

LẬP TRÌNH JAVA

JSF

ThS. Dương Hữu Thành
Khoa CNTT, Đại học Mở Tp.HCM
thanh.dh@ou.edu.vn



1



Nội dung chính

- 1. Giới thiệu JSF**
- 2. Kiến trúc JSF**
- 3. Managed Bean và Facelet**
- 4. Các thành phần UI trong JSF**
- 5. Các thành phần kiểm tra dữ liệu**
- 6. Các thẻ facelet**
- 7. Thẻ thực thi Ajax**
- 8. Chuyển trang trong JSF**
- 9. Sử dụng Resource Bundle**
- 10. Giao tiếp trong JSF**



Giới thiệu JSF

- JavaServer Faces (JSF) là một **framework** chuẩn viết bằng Java để phát triển ứng dụng web **phía server** (server-side) cho Java EE được **phát hành** bởi Oracle.
- Ứng dụng JSF sử dụng kiến trúc **MVC** (Model-View-Controller) **tách biệt** dữ liệu (Model) với kiến trúc người dùng (View).
- Controller chịu trách nhiệm kết hợp tương tác giữa View và Model. Nó xử lý các yêu cầu (action) từ người dùng, tương tác với Model và cập nhật dữ liệu hiển thị lên View.

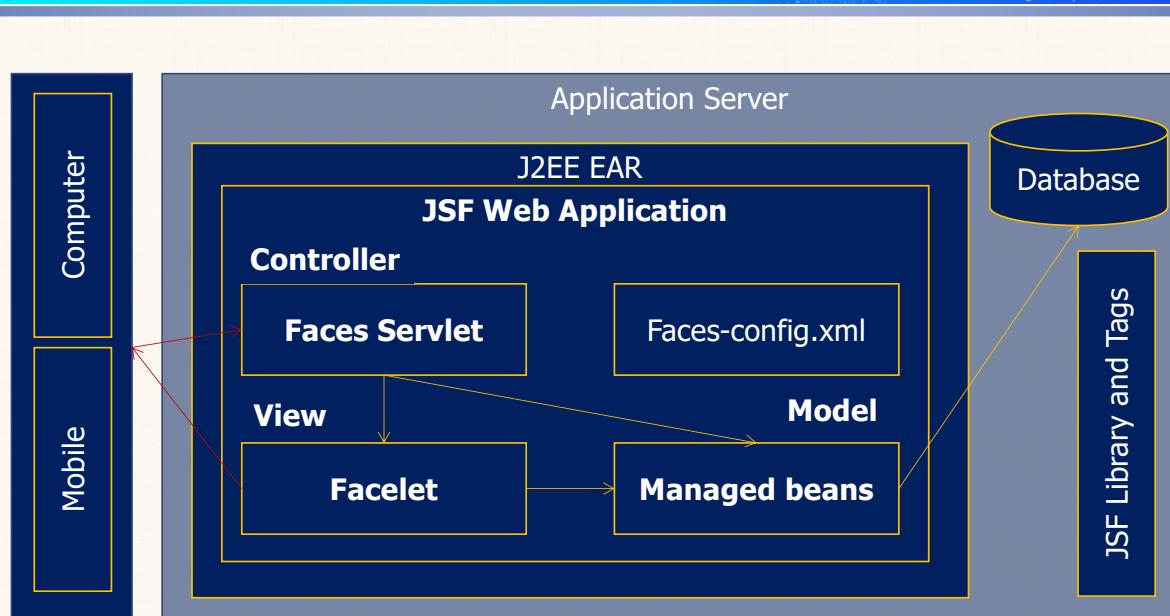
Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

3

3



Kiến trúc JSF



Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

4

4



Kiến trúc JSF

- **Faces Servlet** đóng vai trò là Controller nhận các yêu cầu gửi đến tương tác với Model (Managed Java Bean) và hiển thị dữ liệu lên View (facelet)
- Từ JSF 2.0 sử dụng công nghệ **facelet** để xây dựng view cho ứng dụng web phát triển bằng JSF.
- Các lớp **Managed Bean** đóng vai trò là Model chứa các xử lý logic tương tác với cơ sở dữ liệu.
- Trong JSF 1.2, Managed Bean được đăng ký trong tập tin cấu hình faces-config.xml. Từ JSF 2.0, ta có thể dễ dàng làm điều này bằng cách sử dụng **annotation**.



Tạo project đầu tiên với NetBeans

- Trong ví dụ này sử dụng
 - JDK 13
 - NetBeans IDE 11.2
 - Apache Tomcat 9.0.30



Tạo project đầu tiên với NetBeans

- Apache Tomcat là một Java Web Server phổ biến, truy cập vào trang web tomcat và click vào Tomcat 9 để đến trang tải về.
- <http://tomcat.apache.org/>

The screenshot shows the Apache Tomcat download page at tomcat.apache.org/download-90.cgi. The page displays binary distributions for Tomcat 9.0.30. Under the 'Binary Distributions' section, it lists several options for Core, Full documentation, Deployer, and Embedded distributions, each with links for zip and tar.gz formats.

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

7



Tạo project đầu tiên với NetBeans

- Tạo một maven project mới từ NetBeans

The image contains two side-by-side screenshots of the NetBeans IDE. The left screenshot shows the 'New Project' wizard's 'Choose Project' step, where 'Java with Maven' is selected from the 'Categories' list. The right screenshot shows the 'New Web Application' wizard's 'Name and Location' step, with fields filled for a project named 'jsfdemo' located at '/duonghuuthanh/NetBeansProjects/saleweb/test'. Both screenshots show the standard NetBeans UI with tabs for Help, Back, Next >, Finish, and Cancel.

