Validating User Input Part 1: Basics

JSF 2.2 Version

Topics in This Section

- **Aligning prompts, fields, error messages**
  - h:panelGrid
- **Enforcing that certain fields are filled in**
  - required, requiredMessage, h:messages
- **Putting error messages next to fields**
  - h:message
- **Verifying that input is of the right type**
  - non-String bean properties, converterMessage
- **Checking that values are in the right range or match regular expression**
  - f:validate*Blah* tags, validatorMessage
- **Putting “Fix Errors Below” message at top**
  - Only when there is at least one error in form
  - Moving code to composite component

Preview of Next Section

- **Manual validation**
  - Checking in the action controller method
- **Using a custom validator method**
  - <h:input*Blah* validator="#{someBean.someMethod"/>
- **Using a custom validator component (?)**
  - <f:validator validatorId="someId"/>
- **Localizing (internationalizing) error messages**
  - *blah*Message="#{messages.messageName}"
- **Using the Apache MyFaces validators**
  - For URLs, email addresses, credit cards, cross-field comparisons
- **Using the Apache MyFaces validators with the PrimeFaces extended GUI components**
  - If using 3rd party libraries anyhow, usually use both

Overview

The Need for Form-Field Validation

- **Two tasks that almost every Web application needs to perform:**
  - Checking that all required form fields are present and in the proper format
  - Redisplaying the form when values are missing or malformed
    - With error messages showing what the problem was
    - With valid values maintained in the form
- **This was extremely cumbersome with standard servlet/JSP technology**
  - Even with the JSP 2.0 expression language
  - This is a (the?) major weakness in servlet/JSP technology, and one of main motivations of frameworks

Thumbnail Summary of Entire Section

- **Example code**

```
<h:inputText value="..." required="true" requiredMessage="..."
             converterMessage="..." validatorMessage="..." id="someId">
    <f:validateBlah .../>
</h:inputText>
<h:message for="someId"/>
```

- **Interpretation**
  - If submitted with field empty
    - Redisplay form and show requiredMessage
  - If submitted with field filled in, but with illegal type
    - Redisplay form and show converterMessage
  - If submitted with field of right type but wrong range
    - Redisplay form and show validatorMessage

Quick Aside: h:panelGrid

h:panelGrid – Summary

- **Idea**
  – h:panelGrid is a shortcut for making an HTML <table>. Commonly used to keep prompts, input fields, and error messages together on same line.
- **Syntax**
  – You specify number of columns (default is 1), then each element is placed in a single cell.

```
<h:panelGrid columns="3">
    Prompt 1: <h:inputText…/> <h:message…/>
    Prompt 2: <h:inputText…/> <h:message…/>
 </h:panelGrid>
```

  – Any amount of regular HTML is considered a single element, but you can break it up by using h:outputText.
  – Group multiple elements into single cell with h:panelGroup
- **Rowspan and colspan**
  – Not supported with **h**:panelGrid, but is with **p**:panelGrid (covered in PrimeFaces tutorial). But, you can achieve colspan for *last* entry by putting it below the table. And, of course, nested tables are possible.

h:panelGrid – Typical Usage for Forms with Messages at Top

```
<div align="center">
<h:messages styleClass="someCssName"/>
<h:panelGrid columns="2">
  Field1 Prompt:
  <h:inputText .../>

  Field2 Prompt:
  <h:inputText .../>

  ...
</h:panelGrid>
<h:commandButton value="Centered Button" .../>
</div>
```

Minor CSS trick: I use <h:panelGrid ... styleClass="formTable">.
This makes the first column (the prompts) right-aligned instead of left-aligned.
See my CSS file in the downloadable projects for the definition of formTable.

h:panelGrid – Typical Usage for Forms with Messages by Fields

```
<div align="center">
<h:panelGrid columns="3">
  Field1 Prompt:
  <h:inputText ... id="field1"/>
  <h:message for="field1" .../>

  Field2 Prompt:
  <h:inputText ... id="field2"/>
  <h:message for="field2" .../>


  ...
</h:panelGrid>
<h:commandButton value="Centered Button" .../>
</div>
```

Minor CSS trick: I use <h:panelGrid ... styleClass="formTable">.
In addition to making the first column (the prompts) right-aligned instead of left-aligned, formTable makes the third column (the error message, if any) bold and red. See my CSS file in the downloadable projects for the definition of formTable.

