# ASDV 2620, Web App II

# Standard JSF Converters-Validation

**Conversions ( Reference, Geary & Horstman textbook, chapter 7)**

**The JSF Conversion and Validation Process** The user fills in a field of a web form. When the user clicks the submit button, the browser sends the request value to the server, using an HTTP request. In the *Apply Request Values phase*, the JSF implementation stores the request values in component objects.

The request value is stored in the component object is called a submitted value.

**1. All request values are strings.**

The web application deals with all types, such as int, Date, or custom types.

A conversion process transforms the incoming strings to those types.
The converted values are not immediately transmitted to the corresponding beans. Instead, they are first stored inside the component objects as **local values**.

**2. After conversion, the local values are validated.**

After **all local values have been validated**, the *Update Model* Values phase starts, and the local values are stored in beans, as specified by their value references.
JSF uses a two-step approach to make it easier to preserve model integrity. As all programmers know only too well, users enter wrong information with distressing regularity. Suppose some of the model values had been updated before the first user error was detected. The model might then be in an inconsistent state, and it would be tedious to bring it back to its old state. **For that reason, JSF first converts and validates ALL user input.**
From the browser to the serverside component object and finally to the model bean.

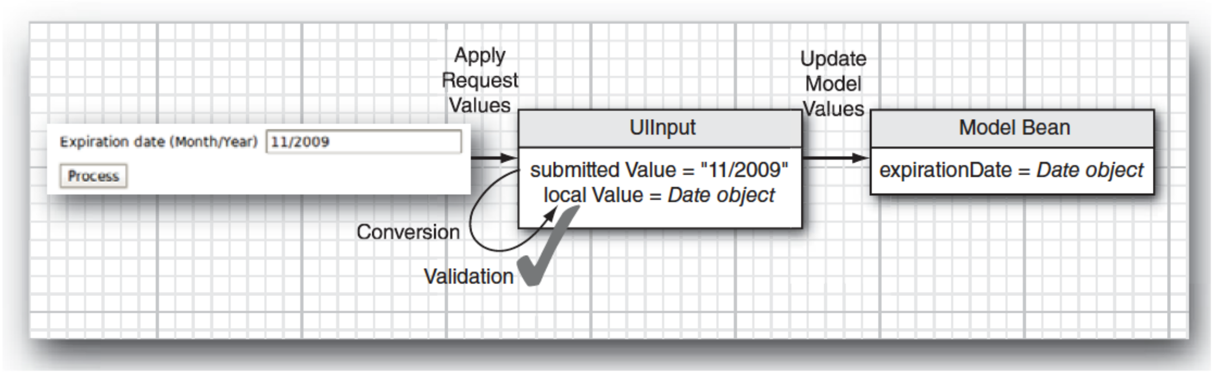

Expiration date (Month/Year)
11/2009
Process
Apply Request Values
UIInput
submitted Value = "11/2009"
local Value = Date object
Conversion
Validation
Update Model Values
Model Bean
expirationDate = Date object