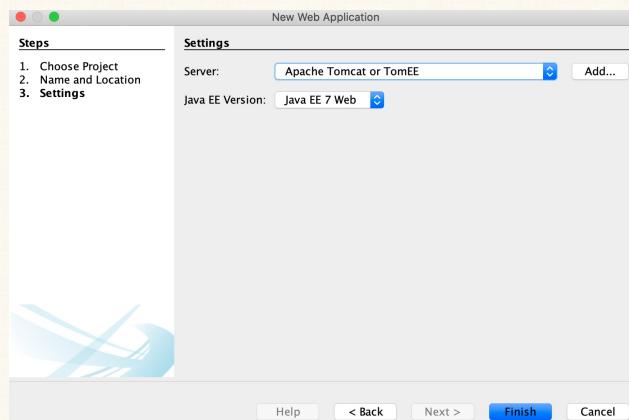
Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

8



Tạo project đầu tiên với NetBeans

▪ Cấu hình Tomcat Server

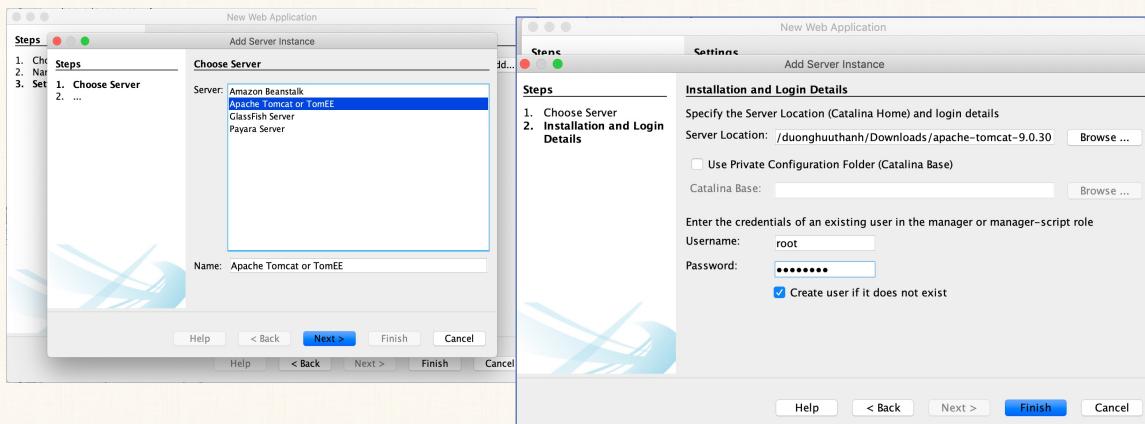


- Nếu chưa có cấu hình Server thì click vào Add để thêm thông tin cấu hình cho Server.



Tạo project đầu tiên với NetBeans

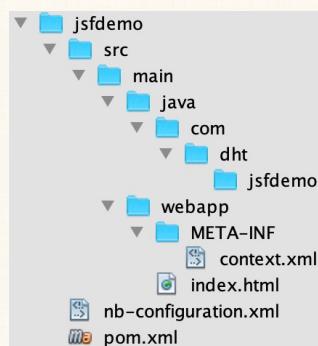
- Sau khi tải Tomcat thì giải nén, chỉ định vị trí thư mục này cho trường Server Location, đồng thời chỉ định thông tin username, password khi chạy server và click nút Finish.





Tạo project đầu tiên với NetBeans

- Cấu trúc project được tạo ra như sau



Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

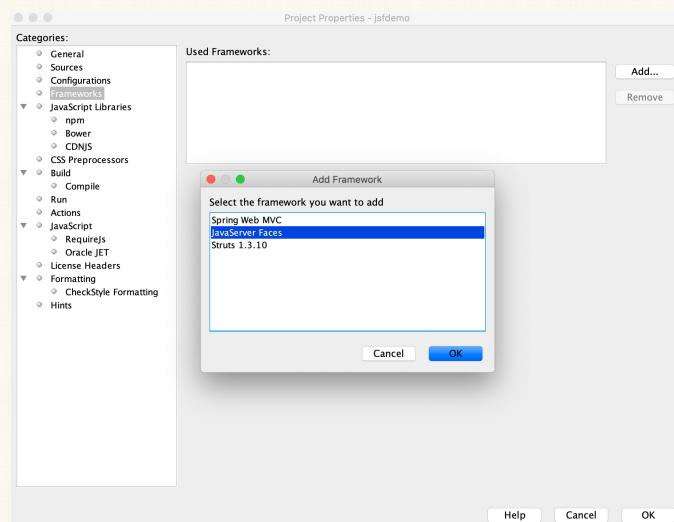
11

11



Tạo project đầu tiên với NetBeans

- Click phải project > Properties > Frameworks
- Click nút Add..., chọn JavaServer Faces



Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

12

12



Managed Bean

- Managed Bean đơn giản là lớp **public**, có **phương thức khởi tạo không tham số**.
- Các thuộc tính quy ước định nghĩa thông qua các phương thức getter và setter của nó, không cần định nghĩa trường dữ liệu tường minh trong lớp.
- Tạo gói com.dht.bean, tạo HelloBean.java.



Managed Bean

- **@Named (value="helloBean")** là annotation chỉ định JSF framework tạo và quản lý các đối tượng HelloBean được sử dụng trong ứng dụng (mặc định là tên lớp với ký tự đầu tiên viết thường helloBean).
- Mỗi lớp Managed Bean cần chỉ định phạm vi hoạt động của các đối tượng của nó, mặc định là **@RequestScope**.



Managed Bean

- **@RequestScoped**: bean tồn tại trong khoảng thời gian HTTP request và response.
- **@ViewScoped**: bean tồn tại khi người dùng chỉ tương tác trên cùng view của trình duyệt.
- **@SessionScoped**: bean tồn tại trong HTTP session.
- **@ApplicationScoped**: bean tồn tại trong suốt thời gian ứng dụng web hoạt động.



Managed Bean

- Sửa nội dung HelloBean.java như sau:

```
package com.dht.bean;

import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.faces.bean.ManagedBean;

@ManagedBean
@Named(value = "helloBean")
@RequestScoped
public class HelloBean {
    public String getMessage() {
        return "Welcome to our Websites!!!!";
    }
}
```



Facelet

- Trong JSF, facelet là view để hiển thị dữ liệu đến người dùng, dữ liệu là các đối tượng Java, các đối tượng có thể được truy cập từ facelet là các đối tượng của Managed Bean.

```
# {expression}
```