Enforcing Required Fields

(and Putting Error Messages at Top of Form)

Overview

- **Approach**
  - Designate fields that cannot be empty on submission
  - Supply error message
  - Put h:messages inside form
- **Behavior**
  - If form submitted with any of the required fields empty
    - Setter methods and action controller method are blocked
    - Form is redisplayed
    - Error messages are shown

Details

- **Steps**
  - Use required="true" to designate required field
  - Use requiredMessage to designate error message
  - Use <h:messages/> to display error message(s) in ul list
    - h:messages has many options re layout (table or list) and types of messages to display. But most common approach is to supply only a CSS class name.
- **Example**

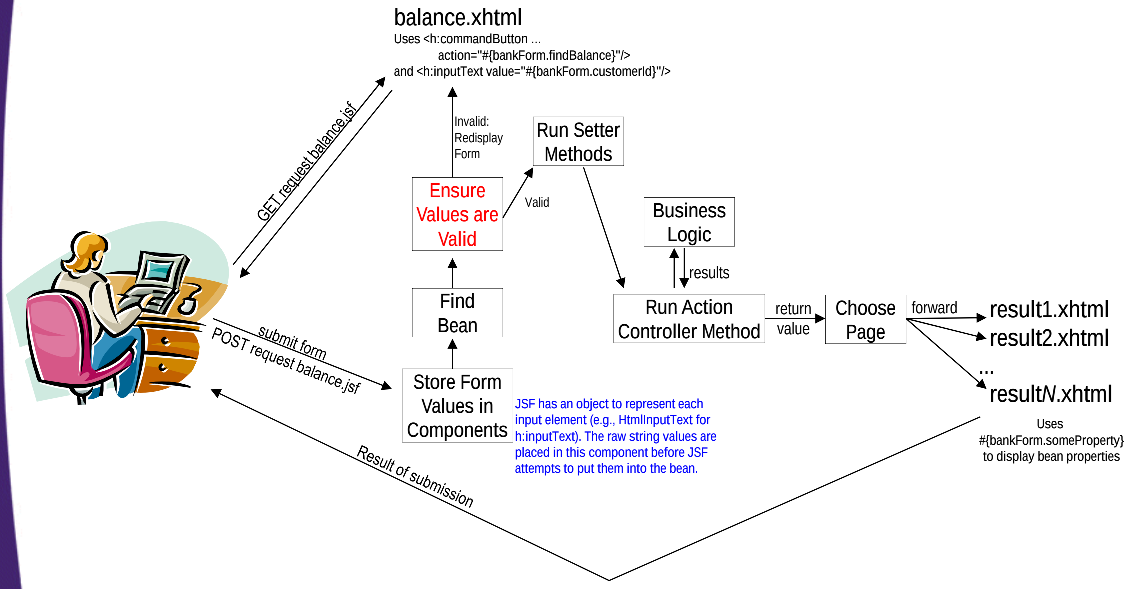
```
<h:form>
<h:messages styleClass="some-css-name"/>
<h:inputText value="#{someBean.someProperty}"
             required="true"
             requiredMessage="Some error message"
…
</h:form>
```

Note that it is h:messages, plural. We will see h:message (singular) in next section for putting each error message next to corresponding field, instead putting of all error messages at the top of the form.

Importance of requiredMessage

- **If requiredMessage supplied**
  - That exact text will be displayed as the error message
  - Easy to localize (internationalize) error messages
    - requiredMessage="#{messages.yourErrorName}“
      - See example in second tutorial section on validation
- **If requiredMessage omitted**
  - A standard error message will be displayed
    - Usually cumbersome and hard to read
    - Not customized for your field
  - You can override the error messages from the builtin Messages.properties file by loading a properties file with certain names (e.g., here, javax.faces.component.UIInput.REQUIRED), but then you get the same error message for all required fields
- **Bottom line**
  - *Always* supply requiredMessage when you have required="true"

JSF Flow of Control (Simplified)



```
Uses <h:commandButton ...
        action="#{bankForm.findBalance}"/>
and <h:inputText value="#{bankForm.customerId}"/>
```

JSF has an object to represent each input element (e.g., HtmlInputText for h:inputText). The raw string values are placed in this component before JSF attempts to put them into the bean.

