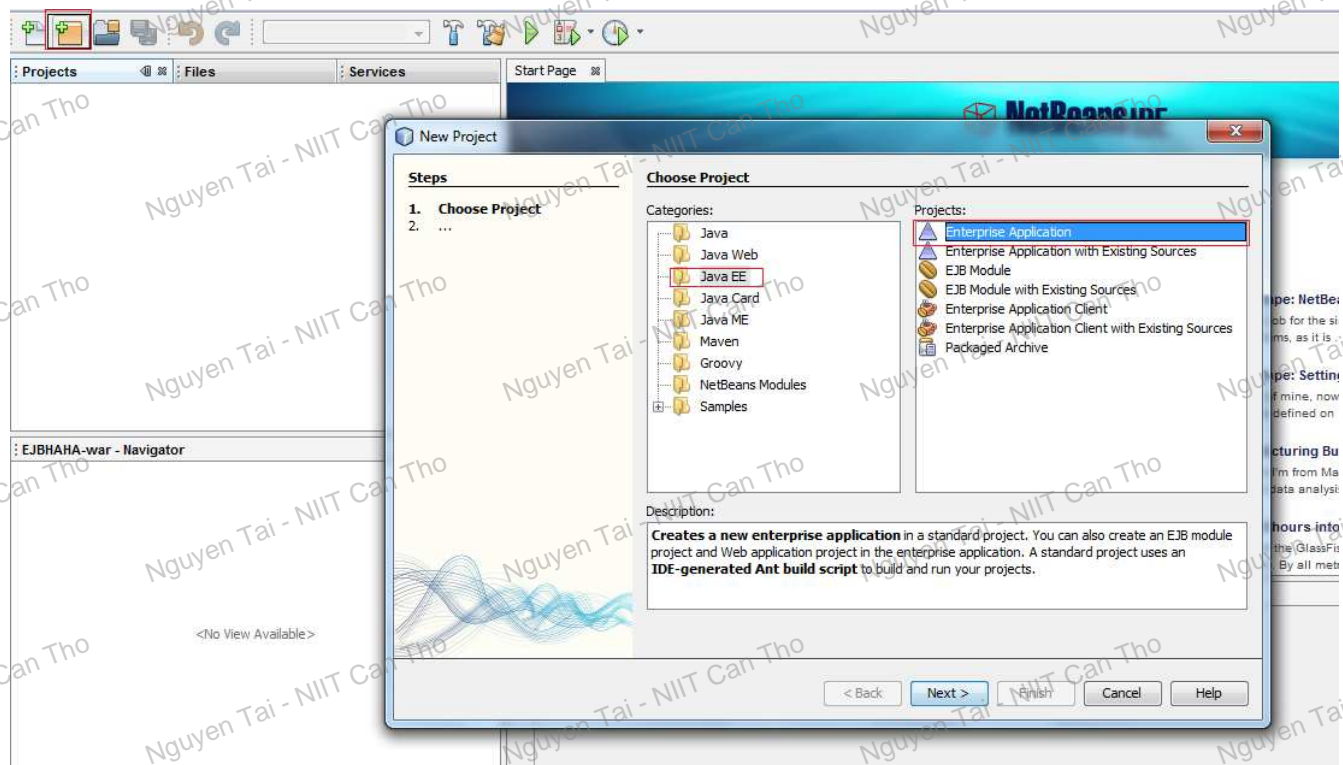


Hình 3 – Thiết lập thông số kết nối giữa Netbeans vào Database

- Các thông số trong hình là:
 - Driver Name: Chọn Driver thích hợp với từng loại CSDL ở đây tôi chọn Driver MySQL 5.1.13 (mặc định Driver của Netbeas là 5.0.6)
 - Host: Thông tin tên server , do dùng máy cá nhân tạo nên để là localhost hoặc 127.0.0.1
 - Port: cổng kết nối mặc định của MySQL là 3306, nếu lúc cài đặt bạn có thay đổi cổng vui lòng điền thông tin thích hợp.
 - Database: tên CSDL trong MySQL cần kết nối.

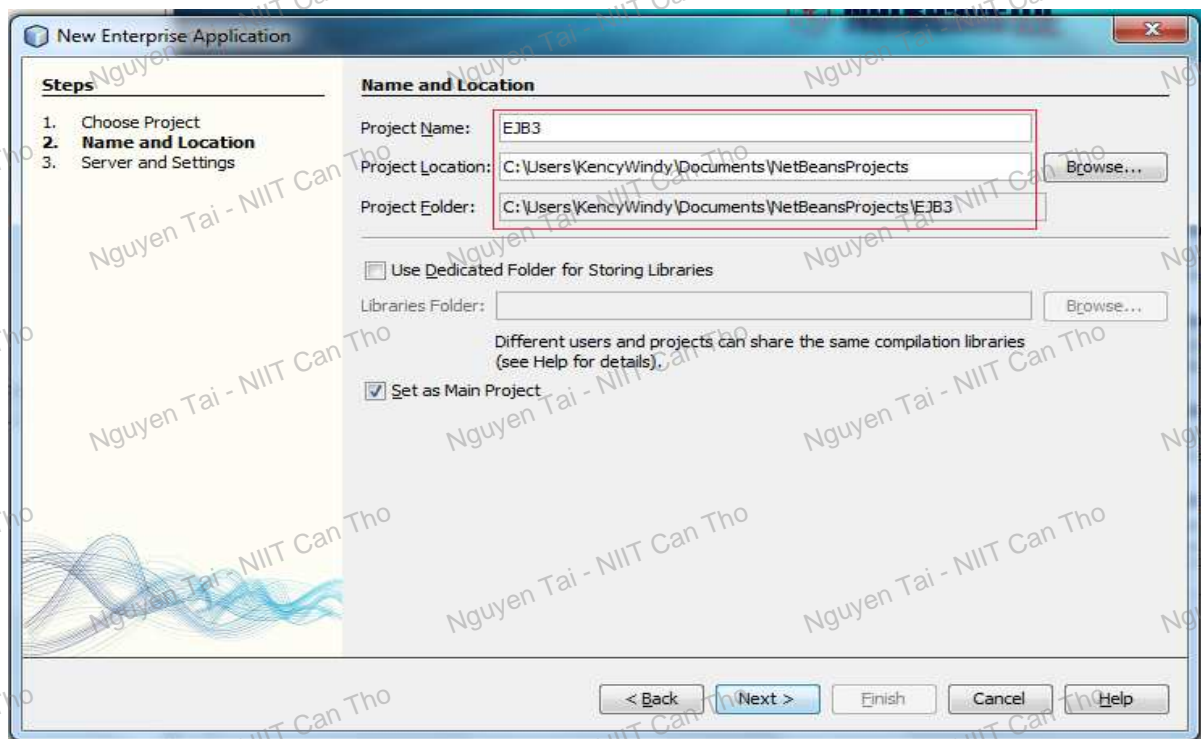
- Username & Password: tên đăng nhập và mật khẩu truy cập vào MySQL
- Đến đây bạn có thể bấm Ok để hoàn tất quá trình tạo kết nối giữa Netbeans và MySQL.