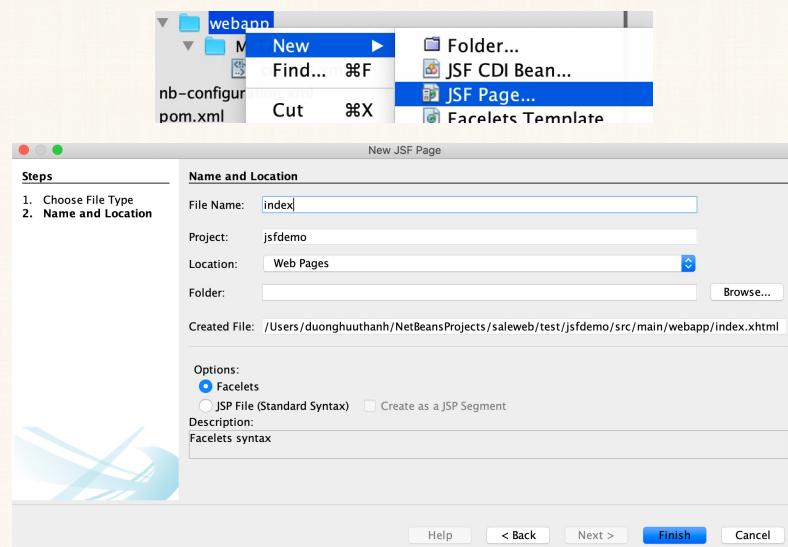
- Ví dụ

```
# {helloBean.message}
```



Facelet

- Xoá index.html trong thư mục webapp, tạo JSF Pages đặt tên index.xhtml.





Facelet

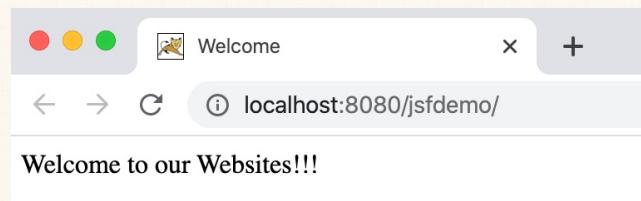
- Sửa tập tin index.xhtml như sau:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Welcome</title>
  </h:head>
  <h:body>
    #{helloBean.message}
  </h:body>
</html>
```



Tạo project đầu tiên với NetBeans

- Click phải vào project > Run
- Truy cập <http://localhost:8080/jsfdemo/>





Các thẻ UI trong JSF

- JSF cung cấp một thư viện các thẻ HTML chuẩn giúp cho thiết kế giao diện dễ dàng và hiệu quả hơn.
- Để sử dụng cần thêm namespace sau:

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



Các thẻ UI trong JSF

- Các thành phần UI cơ bản thông dụng

h:form	Tạo thẻ form.
h:panelGroup	Tổ chức các node từng hàng theo chiều ngang hoặc từng cột theo chiều dọc (Flow).
h:panelGrid	Tổ chức các node theo dạng lưới (Grid).
h:inputText	Ô textbox nhập liệu.
h:outputText	Ô textbox hiển thị.
h:inputTextArea	Tạo textarea nhập nhiều dòng.
h:inputSelect	Ô textbox nhập mật khẩu.
h:outputLabel	Hiển thị nhãn.
h:outputLink	Hiển thị siêu liên kết.
h:message	Hiển thị thông báo nào đó.
h:graphicImage	Hiển thị hình ảnh.
h:commandButton	Hiển thị nút bấm đại diện input button có type là submit.



Các thẻ UI trong JSF

h:selectOneMenu	Tạo combobox cho phép chọn 1 item.
h:selectOneRadio	Hiển thị nhóm radiobutton cho phép chọn 1 item tại một thời điểm.
h:selectManyCheckbox	Tạo các điều khiển checkbox cho phép chọn nhiều lựa chọn.
h:selectOneListbox	Hiển thị dạng danh sách cho phép lựa chọn một.
h:selectManyListbox	Hiển thị dạng danh sách cho phép lựa chọn nhiều.
h:selectOneMenu	Tạo combobox cho phép chọn 1 item.
h:selectOneRadio	Hiển thị nhóm radiobutton cho phép chọn 1 item tại một thời điểm.
h:selectManyCheckbox	Tạo các điều khiển checkbox cho phép chọn nhiều lựa chọn.
h:selectOneListbox	Hiển thị dạng danh sách cho phép lựa chọn một.
f:selectItem	Đại diện các lựa chọn trong h:selectOneMenu, h:selectOneRadio hoặc h:selectManyCheckbox.
h:dataTable	Hiển thị bảng dữ liệu.
h:column	Đại diện cho cột trong bảng dữ liệu.



Các thẻ UI trong JSF

- Tạo trang Web khảo sát như sau:

 **PHIẾU KHẢO SÁT**

Họ tên

Năm sinh

Giới tính Nam Nữ

Ngôn ngữ lập trình yêu thích?

C/C++ Java Python Ruby PHP

Khác

Tại sao bạn yêu thích các ngôn ngữ đó?

Gửi



Các thẻ UI trong JSF

- Tất cả các thẻ sau đều đặt bên trong thẻ `<h:form></h:form>`
- Trong JSF, tất cả các tài nguyên (hình ảnh, css, audio, ...) đều nằm trong thư mục resources (thư mục con của Web Pages).

resources/image/check.png

```
<h:panelGrid columns="2">
    <h:graphicImage name="check.png" width="50"
                     height="50" library="image" />
    <h3>PHIẾU KHẢO SÁT</h3>
</h:panelGrid>
```



Các thẻ UI trong JSF

- Sử dụng InputText và SelectOneRadio

```
<h:panelGrid columns="2">
    <h:outputLabel value="Họ tên" />
    <h:inputText id="txtFullName" />
    <h:outputLabel value="Năm sinh" />
    <h:inputText id="txtDateOfBirth" />
    <h:outputLabel value="Giới tính" />
    <h:selectOneRadio id="rdoGender">
        <f:selectItem
            itemValue="Nam" itemLabel="Nam" />
        <f:selectItem
            itemValue="Nữ" itemLabel="Nữ" />
    </h:selectOneRadio>
</h:panelGrid>
```



Các thẻ UI trong JSF

▪ Sử dụng selectManyCheckbox

```
<h:panelGrid>
    <h:outputLabel value="Ngôn ngữ lập trình yêu thích?" />
    <b><h:selectManyCheckbox id="chkFavoriteLanguages"></b>
        <f:selectItem itemValue="C/C++" />
        <f:selectItem itemValue="Java" />
        <f:selectItem itemValue="Python" />
        <f:selectItem itemValue="Ruby" />
        <f:selectItem itemValue="PHP" />
    </b></h:selectManyCheckbox>
</h:panelGrid>
```



Các thẻ UI trong JSF

▪ Sử dụng inputTextarea