Uses #{bankForm.someProperty} to display bean properties

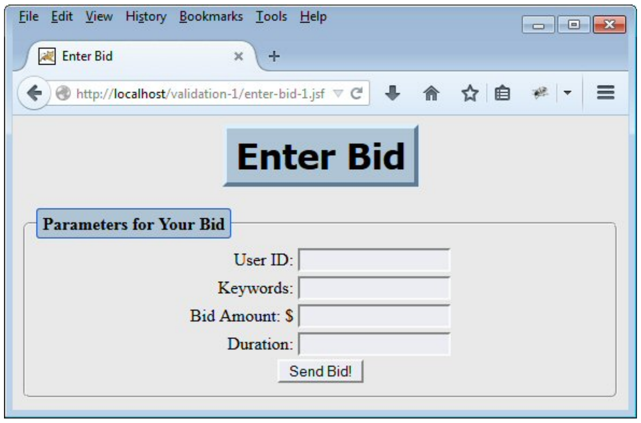
“Valid” here means non-empty (if “required” is set).
Upcoming sections will add other meanings to “valid”.

Precedence of Validation Tests

- **Required**
  - If you mark a field as “required”, and end user omits a value, then error message for missing type is generated
    - requiredMessage is used
      - Warning: if your setter expects a String, whitespace satisfies “required”. I.e, whitespace in a field is not considered empty.
- **Type**
  - If field passes “required” validation (if any), then JSF checks if string can be converted to expected type
    - converterMessage is used
- **Validators**
  - If field passes required and type validation, then any explicit validators are checked
    - validatorMessage is used

Validation of Required Fields: Example

- **Idea**
  - Collect bids for keywords advertiser at search engine site
- **Attributes (all are Strings)**
  - UserID
    - Cannot be missing
  - Keyword
    - Cannot be missing
  - Bid amount
    - Cannot be missing
    - Converts internally to double
  - Bid duration
    - Cannot be missing
    - Converts internally to int


File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-1.jsf
Send Bid!

Enter Bid

Parameters for Your Bid

User ID:
Keywords:
Bid Amount: $
Duration:

Managed Bean: String Properties (No Conversion)

```
@ManagedBean
public class BidBean1 {
  private String userId = "";
  private String keyword = "";
  private String bidAmount;
  private double numericBidAmount = 0;
  private String bidDuration;
  private int numericBidDuration = 0;

  public String getUserId() { return(userId); }

  public void setUserId(String userId) {
    this.userId = userId;
  }

  public String getKeyword() { return(keyword); }

  public void setKeyword(String keyword) {
    this.keyword = keyword.trim();
  }
```

Managed Bean: Numeric Properties (Conversion)

```
  public String getBidAmount() { return(bidAmount); }

  public void setBidAmount(String bidAmount) {
    this.bidAmount = bidAmount;
    try {
      numericBidAmount = Double.parseDouble(bidAmount);
    } catch(NumberFormatException nfe) {}
  }
  public double getNumericBidAmount() {
    return(numericBidAmount);
  }

  public String getBidDuration() { return(bidDuration); }

  public void setBidDuration(String bidDuration) {
    this.bidDuration = bidDuration;
    try {
      numericBidDuration = Integer.parseInt(bidDuration);
    } catch(NumberFormatException nfe) {}
  }
  public int getNumericBidDuration() {
    return(numericBidDuration);
  }
```

Because of the try/catch blocks, that illegal values entered in the fields result in default values of 0 for bid amount and duration. It would be better to simply prohibit illegal values, so that is done in the upcoming section that uses non-String bean properties and converterMessage.