Tiếp theo chúng ta sẽ tạo Project EJB. Chọn New Project → Java EE → Enterprise Application:



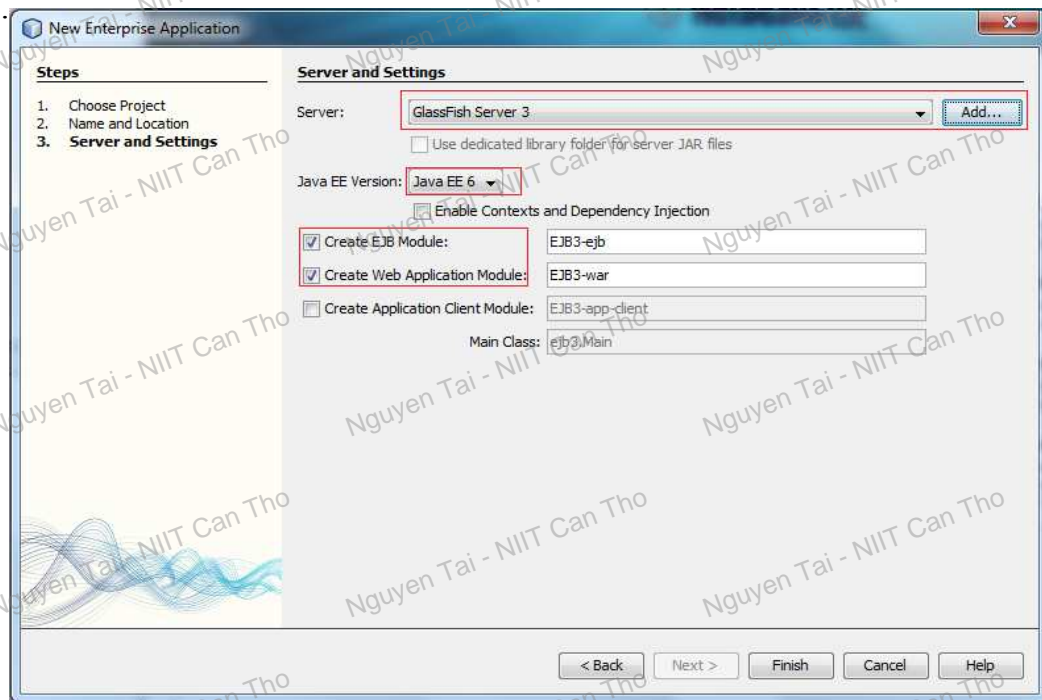
Hình 4 – Tạo mới một Project EJB3 (1)

Tiếp tục bấm Next → đặt Tên Project, ở đây tôi đặt tên là EJB3. Set vị trí Project và Folder chủ Project. Ở đây tôi để mặc định.



Hình 5 – Tạo mới một Project EJB 3 (2)

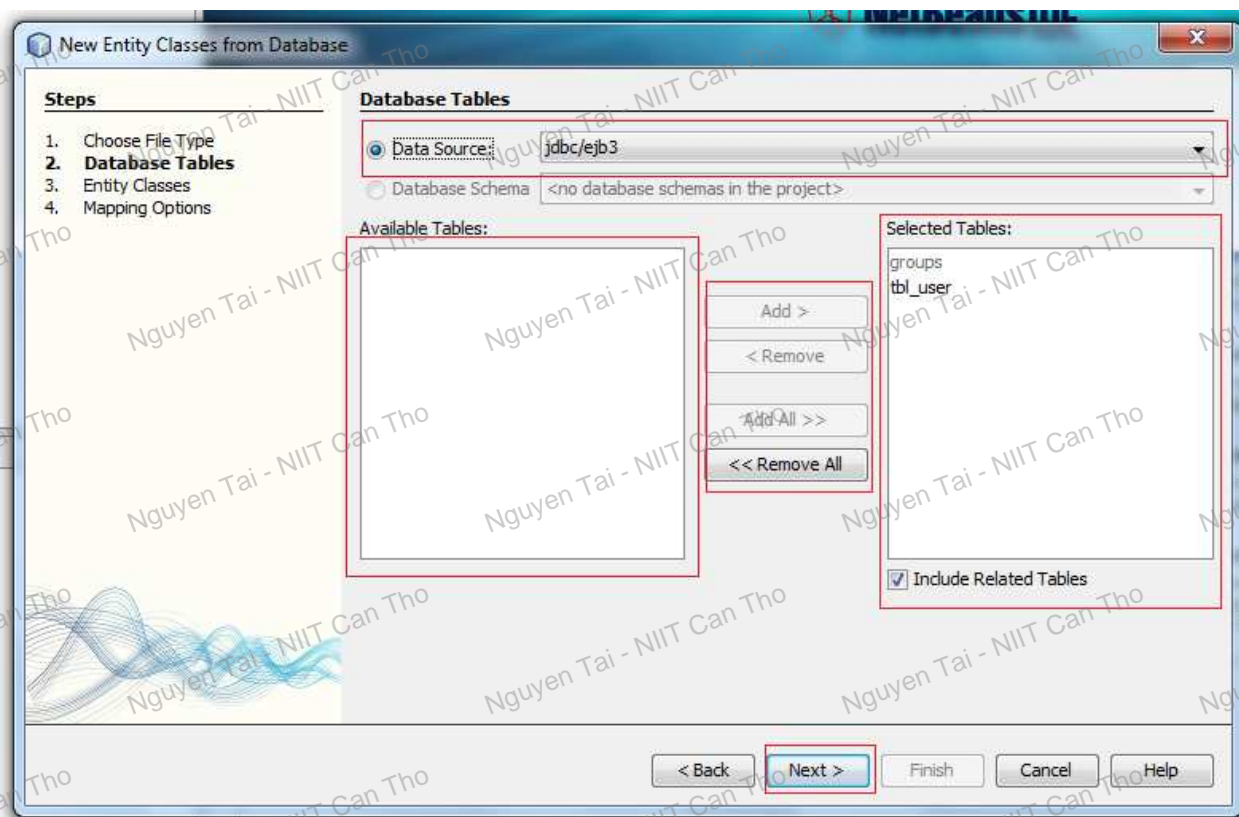
Tiếp tục bấm next và chọn Server GlassFish 3 → Java EE Version 6 (hoặc 5) → Mặc định Netbeans sẽ Check vào tạo 2 module là Web Module và EJB Module. Bấm Finish để kết thúc.



Hình 6- Tạo mới một Proeject EJB3 (3)

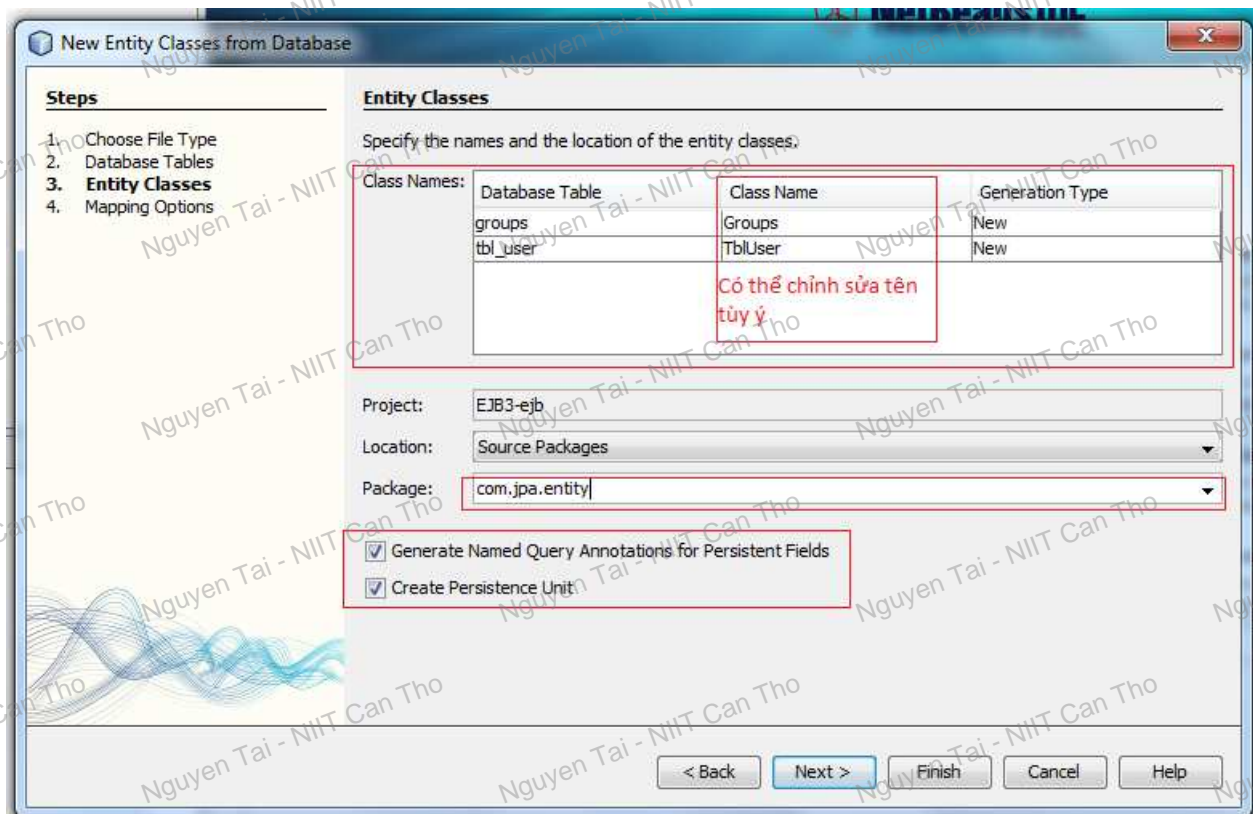
Bây giờ việc tiếp theo là viết các Class Entity để ánh xạ xuống CSDL, ở đây ta có 2 bảng tbl_user và groups vì thế chúng ta tạo 2 class Entity. Netbeans IDE hỗ trợ chúng ta tools để đưa table từ CSDL chuyển thành các class Java.