**Figure 7–1 A value travels from the browser to the model**

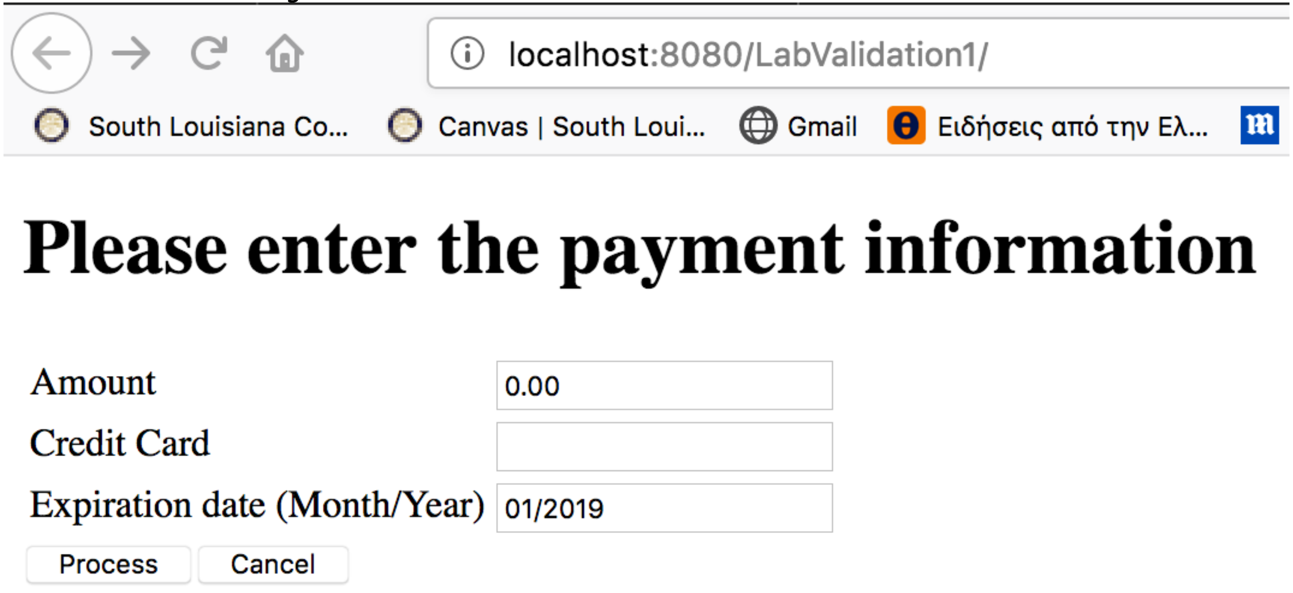
**Using Standard Converters**

Conversion of Numbers and Dates
A web application stores data of many types, but the web user interface deals exclusively with strings.

For *example*, suppose the user needs to edit a Date object that is stored in the business logic. First, the Date object is converted to a string that is sent to the client browser to be displayed inside a text field. The user then edits the text field. The resulting string is returned to the server and must be converted back to a Date object.

The same situation holds, of course, for primitive types, such as int, double, or boolean. The user of the web application edits strings, and the JSF container needs to convert the string to the type required by the application.

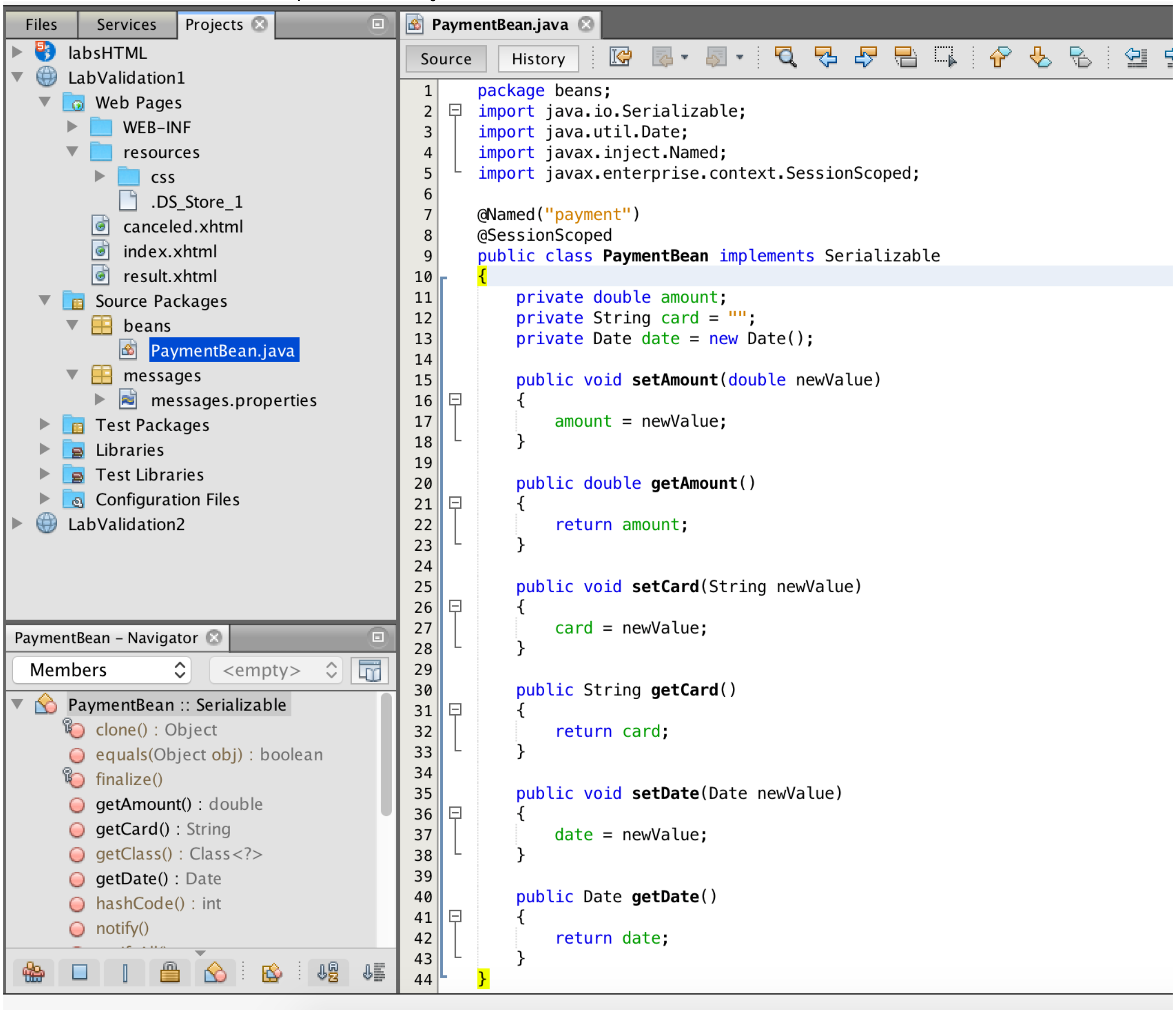
1. Create a new web app called LabValidation1. We will use standard JSF converters to validate numbers, strings and dates.