Managed Bean: Action Controller

```
private void doSomeBusinessLogicWithBid() {
  // Update database with bid, etc.
}

public String doBid() {
  doSomeBusinessLogicWithBid();
  return("show-bid-1");
}
```

Input Form: Top (enter-bid-1.xhtml)

```
...
<h:form>
<h:messages styleClass="error"/>
<h:panelGrid columns="2" styleClass="formTable">
  User ID:
  <h:inputText value="#{bidBean1.userId}"
               required="true"
               requiredMessage="You must enter a user ID"/>

  Keywords:
  <h:inputText value="#{bidBean1.keyword}"
               required="true"
               requiredMessage="You must enter a keyword"/>
```

Value of requiredMessage can be an EL expression, so you can use properties files for localized error messages. See tutorial section on properties and I18N.

Input Form: Continued (enter-bid-1.xhtml)

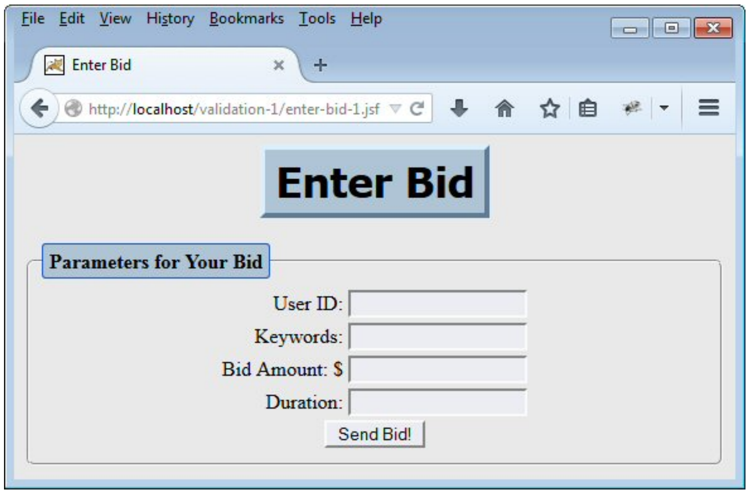
```
  Bid Amount: $
  <h:inputText value="#{bidBean1.bidAmount}"
                required="true"
                requiredMessage="You must enter an amount"/>

  Duration:
  <h:inputText value="#{bidBean1.bidDuration}"
               required="true"
               requiredMessage="You must enter a duration"/>
</h:panelGrid>
  <h:commandButton value="Send Bid!"
                   action="#{bidBean1.doBid}"/>
</h:form>
...
```

Results Page (show-bid-1.xhtml)

```
...
<h2>You have bid successfully.</h2>
<ul>
  <li>User ID: #{bidBean1.userId}</li>
  <li>Keywords: #{bidBean1.keyword}</li>
  <li>Bid Amount: $#{bidBean1.numericBidAmount}</li>
  <li>Duration: #{bidBean1.numericBidDuration}</li>
</ul>
...
```

This results page is also used by the next example (error messages next to fields).

Results (Initial Form)

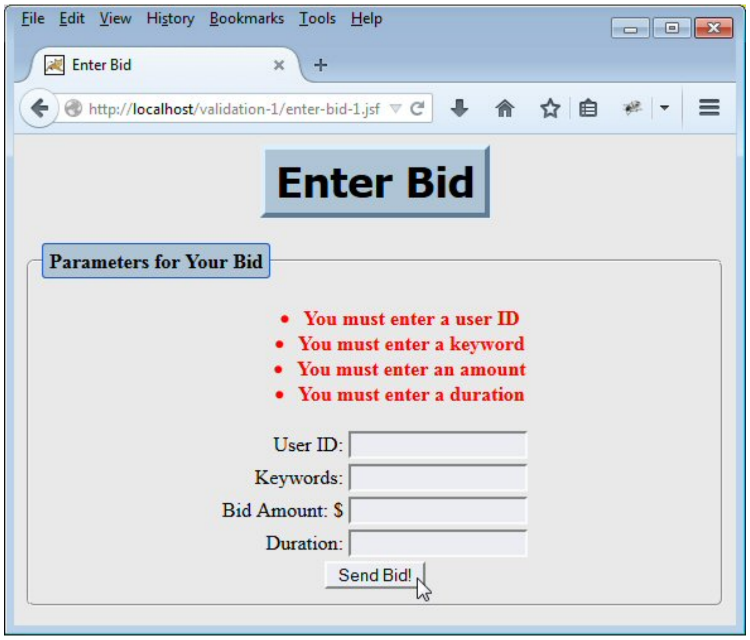

File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-1.jsf

Enter Bid

Parameters for Your Bid

User ID:	
Keywords:	
Bid Amount: $	
Duration:	