```
<h:panelGrid>
    <h:outputLabel value="Tại sao bạn yêu thích
các ngôn ngữ đó?" />
    <b><h:inputTextarea id="txtReason"
rows="4" cols="80" /></b>
</h:panelGrid>
```



Các thẻ UI trong JSF

▪ Sử dụng CommandButton

```
<h:panelGrid>
    <h:commandButton value="Gui"
        style="background-color:blue;color:white" />
</h:panelGrid>

<h:outputText escape="false"
    value="#{surveyBean.response}" />
```



Các thẻ UI trong JSF

▪ Tạo lớp Managed Bean đặt tên SurveyBean

```
@Named(value = "surveyBean")
@RequestScoped
public class SurveyBean {
    private String fullName;
    private String dateOfBirth;
    private String gender;
    private String[] favoriteLanguages;
    private String otherLanguages;
    private String reason;
    public String getResponse() {
        String result = "";
        return result;
    }
    // Các phương thức getter và setter
}
```



Các thẻ UI trong JSF

- Kết buộc các thẻ HTML vào các thuộc tính của Managed Bean.

```
<h:inputText id="txtFullName"
              value="#{surveyBean.fullName}" />
<h:inputText id="txtDateOfBirth"
              value="#{surveyBean.dateOfBirth}" />
<h:selectOneRadio id="rdoGender"
                  value="#{surveyBean.gender}">
</h:selectOneRadio>
<h:selectManyCheckbox id="chkFavoriteLanguages"
                      value="#{surveyBean.favoriteLanguages}">
</h:selectManyCheckbox>
<h:inputText id="txtOtherLanguages"
              value="#{surveyBean.otherLanguages}" />
<h:inputTextarea id="txtReason" rows="4"
                 value="#{surveyBean.reason}" />
```



Các thẻ UI trong JSF

- Phương thức Response

```
public String getResponse() {
    StringBuilder b = new StringBuilder();

    if (FacesContext.getCurrentInstance().isPostback()) {
        b.append("Họ tên: " + this.fullName + "<br>");
        b.append("Ngày sinh: " + this.dateOfBirth + "<br>");
        b.append("Giới tính: " + this.gender + "<br>");
        b.append("Ngôn ngữ yêu thích: <br> ");
        for (String nn: this.getFavoriteLanguages())
            b.append(nn + "\t");
        b.append("<br>Ngôn ngữ khác: "
                + this.otherLanguages + "<br>");
        b.append("Lý do yêu thích: " + this.reason + "<br>");
    }
    return b.toString();
}
```



Các thẻ UI trong JSF

- Chèn CSS, JS trong facelet

```
<h:outputStylesheet name="style.css" library="css" />
<h:outputScript name="script.js" library="js" />
```

- Với các tập tin CSS chèn theo cách thức này có thể sử dụng cú pháp sau lấy các tài nguyên trong thư mục resource

```
footer {
    background: url("#{resource['images/footer.png']}")
}
```



Các thẻ UI trong JSF

- Ghi chú: trong trường hợp, ta cần xử lý khởi động nào đó trước khi phương thức action trong managed bean được thực thi, ta có thể tạo một phương thức public void với annotation @PostConstruct

```
@PostConstruct
public void init() {
}
```



Các thẻ kiểm tra dữ liệu

- JSF cũng cung cấp sẵn các thẻ để kiểm tra dữ liệu đầu vào nhanh chóng và dễ dàng.
- Để sử dụng các thẻ này cần thêm các namespace sau:

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



Các thẻ kiểm tra dữ liệu

- Các thẻ kiểm tra thông dụng.

f:validateLength	Kiểm tra chiều dài dữ liệu nhập.
f:validateDoubleRange	Kiểm tra dữ liệu đầu vào có nằm trong phạm vi cho phép.
f:validateLongRange	Kiểm tra dữ liệu đầu vào không được rỗng.
f:validateRequired	Kiểm tra dữ liệu đầu vào khớp với biểu thức chính quy.
f:validateRegex	Thực thi phương thức kiểm tra dữ liệu đầu vào tự tạo.
f:validateBean	

- Tiếp theo ví dụ phiếu khảo sát với các ràng buộc như sau trước khi gửi lên server.



Các thẻ kiểm tra dữ liệu

- Họ tên không được phép rỗng

```
<h:inputText id="txtFullName"
    value="#{surveyBean.fullName}"
    required="true"
    requiredMessage="Vui lòng nhập họ tên." />
<h:message for="txtFullName" style="color:red;" />
```



Các thẻ kiểm tra dữ liệu

- Năm sinh không được rỗng, chỉ chứa ký số.

```
<h:inputText id="txtDateOfBirth"
    value="#{surveyBean.dateOfBirth}"
    required="true"
    requiredMessage="Vui lòng nhập năm sinh."
    validatorMessage="Năm không hợp lệ">
    <f:validateRegex pattern="[0-9]+">
</h:inputText>
<h:message for="txtDateOfBirth" style="color:red;" />
```



Các thẻ kiểm tra dữ liệu

- Chiều dài tối đa ô lý do yêu thích dài không quá 255 ký tự.

```
<h:inputTextarea id="txtReason" rows="4"
    value="#{surveyBean.reason}"
    validatorMessage="Số ký tự tối đa là 255.">
    <f:validateLength maximum="255" />
</h:inputTextarea>
<h:message for="txtReason" style="color:red;" />
```



Các thẻ kiểm tra dữ liệu

- Phải chọn ít nhất một ngôn ngữ yêu thích hoặc ô nhập ngôn ngữ yêu thích khác không được rỗng.