Please enter the payment information

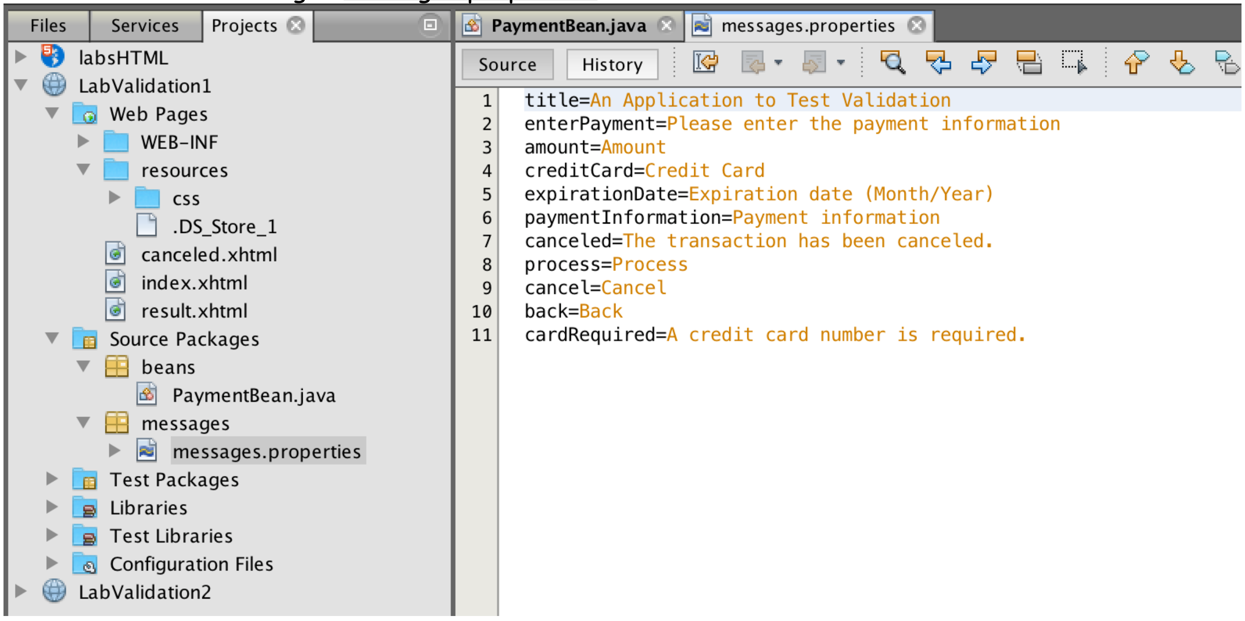
|  |  |
|---|---|
| Amount | 0.00 |
| Credit Card |  |
| Expiration date (Month/Year) | 01/2019 |