Đầu tiên, chuột phải vào EJB module → Chọn Entity Class From Database ... một cửa sổ hiện lên. Ở ô DataSource bạn chọn New Data Source , điền tên và chọn kết nối cần thiết sau đó bấm OK. Ở đây tôi đặt tên là jdbc/ejb3 và chọn kết nối với netbeans vừa tạo. Sau khi đã chỉ ra Datasource việc tiếp theo là ở ô vuông Available Table bạn chọn những table cần chuyển thành Entity Class chọn Add> để đưa sang ô Selected Table. Ở đây tôi chọn Add All> để Add 2 Table hiện tại thành Entity Class. Bấm Next để tiếp tục.



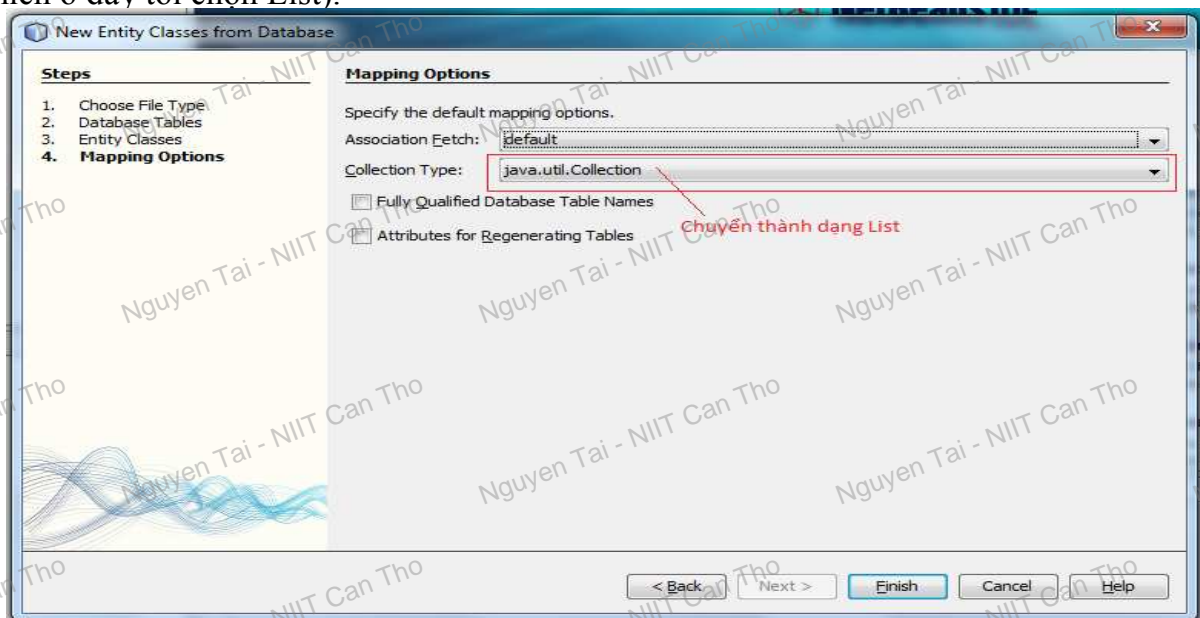
Hình 6 – Tạo Entity Class từ CSDL(1)

Ở cửa sổ tiếp theo bạn sẽ cấu hình thông số như là Tên class, loại sinh ra, đặc biệt là bạn cần tạo 1 package để chứa các Class này. Ở đây tôi tạo 1 package với tên là : com.jpa.entity . Chú ý nên giữ nguyên 2 dấu check mặc định ở giai đoạn này. Hai dấu check vào 2 ô này có ý nghĩa là hệ thống sẽ tự tạo các NameQuery thực hiện truy vấn dựa trên các Field có trong Table đó. Dấu check thứ 2 Tạo Persistence Unit cho các Class. Persistence Unit cơ bản dùng để nhóm các Entity trong ứng dụng lại với nhau.



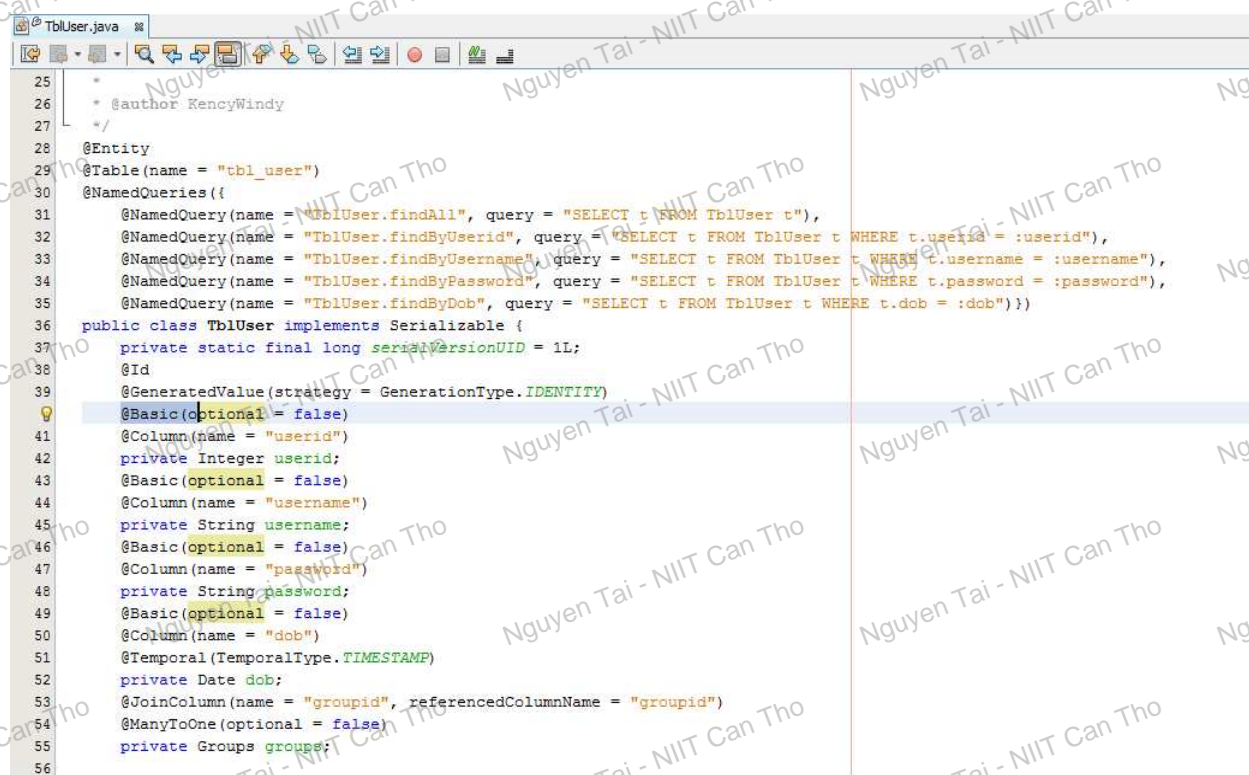
Hình 7 – Tạo Entity Class từ CSDL(2)

Tiếp tục chọn Next ➔ Tùy chỉnh Collect Type là List (bạn có thể chọn theo type mà bạn thích ở đây tôi chọn List).