```
<h:inputText id="txtOtherLanguages"
    value="#{surveyBean.otherLanguages}">
    <f:validator validatorId="LanguageValidator" />
</h:inputText>
<h:message for="txtOtherLanguages"
    style="color:red;" />
```

- LanguageValidator là một validator tạo ra bằng cách hiện thực giao diện Validator.
 - javax.faces.validator.Validator
 - javax.faces.application.FacesMessage



Các thẻ kiểm tra dữ liệu

```
@FacesValidator("LanguageValidator")
public class LanguageValidator
    implements Validator {
    @Override
    public void validate(FacesContext context,
        UIComponent component,
        Object value) throws ValidatorException {
        UISelectMany chkFavoriteLanguages = (UISelectMany)
            component.findComponent("chkFavoriteLanguages");

        if(value.toString().isEmpty() &&
            chkFavoriteLanguages.getSelectedValues().length == 0) {
            String err = "Vui lòng chọn/nhập ngôn ngữ yêu thích";
            FacesMessage msg = new FacesMessage(err);
            msg.setSeverity(FacesMessage.SEVERITY_ERROR);

            throw new ValidatorException(msg);
        }
    }
}
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

41

41



Các thẻ kiểm tra dữ liệu

- Mặc định JSF không kiểm tra các ô dữ liệu rỗng, để làm điều đó ta bổ sung cấu hình trong web.xml

```
<context-param>
    <param-name>javax.faces.VALIDATE_EMPTY_FIELDS</param-name>
    <param-value>true</param-value>
</context-param>
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

42

42



Các thẻ kiểm tra dữ liệu

- Chạy thử chương trình

PHIẾU KHẢO SÁT

Họ tên Vui lòng nhập họ tên.

Năm sinh Vui lòng nhập năm sinh.

Giới tính Nam Nữ Vui lòng chọn giới tính

Ngôn ngữ lập trình yêu thích?

C/C++ Java Python Ruby PHP

Khác Vui lòng chọn/nhập ngôn ngữ yêu thích

Tại sao bạn yêu thích các ngôn ngữ đó?



Các thẻ facelet

- JSF cung cấp các thẻ facelet để thiết kế template chung của ứng dụng web.
- Để sử dụng các thẻ này cần thêm các namespace sau:

```
<html xmlns = "http://www.w3.org/1999/xhtml"  
      xmlns:ui = "http://java.sun.com/jsf/faces">
```



Các thẻ facelet

▪ Các thẻ facelet thông dụng

<ui:insert>	Chỉ định vùng nội dung trong template, nó có thể được ghi đè bởi nội dung của <ui:define>.
<ui:define>	Chỉ định một nội dung nào đó được chèn trong trang của template.
<ui:include>	Thêm nội dung của một trang xhtml này vào trang xhtml khác.
<ui:composition>	Nạp template hoặc định nghĩa một nhóm thành phần sẽ được chèn vào trang xhtml.
<ui:param>	Dùng truyền tham số vào template.



Các thẻ facelet

▪ Tạo các template có phần body như sau

▪ header.xhtml

```
<ui:composition>
    <h1>PHIẾU KHẢO SÁT</h1>
</ui:composition>
```

▪ footer.xhtml

```
<ui:composition>
    <h5>Dương Hữu Thành © 2019</h5>
</ui:composition>
```

▪ content.xhtml

```
<ui:composition>
    <div>Khảo sát lập trình viên</div>
</ui:composition>
```



Các thẻ facelet

- Tạo template đặt tên base.xhtml như sau:

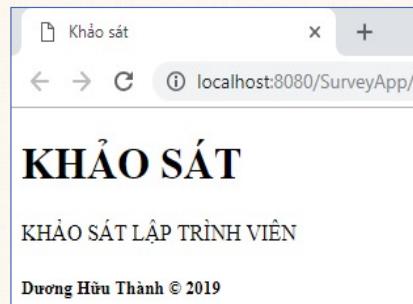
```
<h:body>
    <ui:insert name="header">
        <ui:include src="header.xhtml" />
    </ui:insert>
    <ui:insert name="content">
        <ui:include src="content.xhtml" />
    </ui:insert>
    <ui:insert name="footer">
        <ui:include src="footer.xhtml" />
    </ui:insert>
</h:body>
```



Các thẻ facelet

- Sử dụng template mặc định cho trang chủ, đặt tên index.xhtml như sau:

```
<h:body>
    <ui:composition template="base.xhtml" />
</h:body>
```

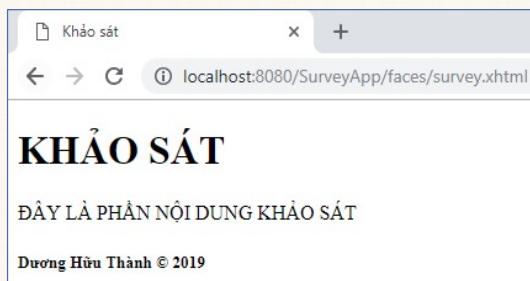




Các thẻ facelet

- Ghi đè phần content của base.xhtml, tạo trang survey.xhtml như sau:

```
<ui:composition template="base.xhtml">
    <ui:define name="content">
        ĐÂY LÀ PHẦN NỘI DUNG KHẢO SÁT
    </ui:define>
</ui:composition>
```



Các thẻ facelet

- Sử dụng <ui:param>, sửa lại header.xhtml:

```
<ui:composition>
    <h1>#{subjectPage}</h1>
</ui:composition>
```

- Sửa trang index.xhtml như sau:

```
<ui:composition template="base.xhtml"subjectPage" value="TRANG CHỦ" />
</ui:composition>
```





Thẻ thực thi Ajax

- Ajax (Asynchronous Javascript and XML) là kỹ thuật gửi và nhận dữ liệu từ Server một cách bất đồng bộ.
- Ajax giúp tăng hiệu năng ứng dụng web và chỉ cập nhật vùng cần thay đổi mà không cần phải nạp lại toàn bộ trang web.
- JSF cung cấp thẻ để xử lý các lời gọi ajax.

```
<f:ajax execute="" render="" />
```



Các thuộc tính <f:ajax>

- **execute**: danh sách id của các thành phần HTML sẽ được kèm vào request.
- **render**: danh sách id của các thành phần HTML được cập nhật sau ajax request.
- **event**: tên sự kiện sẽ thực thi ajax request, chẳng hạn click, change, blur.



Các thuộc tính <f:ajax>

- **listener**: phương thức ở bean sẽ được gọi khi thực thi ajax request.
- **onevent**: tên hàm javascript (callback) được gọi thực thi xử lý giao diện người dùng.
- **onerror**: tên hàm javascript được gọi thực thi nếu có lỗi trong quá trình thực hiện ajax request.



Thẻ thực thi Ajax

- Tạo Managed Bean đặt tên AjaxBean.xhtml

```
@Named(value = "ajBean")
@RequestScoped
public class AjaxBean {
    private String name;

    public String getMessage() {
        return "Welcome " + this.name;
    }

    // Phương thức getter và setter của name
}
```



Thẻ thực thi Ajax

- Tạo facelet đặt tên AjaxBean.xhtml