2. Create the bean PaymentBean.java as shown below:



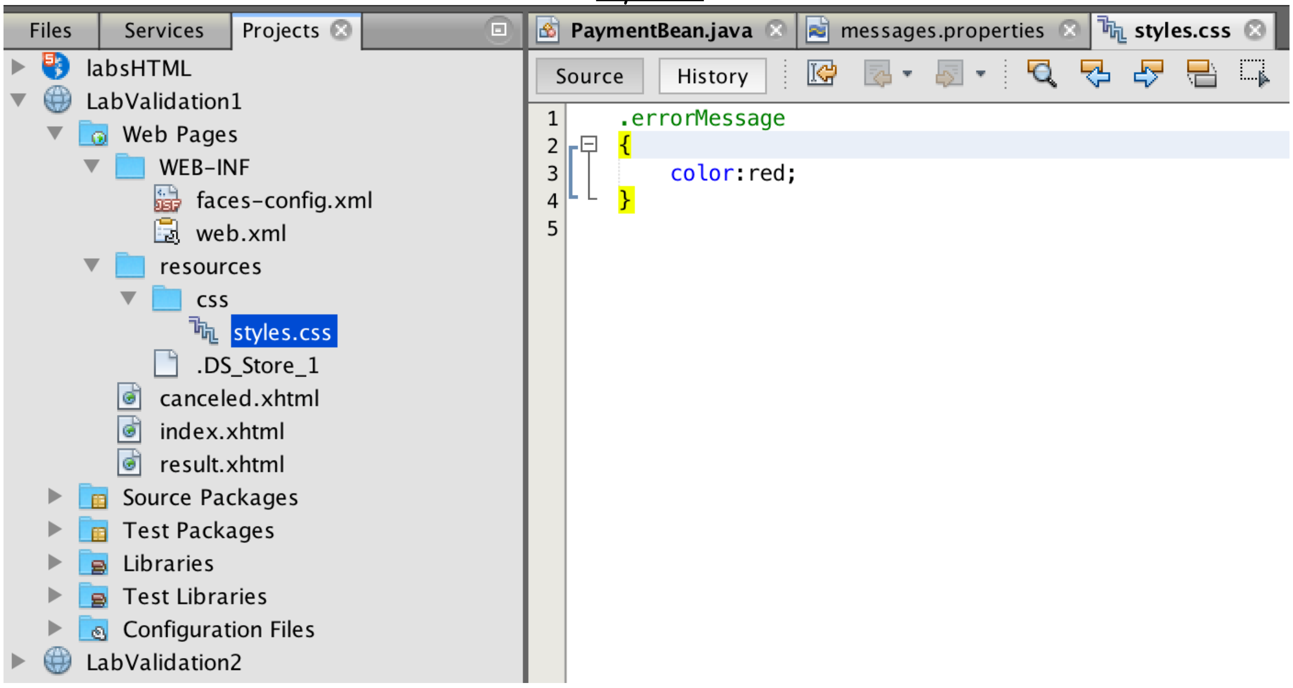
```java
package beans;
import java.io.Serializable;
import java.util.Date;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

@Named("payment")
@SessionScoped
public class PaymentBean implements Serializable
{
    private double amount;
    private String card = "";
    private Date date = new Date();

    public void setAmount(double newValue)
    {
        amount = newValue;
    }

    public double getAmount()
    {
        return amount;
    }

    public void setCard(String newValue)
    {
        card = newValue;
    }

    public String getCard()
    {
        return card;
    }

    public void setDate(Date newValue)
    {
        date = newValue;
    }

    public Date getDate()
    {
        return date;
    }
}
```