Hình 8– Tạo Entity Class từ CSDL(3)

Sau khi tạo xong Package chứa các Class Entity xuất hiện, mở thử 1 trong 2 class các bạn sẽ thấy các câu NameQuery được tạo và các câu lệnh ánh xạ các thuộc tính trong Class xuống các cột trong CSDL như `@Column(name="username")`.



```
25
26 *
27 * @author KencyWindy
28 */
29 @Entity
30 @Table(name = "tbl_user")
31 @NamedQueries({
32     @NamedQuery(name = "TblUser.findAll", query = "SELECT t FROM TblUser t"),
33     @NamedQuery(name = "TblUser.findById", query = "SELECT t FROM TblUser t WHERE t.userid = :userid"),
34     @NamedQuery(name = "TblUser.findByUsername", query = "SELECT t FROM TblUser t WHERE t.username = :username"),
35     @NamedQuery(name = "TblUser.findByPassword", query = "SELECT t FROM TblUser t WHERE t.password = :password"),
36     @NamedQuery(name = "TblUser.findByDob", query = "SELECT t FROM TblUser t WHERE t.dob = :dob"))
37 }
38
39 public class TblUser implements Serializable {
40     private static final long serialVersionUID = 1L;
41     @Id
42     @GeneratedValue(strategy = GenerationType.IDENTITY)
43     @Basic(optional = false)
44     @Column(name = "userid")
45     private Integer userid;
46     @Basic(optional = false)
47     @Column(name = "username")
48     private String username;
49     @Basic(optional = false)
50     @Column(name = "password")
51     private String password;
52     @Basic(optional = false)
53     @Column(name = "dob")
54     @Temporal(TemporalType.TIMESTAMP)
55     private Date dob;
56     @JoinColumn(name = "groupid", referencedColumnName = "groupid")
57     @ManyToOne(optional = false)
58     private Groups groups;
```

Hình 9– Entity Class

Như thế là ta đã đưa Table chuyển thành Entity Class việc tiếp theo là thao tác với CSDL sử dụng Session Bean. Chuột phải vào EJB module. Chọn Session Bean, đặt tên là UserBean, nằm trong Package com.DAO. Chọn Stateless Bean và không cần Interface (điểm mới của EJB 3.1). Sau đó click Finish để hoàn tất việc tạo Stateless Bean, hệ thống sẽ tự động sinh ra cho bạn 1 Class Stateless Bean và Interface sẽ sử dụng Annotation là `@LocalBean` xác định bean này là Bean cục bộ, nếu muốn remote bạn có thể tạo Interface cho nó, do bài viết mang tính chất demo tôi chỉ dừng lại ở đây.

Tiếp theo sau khi đã tạo thành công class UserBean, mở class này lên tiếp tục trong vùng màn hình code chuột phải chọn Persistence ➔ user EntityManager. EntityManager sẽ được tạo ra với Annotation Persistence Context để đưa Container quản lý EntityManager vào class này.

New Session Bean

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

EJB Name:

Project:

Location:

Package:

Session Type:

☒ Stateless

☐ Stateful

☐ Singleton

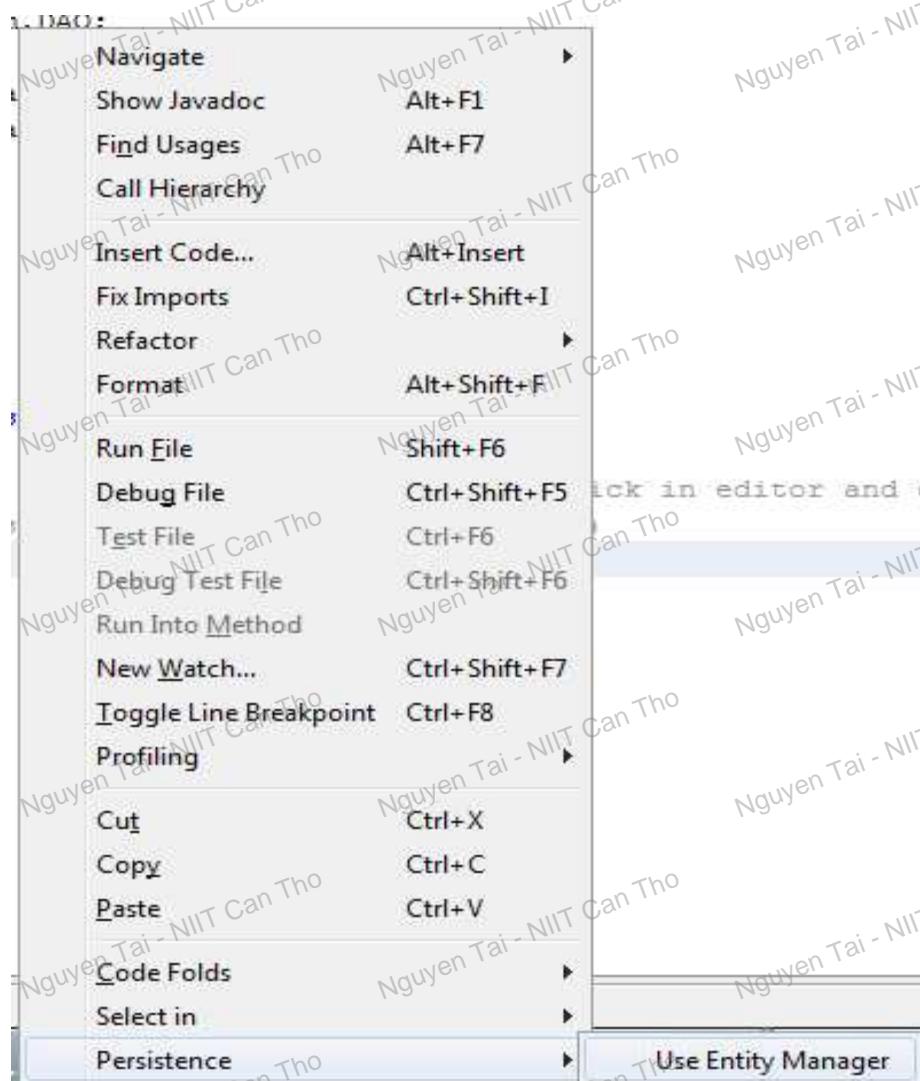
Create Interface:

☐ Local

☐ Remote

< Back Next > **Finish** Cancel Help

Hình 10- Tạo SessionBean



Hình 11– Tạo EntityManager

Sau khi đã tạo EntityManager việc tiếp theo là viết các phương thức để thực hiện Thêm – Sửa – Xóa

```
//Tra ve danh sach toan bo user
public List<TblUser> retrieveAllUsers() {

    return em.createNamedQuery("TblUser.findAll").getResultList();
}

//thuc hien add 1 user moi
public TblUser persistUser(TblUser user){

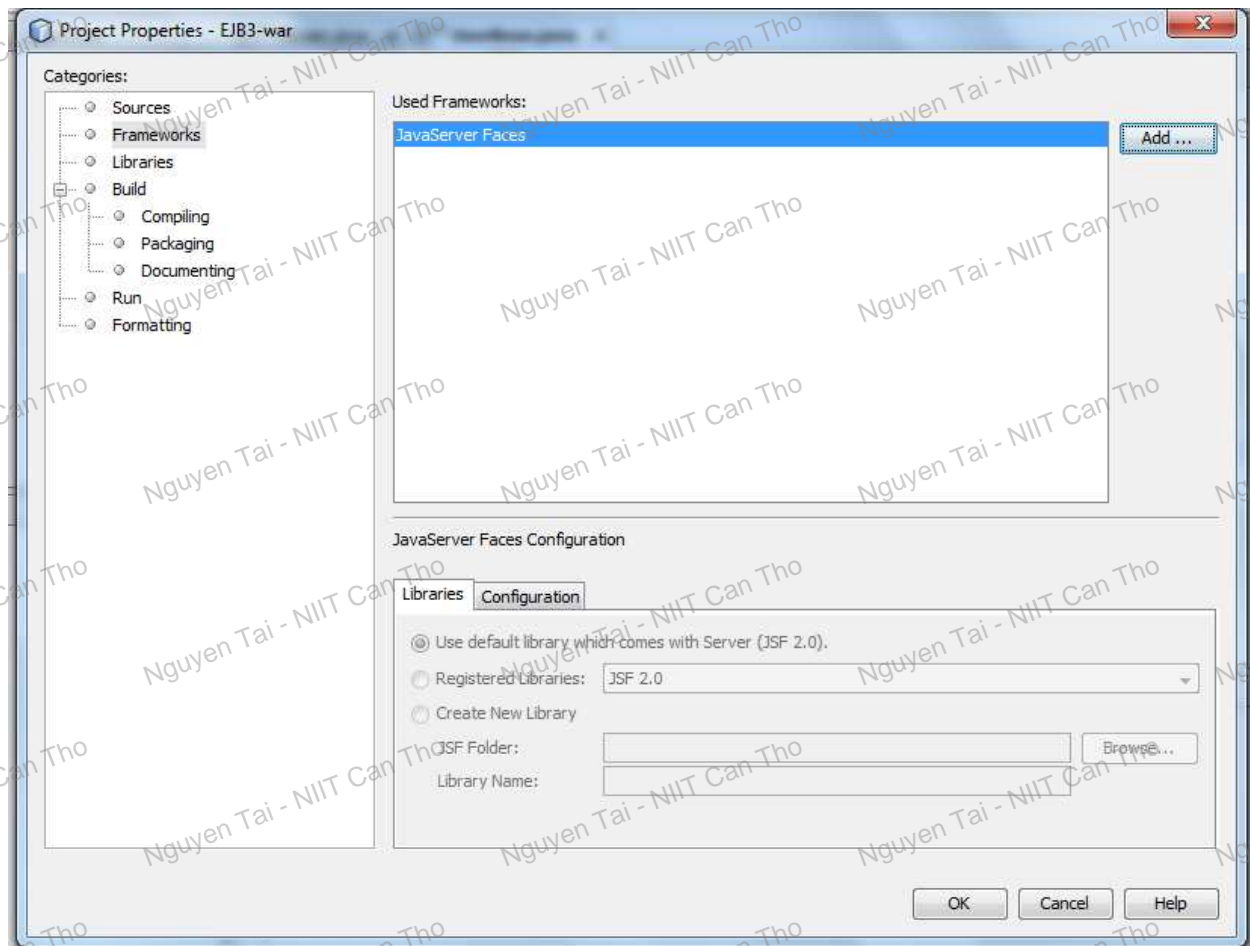
    em.persist(user);
    return user;
}

//Thuc hien cap nhat 1 record user trong CSDL
public TblUser updateUser(TblUser user){
    return em.merge(user);
}

//Thuc xoa 1 record user trong CSDL
public void deleteUser(TblUser user){
    em.remove(em.merge(user));
}
}
```

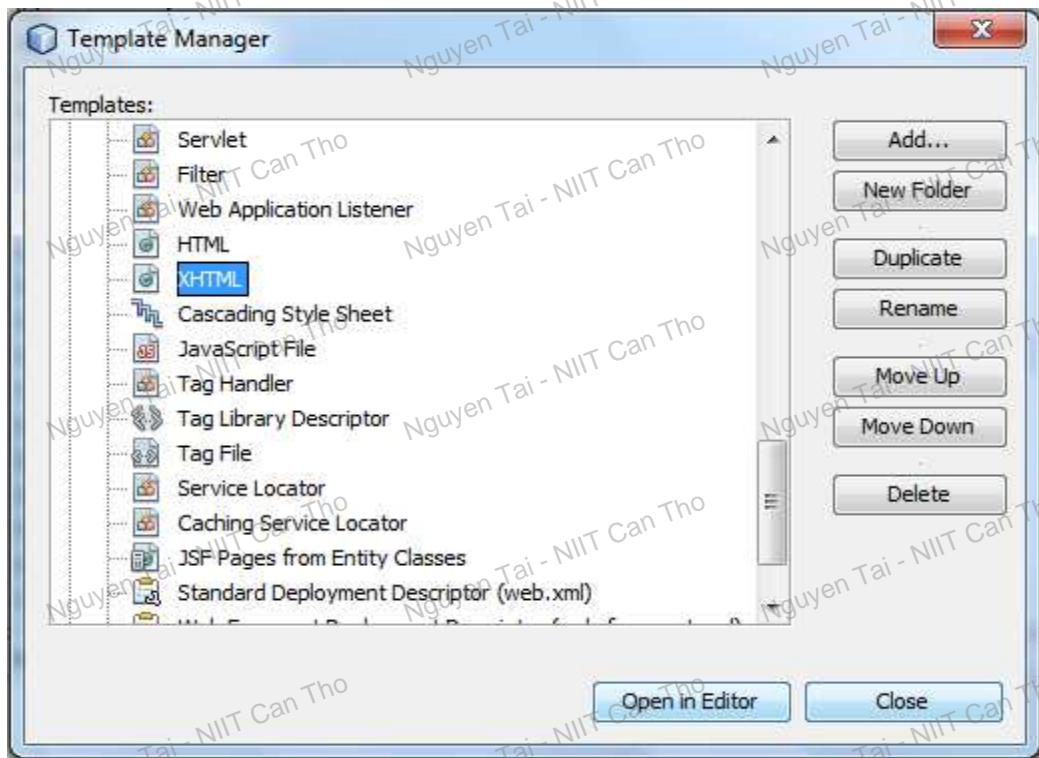
Sau khi đã viết xong các phương thức cần thiết để thực hiện việc thêm sửa xóa ta cần tạo 1 class điều hướng các phương thức này ra JSF. Ở JSF các thông tin đưa ra View thường được chứa trong back bean (trừ 1 số giá trị resource được ghi vào file Bundle.properties). Để tạo back bean cho JSF điều đầu tiên ta cần add Framework JSF 2.0 vào module Web EJB-war.

Chuột phải vào module Web chọn Properties → chọn Framework → chọn Add → Select Java Server Faces → click Ok → Click Ok để add Framework vào Web module.



Hình 12– Thêm Framework JSF 2.0 vào Web Module.

Tiếp tục ta sẽ tạo Back Bean để đưa dữ liệu ra ngoài JSF, trước khi thực hiện điều này ta cần edit 1 tí trang XHTML. Vào Tools Menu → Templates → Web → XHTML → Click chọn Open in Editor.



Hình 12– Chỉnh sửa templates XHTML.