```
<h1>Welcome Page</h1>
<h:form>
    <h:inputText id="inputName" value="#{ajBean.name}" />
    <h:commandButton value="Show">
        <f:ajax execute="inputName" render="outputName" />
    </h:commandButton>
    <h2>
        <h:outputText id="outputName"
            value="#{ajBean.name!=null?ajBean.message:' '}"/>
    </h2>
</h:form>
```



Chuyển trang trong JSF

- Trong JSF 2.0, ta chỉ cần chỉ định tên các view thì nó sẽ tự động tìm kiếm chính xác view.
- Ví dụ

```
<h1>Trang chủ</h1>
<h:form>
    <h:commandButton action="SurveyBean"
        value="Đến trang khảo sát" />
</h:form>
```

- JSF sẽ tìm tập tin view SurveyBean.xhtml trong thư mục hiện hành.



Chuyển trang trong JSF

- Ví dụ chuyển trang trong Managed Bean. Giả sử trong trang khảo sát thêm một button “Trở về trang chủ”.

```
<h:form>
    <h:commandButton action="#{surveyBean.backHome}"
                      value="Trở về trang chủ" />
</h:form>
```

- Action **backHome ()** như sau

```
public String backHome() {
    return "index";
}
```



Chuyển trang trong JSF

- Để ý rằng khi chuyển trang thì URL trên thanh địa chỉ của trình duyệt không đổi.
- Để thay đổi trên thanh địa chỉ, ta thêm tham số faces-redirect=true vào sau tên view:

```
public String backHome() {
    return "index?faces-redirect=true";
}
```



Sử dụng Resource Bundle

- Thông thường, ta ưu tiên sử dụng các message được lưu trong các tập tin property hơn là sử dụng cố định message trực tiếp trong các trang.
- Resource bundle là các cặp chuỗi key/value lưu trong các tập tin có đuôi .properties.
- Tạo thư mục resources trong **src/main**, tạo gói com.dht.resourcebundle trong thư mục resources.

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

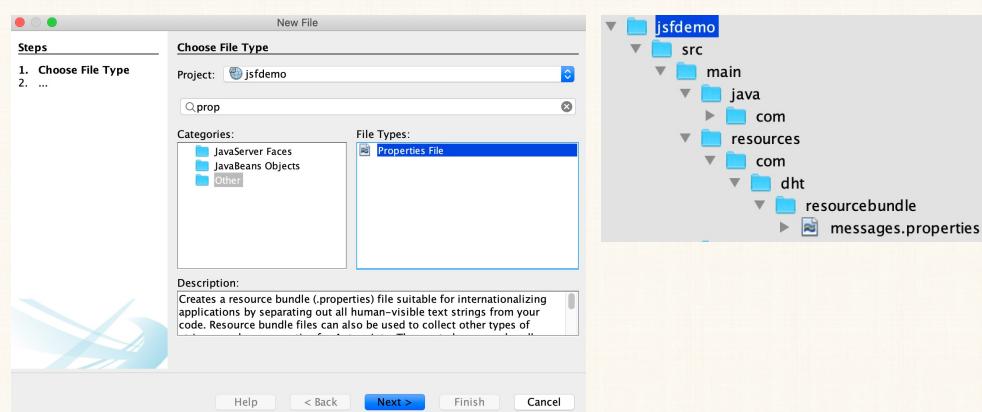
59

59



Sử dụng Resource Bundle

- Tạo tập tin messages.properties trong gói này.



- Nội dung tập tin messages.properties như sau:

```
message = Welcome to our Websites!!!
message.pagename = DHT'S PAGE
message.welcome = Nice to meet you, {0}!
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

60

60



Sử dụng Resource Bundle

- Trong facelet sử dụng thẻ **<f:loadBundle />** dùng để nạp các tập tin property cú pháp:

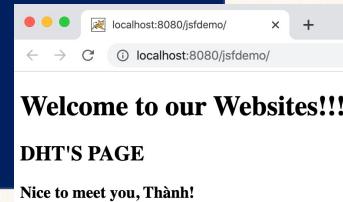
```
<f:loadBundle var="" basename="" />
```

- basename**: chỉ định vị trí tập tin properties.
- var**: chỉ định tên biến dùng truy cập các giá trị trong tập tin property.



Sử dụng Resource Bundle

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<h:head>
    <f:loadBundle var="msg"
                  basename="com.dht.resourcebundle.messages" />
</h:head>
<h:body>
    <h1>#{msg['message']}</h1>
    <h2>#{msg['message.pagename']}</h2>
    <h3>
        <h:outputFormat
            value="#{msg['message.welcome']} ">
            <f:param value="Thành" />
        </h:outputFormat>
    </h3>
</h:body>
</html>
```





Sử dụng Resource Bundle

- Ngoài ra để cấu hình resource bundle toàn cục trong hệ thống dùng cấu hình faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
    version="2.0">
<application>
    <resource-bundle>
        <base-name>com.dht.resourcebundle.messages</base-name>
        <var>msg</var>
    </resource-bundle>
</application>
</faces-config>
```



Giao tiếp Managed Bean và Facelet

- JSF cung cấp nhiều giải pháp giúp giao tiếp linh hoạt giữa các thành phần trong JSF và giữa JSF và các trang XHTML, mã nguồn Javascript và các thành phần bên thứ ba.



Sử dụng Context Params

- Context param được định nghĩa thông qua thẻ `<context-param>` trong tập tin web.xml.
 - `<param-name>` chỉ định tên tham số.
 - `<param-value>` chỉ định giá trị tham số.

```
<context-param>
    <param-name>author</param-name>
    <param-value>Dương Hữu Thành</param-value>
</context-param>
```



Sử dụng Context Params

- Để sử dụng các tham số này trong các trang facelet như sau:

```
<h:outputText value="#{initParam['author']}' />
<h:outputText value="#{facesContext.getExternalContext
    .initParameterMap['author']}' />
```

- Trong managed bean có thể sử dụng các tham số này như sau:

```
facesContext.getExternalContext()
    .getInitParameter('author');
```



Làm việc với Request Param

- Để truyền tham số từ facelet tới managed bean hoặc giữa các facelet sử dụng thẻ `<f:param>`.
- Thẻ `<f:param>` sử dụng bên trong các thẻ `<h:commandButton>` và `<h:commandLink>` để gửi các tham số request đến managed bean.

```
<h:form>
    <h:commandButton value="Search"
                      action="#{testBean.paramAction()}">
        <f:param id="kw" name="keyword"
                  value="test" />
    </h:commandButton>
</h:form>
```



Làm việc với Request Param

- Khi click vào button, các tham số request sẽ được gửi và phương thức `paramAction()` sẽ được gọi, ta có thể dễ dàng truy cập vào các tham số request thông qua thể hiện của `FacesContext` như sau:

```
public String paramAction() {
    FacesContext fc = FacesContext.getCurrentInstance();
    Map<String, String> p = fc.getExternalContext()
                           .getRequestParameterMap();
    String kw = p.get("keyword");

    return "some_page";
}
```



Làm việc với Request Param

- Trong facelet “some_page” có thể sử dụng biến param (biến được định nghĩa trước tham chiếu đến map các tham số của request).