3. Create the messages messages.properties file:



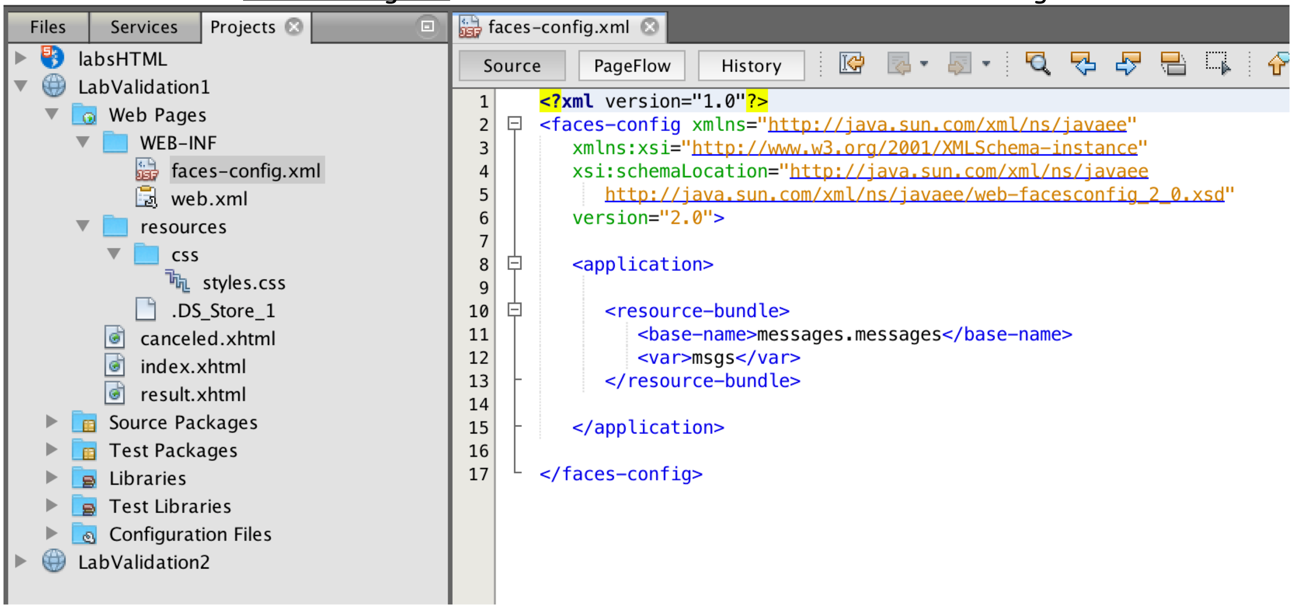
```
title=An Application to Test Validation
enterPayment=Please enter the payment information
amount=Amount
creditCard=Credit Card
expirationDate=Expiration date (Month/Year)
paymentInformation=Payment information
canceled=The transaction has been canceled.
process=Process
cancel=Cancel
back=Back
cardRequired=A credit card number is required.
```