Sau khi open in Editor, template XHTML sẽ được hiển thị. Việc tiếp theo là xóa toàn bộ và thay bằng

```
<?xml version="1.0" encoding="${encoding}"?>
<#assign licenseFirst = "<!--">
<#assign licensePrefix = "">
<#assign licenseLast = "-->">
<#include "../Licenses/license-${project.license}.txt">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <meta http-equiv="Content-Type" content="text/html;
charset=${encoding}"/>
    <title>TODO supply a title</title>
  </h:head>
  <h:body>
    TODO write content
  </h:body>
</html>
```


Tiếp tục save lại như thể đã thay thế Template của XHTML theo ý ta muốn. Tiếp theo vào file Web.xml chọn Tab XML tìm đoạn:

```
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
```

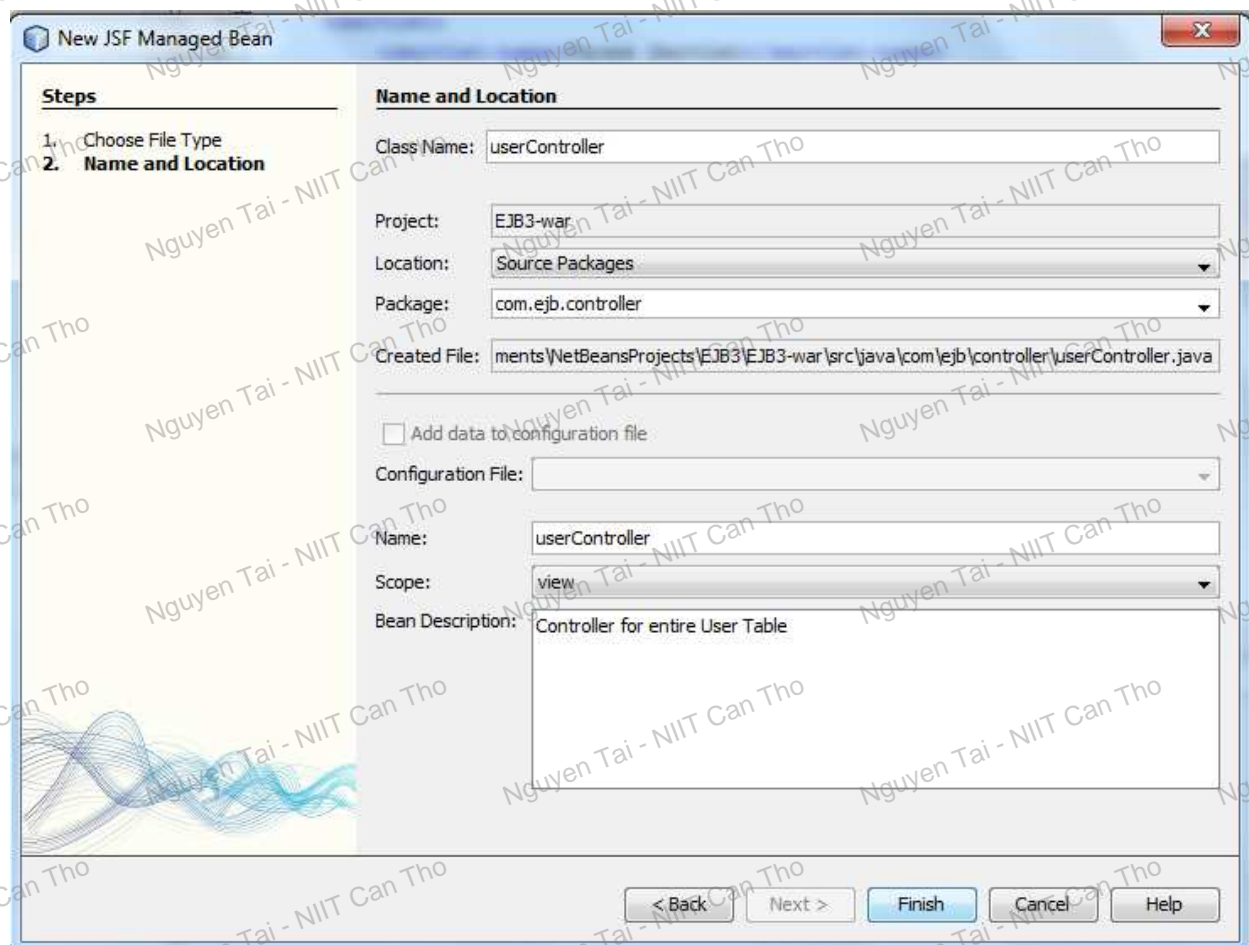
Thay bằng

```
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

Bây giờ đã xong cấu hình cho JSF, tiếp tục ta sẽ tạo Back Bean cho JSF. Chuột phải vào Web module chọn New . . . Chọn JSF Managed Bean (nếu không tìm thấy chọn Others → JavaServerFace → Khung bên phải JSF Managed Bean).

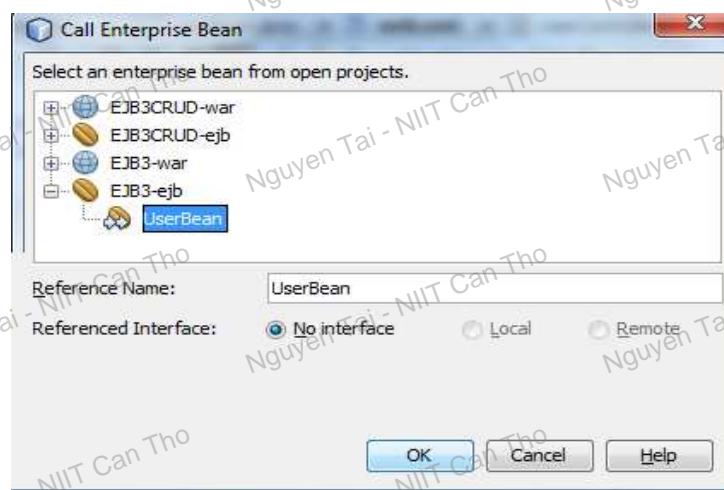
Đặt tên Bean này là UserController → nằm trong Package com.ejb.Controller → Scope chọn Session.

Sau khi click Finish một Back bean được tạo ra với 2 Annotation trước tên class mà @ManagedBean và @SessionScope



Hình 13– Tạo JSF Managed Bean.

Tiếp theo là đưa EJB module vào Back bean , chuột phải vào vùng code chọn Insert Code
 → Chọn tiếp Call Enterprise Bean → một cửa sổ xuất hiện chọn UserBean → No Interface → Click ok (do lúc này ta đã không tạo Interface nên ở đây ta chọn no interface).



Hình 14– Gọi 1 EJB Bean.

Sau khi gọi 1 EJB Bean từ EJB module 1 dependency inject @EJB được đặt trước Bean EJB vừa được gọi, điều này là sự khác biệt giữa EJB 3 và EJB 2 nếu không dùng Annotation @EJB thì ta phải tạo InitialContext để lookup EJB module.

```
@EJB  
private UserBean userBean;
```

Hình 15- Dependency Inject EJB module

Tiếp tục khai báo 1 biến user kiểu TblUser với hình thức truy cập là `private`, thực hiện các phương thức get/set cho thuộc tính này. Điều này để cho bạn có thể get và set giá trị mới cho một user. Tiếp theo chúng ta sẽ khởi tạo đối tượng TblUser trong Constructor của Controller.