Results (Missing Input)


File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-1.jsf

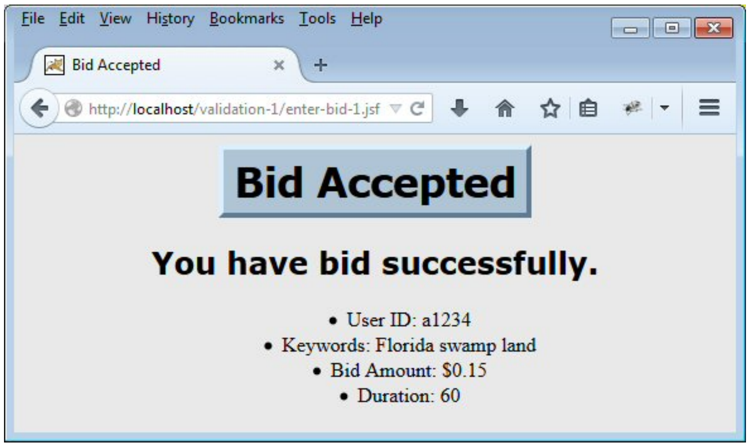
Enter Bid

Parameters for Your Bid

- You must enter a user ID
- You must enter a keyword
- You must enter an amount
- You must enter a duration

User ID:	
Keywords:	
Bid Amount: $	
Duration:	

Results (Good Input)


File Edit View History Bookmarks Tools Help
Bid Accepted
http://localhost/validation-1/enter-bid-1.jsf

Bid Accepted

You have bid successfully.

- User ID: a1234
- Keywords: Florida swamp land
- Bid Amount: $0.15
- Duration: 60

Putting Error Messages Next to Offending Fields

Problem with Previous Approach

- **All the error messages are at the top**
  - Fine if form is short, but not good if form is long, and user might have to scroll down to fix problem
    - Error message could scroll off screen
    - User might be confused as to which field has problem
- **Solution**
  - Give input fields ids
  - Use h:message and tie message to field with for="the-id"
  - Use HTML table (via h:panelGrid) to be sure error message appears beside the field, not below it
- **Example**

```
<h:inputText … id="field1"/>
<h:message for="field1"/>
```

Last example used h:messages;
this example uses h:message.

Example: Search Engine Advertiser Keywords Form

- **Bean code**
  - No changes from previous example
- **Results page**
  - No changes from previous example
- **Input page**
  - Removes h:messages from top of form
  - Gives IDs to each form field
  - Puts <h:message for="the-form-id"/> next to each field
  - Uses a 3-column table to be sure message stays next to field even if browser is small

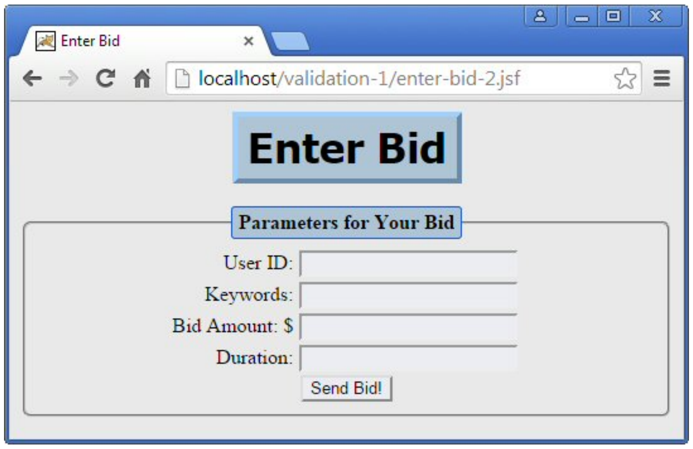
Input Form: Top (enter-bid-2.xhtml)

```
...
<h:form>
<h:panelGrid columns="3" styleClass="formTable">
  User ID:
  <h:inputText value="#{bidBean1.userId}"
               required="true"
               requiredMessage="You must enter a user ID"
               id="userId"/>
  <h:message for="userId"/>

  Keywords:
  <h:inputText value="#{bidBean1.keyword}"
               required="true"
               requiredMessage="You must enter a keyword"
               id="keyword"/>
  <h:message for="keyword"/>
```