4. Under folder resources\css create the style.css shown



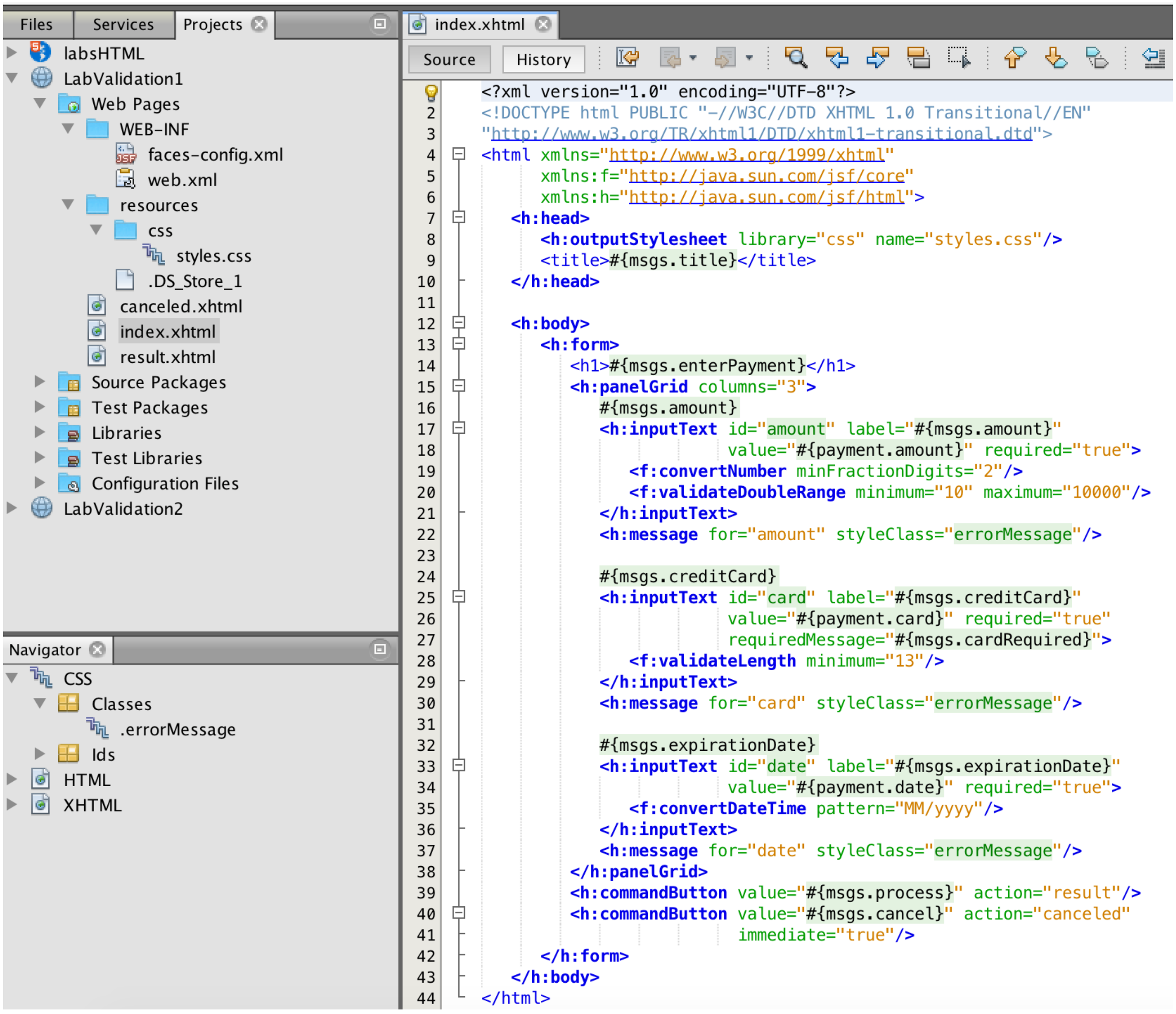
```
.errorMessage
{
    color:red;
}

```

5. Create the faces-config.xml and include the resource bundle for the messages:



```
<?xml version="1.0"?>
<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>messages.messages</base-name>
         <var>msgs</var>
      </resource-bundle>

   </application>

</faces-config>
```

6. The index.xhtml uses standard conversions, and errors messages at:
   lines 19-20, 22
   lines 28, 30
   lines 35, 37
   Observe that error all the messages shown in lines 22, 30 and 37 use the IDs of the component they display the message for.



```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 4  <html xmlns="http://www.w3.org/1999/xhtml"
 5        xmlns:f="http://java.sun.com/jsf/core"
 6        xmlns:h="http://java.sun.com/jsf/html">
 7      <h:head>
 8          <h:outputStylesheet library="css" name="styles.css"/>
 9          <title>#{msgs.title}</title>
10      </h:head>
11  
12      <h:body>
13          <h:form>
14              <h1>#{msgs.enterPayment}</h1>
15              <h:panelGrid columns="3">
16                  #{msgs.amount}
17                  <h:inputText id="amount" label="#{msgs.amount}"
18                               value="#{payment.amount}" required="true">
19                      <f:convertNumber minFractionDigits="2"/>
20                      <f:validateDoubleRange minimum="10" maximum="10000"/>
21                  </h:inputText>
22                  <h:message for="amount" styleClass="errorMessage"/>
23  
24                  #{msgs.creditCard}
25                  <h:inputText id="card" label="#{msgs.creditCard}"
26                               value="#{payment.card}" required="true"
27                               requiredMessage="#{msgs.cardRequired}">
28                      <f:validateLength minimum="13"/>
29                  </h:inputText>
30                  <h:message for="card" styleClass="errorMessage"/>
31  
32                  #{msgs.expirationDate}
33                  <h:inputText id="date" label="#{msgs.expirationDate}"
34                               value="#{payment.date}" required="true">
35                      <f:convertDateTime pattern="MM/yyyy"/>
36                  </h:inputText>
37                  <h:message for="date" styleClass="errorMessage"/>
38              </h:panelGrid>
39              <h:commandButton value="#{msgs.process}" action="result"/>
40              <h:commandButton value="#{msgs.cancel}" action="canceled"
41                               immediate="true"/>
42          </h:form>
43      </h:body>
44  </html>
```