Keyword: #{param.keyword}



Làm việc với Request Param

- Ta cũng có thể sử dụng @ManagedProperty để thiết lập các thuộc tính của managed bean và liên kết giá trị vào các tham số request.

```
@ManagedProperty(value="#{param.keyword}")  
private String keyword;
```



Làm việc với Request Param

- Ngoài ra, thẻ `<f:param>` sử dụng bên trong thẻ `<h:outputFormat>` để chỉ định giá trị cho các định dạng định nghĩa trong thuộc tính `value` của thẻ `<h:outputFormat>`

```
<h:outputFormat  
    value="FirstName: {0}; LastName: {1}">  
    <f:param value="Thanh" />  
    <f:param value="Duong" />  
</h:outputFormat>
```



Làm việc với View Param

- View Param được bổ sung từ JSF 2.0 thông qua thẻ `<f:viewParam>` sử dụng trong `<f:metadata>`

- Ví dụ**

```
<f:metadata>  
    <f:viewParam name="fn"  
        value="#{userBean.firstName}" />  
    <f:viewParam name="ln"  
        value="#{userBean.lastName}" />  
</f:metadata>  
<h:body>  
First Name: <h:outputText  
            value="#{userBean.firstName}" />  
Last Name: <h:outputText  
            value="#{userBean.lastName}" />  
</h:body>
```



Làm việc với View Param

- Khối <f:metadata>
 - JSF lấy giá trị các tham số request thông qua tên từ URL và áp dụng converter/validator chỉ định.
 - Sau đó, JSF kết buộc các giá trị các tham số fn và ln với các thuộc tính firstName và lastName của managed bean bằng cách gọi các phương thức setter.
- Khối <h:body>
 - Giá trị các thuộc tính được hiển thị thông qua các phương thức getter trong managed bean.



Làm việc với View Param

- View Param được gắn vào URL bằng cách sử dụng thuộc tính includeViewParams="true" trong thẻ <h:link>.

```
<h:link value="Go"
        outcome="results?faces-redirect=True"
        includeViewParams="true" />
```

- hoặc tham số includeViewParams=true trong các URL request

```
<h:commandButton value="Go" action="results?faces-
redirect=true&includeViewParams=true" />
```



Làm việc với View Param

- Trang results.xhtml như sau:

```
<f:metadata>
    <f:viewParam name="fn"
                  value="#{userBean.firstName}" />
    <f:viewParam name="ln"
                  value="#{userBean.lastName}" />
</f:metadata>
<h:body>
First Name: <h:outputText
                value="#{userBean.firstName}" />
Last Name: <h:outputText
                value="#{userBean.lastName}" />
</h:body>
```



Sử dụng flask

- JSF Flask là một giải pháp thuận tiện cho phép truyền dữ liệu giữa các view mà không cần lưu trữ dữ liệu trong session.
- Nó hiệu quả trong xử lý POST-redirect-GET.
- Ví dụ tạo 2 trang JSF
 - Trang 1 cho nhập First Name, Last Name và một button chuyển đến trang 2.
 - Trang 2 hiển thị lời chào “Welcome first last!!!”



Sử dụng flask

```
<h:form>
    <h:panelGrid columns="3">
        <h:outputLabel value="First Name: " />
        <h:inputText id="txtFirstName"
            value="#{userBean.firstName}"
            required="true"
            requiredMessage="Please enter first name." />
        <h:message for="txtFirstName" />
        <h:outputLabel value="Last Name: " />
        <h:inputText id="txtLastName"
            value="#{userBean.lastName}"
            required="true"
            requiredMessage="Please enter last name." />
        <h:message for="txtLastName" />
        <h:commandButton value="Send"
            action="#{userBean.sendMessage()}" />
    </h:panelGrid>
</h:form>
```

user.xhtml

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

77

77



Sử dụng flask

- Phương thức sendMessage() trong managed bean

```
public String sendMessage() {
    Flash f = FacesContext.getCurrentInstance()
        .getExternalContext()
        .getFlash();

    f.put("fn", this.firstName);
    f.put("ln", this.lastName);

    return "results?faces-redirect=true";
}
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

78

78



Sử dụng flask

- Khi người dùng click vào button "Send" sẽ chuyển sang trang results.xhtml như sau:

```
<h:head>
    <f:metadata>
        <f:event type="preRenderView"
            listener="#{userBean.showMessage()}" />
    </f:metadata>
</h:head>
<h:body>
    <h:outputFormat value="Welcome {0} {1}!!!">
        <f:param value="#{userBean.firstName}" />
        <f:param value="#{userBean.lastName}" />
    </h:outputFormat>
</h:body>
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

79

79



Sử dụng flask

- Phương thức showMessage()

```
public void showMessage() {
    Flash f = FacesContext.getCurrentInstance()
        .getExternalContext()
        .getFlash();

    this.firstName = (String) f.get("fn");
    this.lastName = (String) f.get("ln");
}
```

Demo

localhost:8080/surveyweb/faces/user.xhtml

First Name: Thanh

Last Name: Duong

Send

localhost:8080/surveyweb/faces/results.xhtml

Welcome Thanh Duong!!!