```
user = new TblUser();
```

Việc tiếp theo chúng ta cần làm là tạo các hàm cần thiết trên controller

```
// Lay ra danh sach thanh vien goi tu DAO
```

```
public List<TblUser> getUsers(){  
    return userBean.retrieveAllUsers();  
}
```

```
// Lay thông tin chi tiết của một thành viên
```

```
public TblUser getDetails(){  
    return user;  
}
```

```
//hiển thị thông tin chi tiết của thành viên
```

```
public String showDetails(TblUser user){  
    this.user = user;  
    return "DETAILS";  
}
```

```
//Tạo combobox cho người dùng lựa chọn
public javax.faces.model.SelectItem[] getGroupList()
{
    SelectItem[] options = null;

    List<Groups> grouplist = userBean.getAllGroups();

    if (grouplist != null && grouplist.size() > 0)
    {
        int i = 0;

        options = new SelectItem[grouplist.size()];
        for (Groups iGroup : grouplist)
        {
            options[i++] = new SelectItem(iGroup,iGroup.getGroupname());
        }
    }

    return options;
}
```

Sau khi đã hoàn thành class Controller việc tiếp theo là ta phải tạo 1 class Convert để chuyển đổi kiểu dữ liệu Groups vì EL (Expression Language) không thể hiểu kiểu dữ liệu của Groups (1 entity) nên bây giờ ta sẽ tạo 1 Class có tên GroupsConverter và sẽ implements Interface Converter.

Trong class này ta sử dụng @FacesConverter để chỉ ra class GroupsConverter convert cho class Groups. Trong class này ta cũng override lại 2 phương thức từ Interface Converter là getAsObject và getAsString. getAsObject biến đổi một trường thành một đối tượng và getAsString. JSF cần hiển thị giá trị được chọn hiện tại nên nó dùng method getAsString để làm điều này và nó biến đổi một đối tượng thành một chuỗi ký tự.

Code của class này sẽ như sau


```
import com.jp.a.entity.Groups;

import javax.faces.application.FacesMessage;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.ConverterException;

import javax.faces.convert.FacesConverter;

/**
 *
 * @author Kency
 */
@FacesConverter(forClass=Groups.class)
public class GroupsConverter implements Converter{

    private Groups iGroups;

    @Override
    public Object getAsObject(FacesContext context, UIComponent component, String value) {

        if(iGroups == null){

            iGroups = new Groups(Integer.valueOf(value));

            return iGroups;

        }else{

            throw new ConverterException(new FacesMessage(String.format("Cannot convert %s to Groups", value)));

        }

    }

}
```

```
@Override
```

```
public String getAsString(FacesContext context, UIComponent component, Object value) {
```

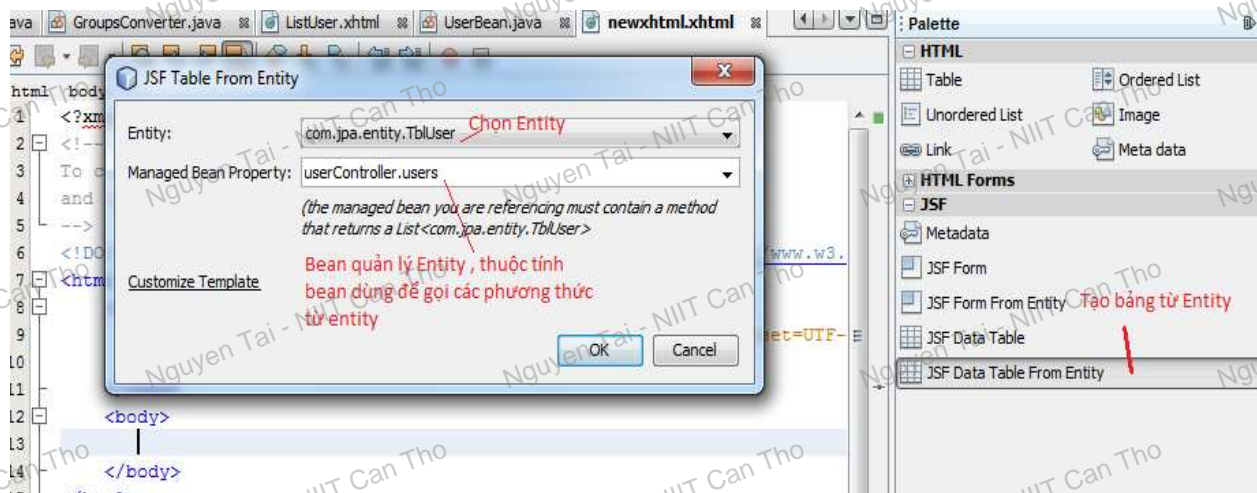
```
    return String.valueOf(((Groups) value).getGroupid());
```

```
}
```

```
}// end class
```

Sau khi đã tạo 1 class Convert cho Groups ta tiếp tục tạo Table để hiển thị danh sách thành viên.

Đầu tiên bạn tạo 1 file XHTML, sau đó bấm tổ hợp phím Ctrl + Shift + 8 ➔ Palette sẽ hiện ra, bấm dấu cộng ở JSF chọn Table From Entity kéo ra ngoài vùng HTML code



Hình 16 – Tạo Bảng từ Entity

Bây giờ để xem được chi tiết của một thành viên, bạn chỉnh sửa code ở cột userid như thành

```

<h:column>

    <f:facet name="header">

        <h:outputText value="Userid"/>

    </f:facet>

    <h:commandLink action="#{userController.showDetails(item)}" value="#{item.userid}"/>

</h:column>

```

Giải thích code này ta thấy ở commandLink thuộc tính Action sẽ gọi hàm ShowDetails và truyền tham số item, tham số item chính là 1 đối tượng user và để xác định được đối tượng này ta sử dụng item.userid để lấy id và các thông tin của đối tượng này. Sau đó ta dùng Browser gõ địa chỉ <http://localhost:8080/EJB3-war/ListUser.jsf> (ở đây tôi đặt tên của file XHTML vừa rồi là ListUser). Ta sẽ được như hình:

List

Userid	Username	Password	Dob	Groups	Delete
<u>1</u>	ThuanMinh	12345	01/01/2000 00:00:00	Member	delete

Hình 17 – Bảng thông tin thành viên tạo từ Entity

Vậy ta đã hoàn thành việc tạo danh sách thành viên từ Database, tiếp theo là sửa thông tin thành viên từ Database. Việc này cũng khá đơn giản, quay lại với class UserController ta viết thêm 1 hàm để thực hiện việc cập nhật thông tin thành viên như sau:

```

public void updateUser(){

    user = userBean.updateUser(user);

    FacesContext context = FacesContext.getCurrentInstance();

    context.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO,

        "Chúc mừng bạn", "Cập nhật thành công!"));

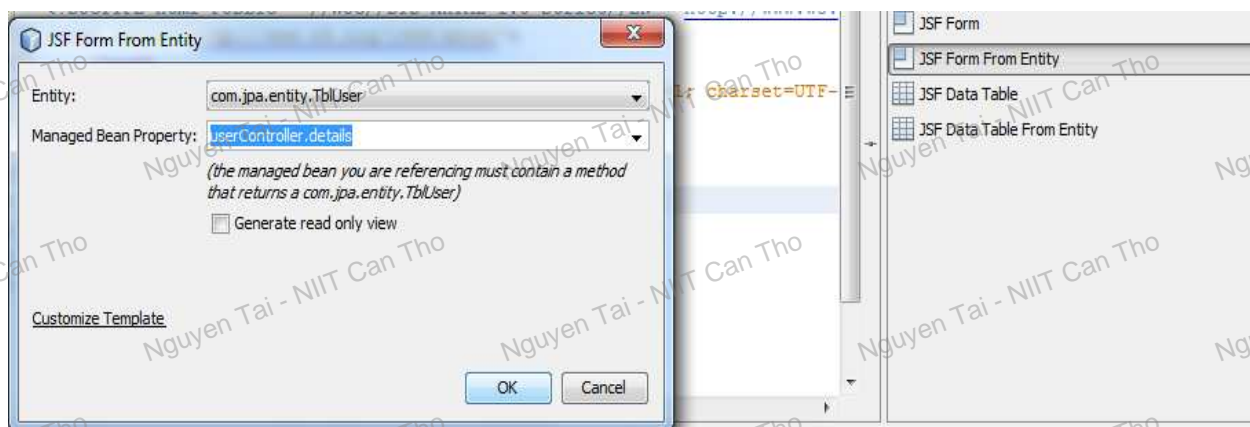
}

```

Tiếp tục viết thêm một hàm liên kết để trang update có thể quay về danh sách thành viên với code sau đây:

```
public String list()
{
    System.out.println("###LIST###");
    return "LIST";
}
```

Sau đó ta tạo 1 trang userDetails và sử dụng Form from Entity như hình:

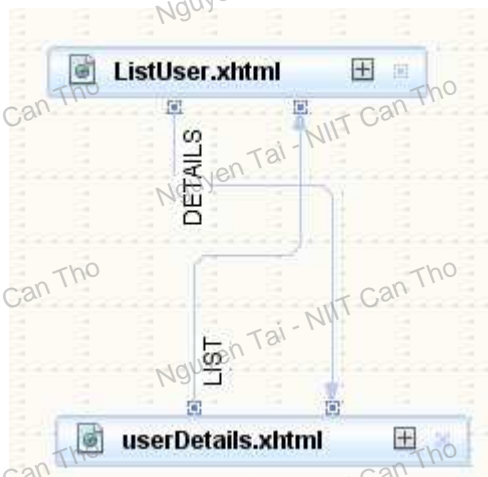


Hình 18 – Tạo trang truy vấn chi tiết thành viên

Ở đây tương tự như tạo Table từ entity nhưng khác là phương thức để quản lý cho trang này là phương thức getDetails (lưu ý với các từ khóa get/set JSF tự động bỏ đi và chuyển cụm từ sau từ khóa get/set thành lowercase).

Việc tiếp theo là tạo file Face-config.xml để cấu hình liên kết giữa 2 trang ListUser và userDetails.

Bạn cũng có thể cấu hình chúng thông qua XML cũng nằm trong Face-config.xml thông qua navigation rule của JSF .



Hình 19 – Mối quan hệ giữa 2 trang web

Như vậy ta đã có trang danh sách và chỉnh sửa danh sách thành viên và sau đây là code XHTML của trang userDetails.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
To change this template, choose Tools | Templates
and open the template in the editor.
-->

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      >

  <h:head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

    <title>TODO supply a title</title>

  </h:head>

  <h:body>

    <f:view>

      <h:form>
```

```
<h1><h:outputText value="Edit"/></h1>
```

```
<h:panelGrid columns="2">
```

```
  <h:outputLabel value="Username:" for="username" />
```

```
  <h:inputText id="username" value="#{userController.details.username}" title="Username"
  required="true" requiredMessage="The Username field is required."/>
```

```
  <h:outputLabel value="Password:" for="password" />
```

```
  <h:inputText id="password" value="#{userController.details.password}" title="Password"
  required="true" requiredMessage="The Password field is required."/>
```

```
  <h:outputLabel value="Dob:" for="dob" />
```

```
  <h:inputText id="dob" value="#{userController.details.dob}" title="Dob" required="true"
  requiredMessage="The Dob field is required.">
```

```
    <f:convertDateTime pattern="MM/dd/yyyy"/>
```

```
  </h:inputText>
```

```
  <h:outputLabel value="Groups:" for="groups" />
```

```
  <h:selectOneMenu id="groups" value="#{userController.details.groups}" title="Groups"
  required="true" requiredMessage="The Groups field is required.">
```

```
    <!-- TODO: update below reference to list of available items-->
```

```
    <f:selectItems value="#{userController.groupList}"/>
```

```
  </h:selectOneMenu>
```

```
</h:panelGrid>
```

```
<h:commandButton id="back" value="Back" action="#{userController.list}"/>
```

```
<h:commandButton id="update" value="Update" action="#{userController.updateUser}"/>
```

```
</h:form>
```

```
</f:view>
```

```
</h:body>
```

```
</html>
```

Form trên đã thực hiện 1 số thao tác bắt lỗi rồi, tiếp theo ta tạo 1 chức năng xóa thành viên. Như đã viết ở trên phần DAO ta đã có 1 phương thức delete thành viên, để phương thức này có thể đưa ra bean ta cần viết 1 phương thức để truyền tham số thành viên này vào để xóa thành viên này như sau:

```
public void deleteUser(TblUser users){  
    userBean.deleteUser(users);  
}
```

Ta cần chỉnh lại code trang danh sách sau khi đã thêm chức năng delete vào như sau:

```
<h:commandLink action="#{userController.deleteUser(item)}" value="delete"/>
```

Ở đây item sẽ là biến được set ở h:datatable với giá trị là thành viên cần xóa.

Sau khi hoàn tất ta sẽ được như hình

List

Userid	Username	Password	Dob	Groups	Delete
5	Tai Nguyen	12345	04/16/2010 00:00:00	Admin	delete

Hình – 20 Tạo chức năng delete

Ok! Như vậy là chúng ta đã hoàn thành xong việc liệt kê danh sách , cập nhật và xóa, còn thêm thì sao nhỉ?

Bây giờ chúng ta bắt đầu làm chức năng thêm

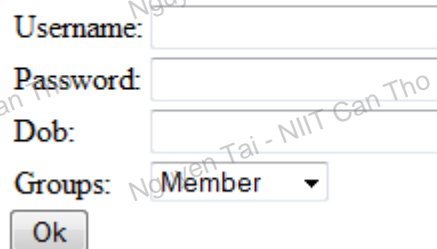
Ta cũng làm tương tự như các chức năng trên nhưng chức năng thêm sẽ nằm ở 1 trang riêng vì do không giống Scoped của nhau nên tôi phải tách riêng chức năng này một trang khác. Ta cũng sử dụng JSF Managed Bean để tạo 1 class là userAddController. Với scope là viewscope (bạn cũng có thể chọn requestscoped).

Ta cũng Inject 1 EJB Bean module và 1 biến thuộc loại TblUser vào class này và sau đó viết một hàm add để gọi phương thức add từ DAO để có thể đưa ra bean. Ngoài ra ta cũng sử dụng cặp phương thức get/set cho biến loại TblUser và khởi tạo biến này trong

Constructor của Class này. Class này cũng phải implements từ Serializable. Và tạo 1 phương thức select để tạo combobox cho form.

Sau khi hoàn thành ta tạo 1 form tương tự form userDetails. Từ Entity TblUser và được quản lý bởi Bean là userAddController . Sau khi tạo xong ta sẽ được một form mới dùng để thêm thành viên.

Create/Edit



Username:

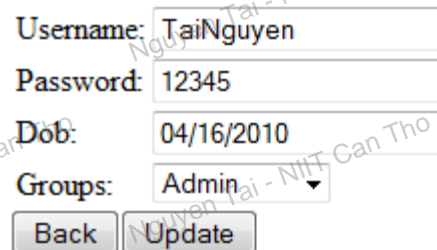
Password:

Dob:

Groups:

Hình 21 – Form để thêm thành viên

Edit



Username:

Password:

Dob:

Groups:

Hình 22 – Form dùng để chỉnh sửa thông tin thành viên

```
public void addUser(){  
    userTbl = userBean.persistUser(userTbl);  
}
```

Code thêm thành viên mới nằm ở Bean userAddController.

Lời kết

Như vậy là chúng ta đã hoàn thành việc thêm, xóa sửa, một entity sử dụng EJB3 và JSF2.0 với IDE Netbeans và GlassFish Server 3.0.

Bài hướng dẫn chỉ dừng ở mức độ hướng dẫn thực hành, không đi sâu quá nhiều vào tìm hiểu các thành phần của EJB 3 hay JSF 2.0. Tuy nhiên tôi đã cố gắng chú thích và diễn giải một số yếu tố dùng trong bài hướng dẫn. Mong rằng với bài hướng dẫn này sẽ giúp các bạn thực hiện được các bài tập hay website mong muốn.

Chúc các bạn thành công !