**Table 7–1 Attributes of the f:convertNumber Tag**

| Attribute | Type | Value |
|---|---|---|
| type | String | number (default), currency, or percent |
| pattern | String | Formatting pattern, as defined in java.text.DecimalFormat |
| maxFractionDigits | int | Maximum number of digits in the fractional part |
| minFractionDigits | int | Minimum number of digits in the fractional part |
| maxIntegerDigits | int | Maximum number of digits in the integer part |
| minIntegerDigits | int | Minimum number of digits in the integer part |
| integerOnly | boolean | True if only the integer part is parsed (default: false) |
| groupingUsed | boolean | True if grouping separators are used (default: true) |
| locale | java.util.Locale or String | Locale whose preferences are to be used for parsing and formatting |
| currencyCode | String | ISO 4217 currency code, such as USD or EUR, for selecting a currency converter |
| currencySymbol | String | This string is passed to DecimalFormat.setDecimalFormatSymbols, overriding the locale-based symbol; not recommended—use currencyCode instead |

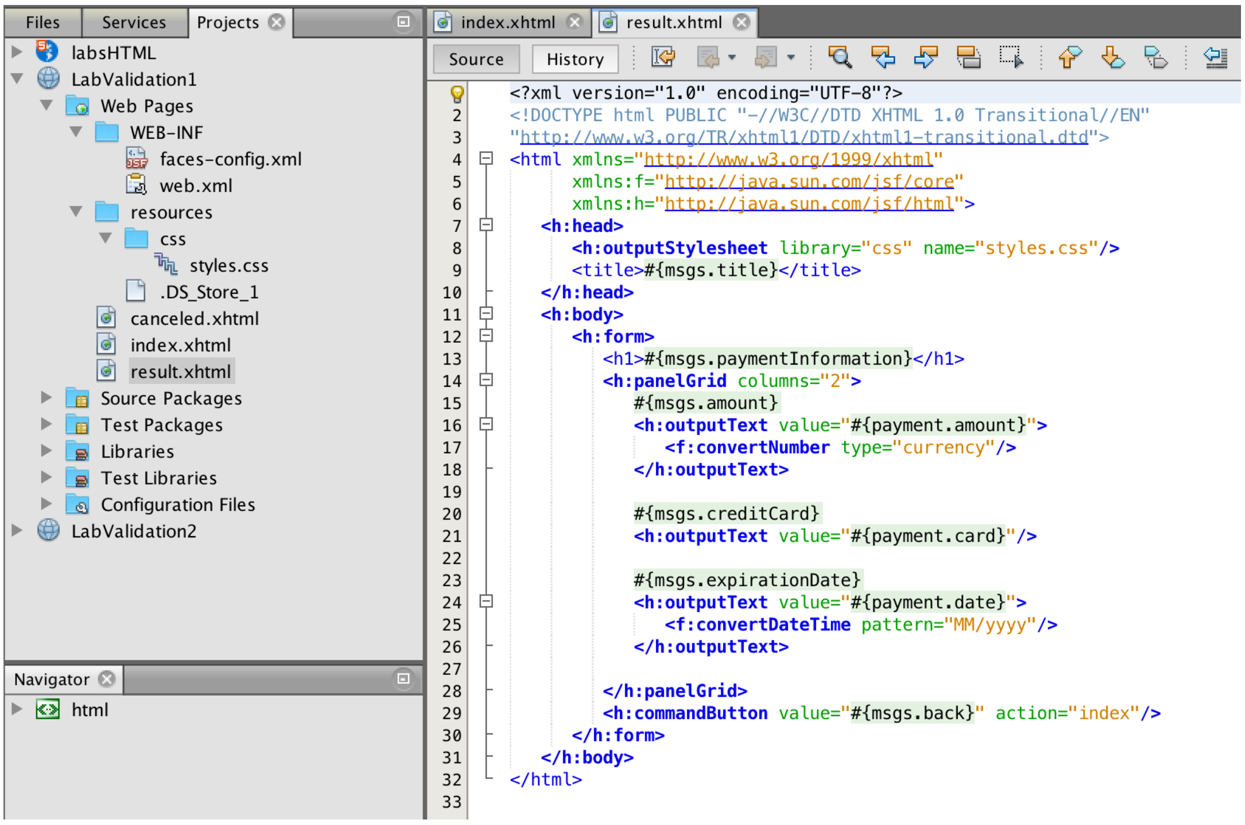
7. Tag f:convertNumber

8. Tag f:convertDateTime

**Table 7–2 Attributes of the `f:convertDateTime` Tag**

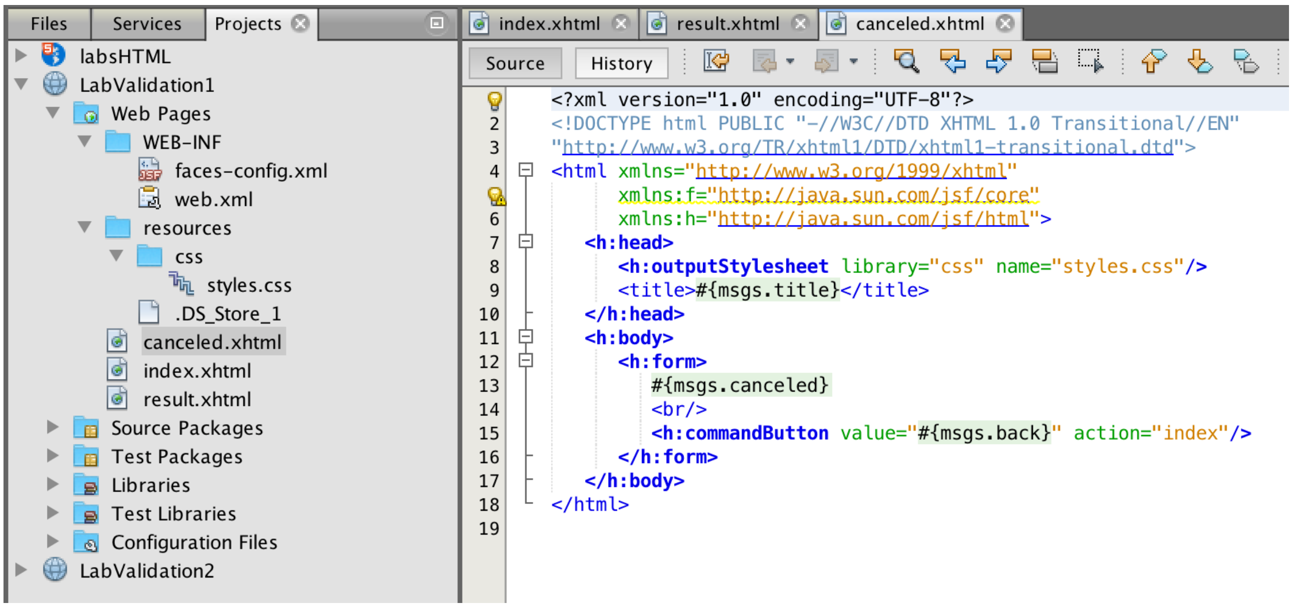
| Attribute | Type | Value |
|---|---|---|
| `type` | `String` | `date` (default), `time`, or `both` |
| `dateStyle` | `String` | `default`, `short`, `medium`, `long`, or `full` |
| `timeStyle` | `String` | `default`, `short`, `medium`, `long`, or `full` |
| `pattern` | `String` | Formatting pattern, as defined in `java.text.SimpleDateFormat` |