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

80

80

LẬP TRÌNH JAVA

XÂY DỰNG WEB BÁN HÀNG JSF + HIBERNATE

ThS. Dương Hữu Thành
Khoa CNTT, Đại học Mở Tp.HCM
thanh.dh@ou.edu.vn



81

81



Cơ sở dữ liệu



Thiết kế HTML Template

Java

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

83

83



Tạo HibernateUtil để cấu hình

Java

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

84

84



Tạo các Persistent Class

- Ghi đè các phương thức `toString`, `equals`, `hashCode`



Tạo các lớp Service tương tác CSDL

- Tạo gói `com.dht.service`
 - `CategoryService`
 - `ManufacturerService`
 - `ProductService`



Phát triển các chức năng

- Nạp danh sách danh mục
- Nạp danh sách nhà sản xuất
- Quản lý sản phẩm
 - Thêm, xoá, sửa sản phẩm
 - Tìm kiếm sản phẩm
- Hiển thị danh sách sản phẩm phía client, cho phép tìm kiếm theo tên và các khoảng giá.
- Giỏ hàng.



Tạo các lớp Managed Bean

- Tạo gói com.dht.bean
 - CommonBean
 - ProductBean



Upload File

- Thẻ <h:inputFile />
 - value: tập tin được upload javax.servlet.http.Part
 - required: bắt buộc cung cấp giá trị.
 - validator: chỉ định validator.
 - converter: chỉ định converter.
 - valueChangeListener: chỉ định phương thức được gọi khi giá trị thay đổi.



Upload File

- Tạo form phía client
 - Thuộc tính enctype phải là multipart/form-data

```
<h:form enctype="multipart/form-data">
    <div class="form-group">
        <h:outputLabel value="Ảnh đại diện" />
        <h:inputFile styleClass="form-control"
                    value="#{productBean.file}" />
    </div>
    <div class="form-group">
        <h:commandButton value="Tải tập tin"
                        action="#{productBean.uploadFile()}"
                        styleClass="btn btn-warning" />
    </div>
</h:form>
```



Upload File

- Trong Managed Bean khai báo thuộc tính file kiểu Part.

```
@ManagedBean  
@Named(value = "productBean")  
@RequestScoped  
public class ProductBean implements Serializable {  
    private Part file;  
  
    public String uploadFile() {  
    }  
  
    // Các phương thức getter/setter  
}
```



Upload File

- Lấy dữ liệu upload dạng stream bằng phương thức getInputStream() của đối tượng Part.

```
public String uploadFile() throws IOException {  
    String path = FacesContext.getCurrentInstance()  
        .getExternalContext()  
        .getRealPath("/resources/images/upload")  
        + "/" + this.file.getSubmittedFileName();  
    try (InputStream in = this.file.getInputStream();  
        FileOutputStream out = new FileOutputStream(path)) {  
        int byteRead;  
        byte[] chunk = new byte[1024];  
        while ((byteRead = in.read(chunk)) != -1)  
            out.write(chunk, 0, byteRead);  
    }  
    return "index";  
}
```



Tạo Validator cho Upload File

```
@FacesValidator(value = "UploadValidator")
public class UploadValidator implements Validator {
    @Override
    public void validate(FacesContext fc, UIComponent uic,
                         Object value) throws ValidatorException {
        Part p = (Part) value;
        // Ảnh phải là png hoặc jpg
        if (!p.getContentType().equals("image/png") &&
            !p.getContentType().equals("image/jpg")) {
            FacesMessage msg = new FacesMessage("Need png/jpg");
            throw new ValidatorException(msg);
        }
        // Dung lượng phải nhỏ hơn hoặc bằng (2MB)
        if (p.getSize() > 2097152) {
            FacesMessage msg = new FacesMessage("Size <= 2MB");
            throw new ValidatorException(msg);
        }
    }
}
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

93

93



Tạo giỏ hàng

- Tạo Managed Bean đặt tên CartBean, phương thức @PostConstruct tạo cart kiểu Map rỗng trong session.

```
@PostConstruct
public void init() {
    ExternalContext ctx = FacesContext.getCurrentInstance()
        .getExternalContext();
    if (ctx.getSessionMap().get("cart") == null)
        ctx.getSessionMap().put("cart", new HashMap<>());
}
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

94

94



Tạo giỏ hàng

```
public String addToCart(int proId, String proName, BigDecimal price)
    ExternalContext ctx = FacesContext.getCurrentInstance()
        .getExternalContext();

    Map<Integer, Object> cart
        = (Map<Integer, Object>) ctx.getSessionMap().get("cart");
    if (cart.get(proId) == null) {
        Map<String, Object> p = new HashMap<>();
        p.put("productId", proId);
        p.put("productName", proName);
        p.put("productPrice", price);
        p.put("count", 1);
        cart.put(proId, p);
    } else {
        Map<String, Object> p
            = (Map<String, Object>) cart.get(proId);
        p.put("count",
            Integer.parseInt(p.get("count").toString()) + 1);
    }
    return "successful";
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

95

95



- Tại facelet thực thi chức năng thêm sản phẩm vào giỏ.

```
<h:form>
    <h:commandButton class="btn btn-info" value="Thêm vào giỏ">
        <f:ajax
            listener="#{cartBean.addToCart(1, 'iPhone 7', 18)}" />
    </h:commandButton>
</h:form>
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

96

96



- Tính mã băm mật khẩu

```
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.14</version>
</dependency>
```



Đăng ký

- Trong lớp UserService viết phương thức cho phép đăng ký thông tin người dùng mới, trong đó mật khẩu được băm theo thuật toán MD5.

```
User u = new User();
u.setName(name);
u.setEmail(email);
u.setUsername(username);
u.setUserRole("USER");
u.setPassword(DigestUtils.md5Hex(password));
try (Session session
        = HibernateUtil.getSessionFactory().openSession()) {
    session.beginTransaction();
    session.save(u);
    session.getTransaction().commit();
}
```



Đăng nhập

- Tạo bộ lọc (filter) request kiểm tra quyền người dùng được phép truy cập các chức năng hệ thống.

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

99

99



Đăng nhập

- Trong <h:head> của trang login kiểm tra người dùng ở trạng thái login sẽ chuyển về trang chủ.

```
<f:metadata>
    <f:viewAction action="#{userBean.checkLogin()}" />
</f:metadata>
```

- Phương thức checkLogin() trong UserBean

```
public String checkLogin() {
    if (FacesContext.getCurrentInstance()
        .getExternalContext()
        .getSessionMap().get("user") != null)
        return "index?faces-redirect=true";

    return null;
}
```

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

100

100



JSF 2.3

- <https://dzone.com/articles/javaserver-faces-23-1>

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

101

101

Q&A

Dương Hữu Thành, Khoa CNTT, Đại học Mở Tp.HCM

10
2

102