| `locale` | `java.util.Locale` or `String` | Locale whose preferences are to be used for parsing and formatting |
| `timeZone` | `java.util.TimeZone` | Time zone to use for parsing and formatting; if you do not supply a time zone, the default is GMT<br><br>Note: As of JSF 2.0, you can change the default to `TimeZone.getDefault()` by setting `javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE` to `true` in `web.xml`. |

9. The result.xhtml with its validations:



```
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <h:outputStylesheet library="css" name="styles.css"/>
        <title>#{msgs.title}</title>
    </h:head>
    <h:body>
        <h:form>
            <h1>#{msgs.paymentInformation}</h1>
            <h:panelGrid columns="2">
                #{msgs.amount}
                <h:outputText value="#{payment.amount}">
                    <f:convertNumber type="currency"/>
                </h:outputText>

                #{msgs.creditCard}
                <h:outputText value="#{payment.card}"/>

                #{msgs.expirationDate}
                <h:outputText value="#{payment.date}">
                    <f:convertDateTime pattern="MM/yyyy"/>
                </h:outputText>

            </h:panelGrid>
            <h:commandButton value="#{msgs.back}" action="index"/>
        </h:form>
    </h:body>
</html>
```

10. The canceled.xhtml



```
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <h:outputStylesheet library="css" name="styles.css"/>
        <title>#{msgs.title}</title>
    </h:head>
    <h:body>
        <h:form>
            #{msgs.canceled}
            <br/>
            <h:commandButton value="#{msgs.back}" action="index"/>
        </h:form>
    </h:body>
</html>
```