Input Form: Continued (enter-bid-2.xhtml)

```
  Bid Amount: $
  <h:inputText value="#{bidBean1.bidAmount}"
                required="true"
                requiredMessage="You must enter an amount"
                id="amount"/>
  <h:message for="amount"/>

  Duration:
  <h:inputText value="#{bidBean1.bidDuration}"
               required="true"
               requiredMessage="You must enter a duration"
               id="duration"/>
  <h:message for="duration"/>
</h:panelGrid>
  <h:commandButton value="Send Bid!"
                   action="#{bidBean1.doBid}"/>
</h:form>
...
```

Results (Initial Form)


Enter Bid

localhost/validation-1/enter-bid-2.jsf

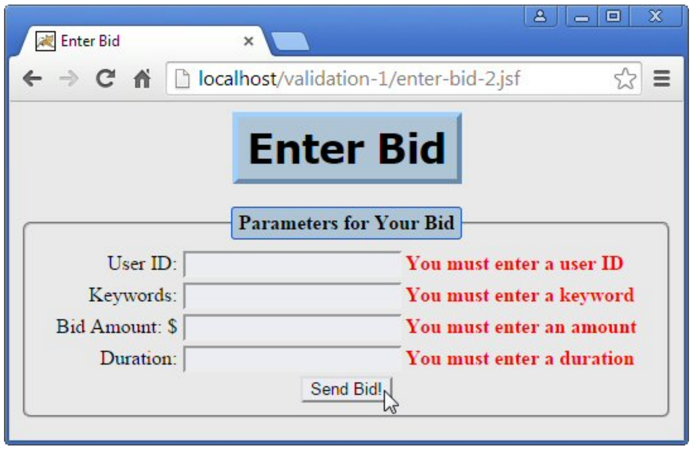
Enter Bid

Parameters for Your Bid

User ID:	
Keywords:	
Bid Amount: $	
Duration:	

Send Bid!

Results (Missing Input)


Enter Bid

localhost/validation-1/enter-bid-2.jsf

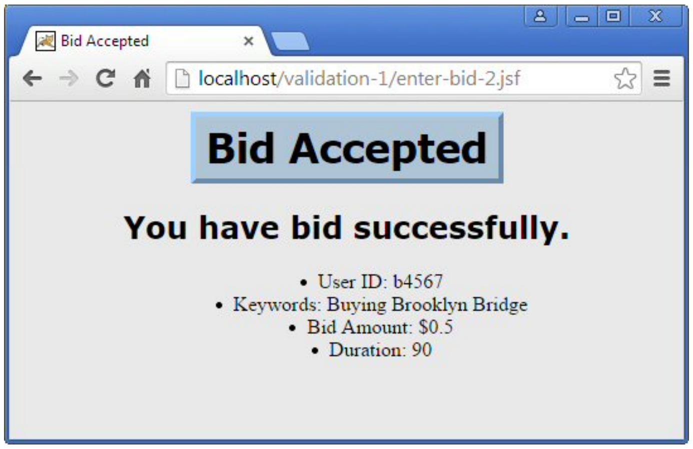
Enter Bid

Parameters for Your Bid

User ID:		You must enter a user ID
Keywords:		You must enter a keyword
Bid Amount: $		You must enter an amount
Duration:		You must enter a duration

Send Bid!

Results (Good Input)



Bid Accepted

You have bid successfully.

- User ID: b4567
- Keywords: Buying Brooklyn Bridge
- Bid Amount: $0.5
- Duration: 90

Checking Field Types

First Problem with Previous Approach

- **Developer had to parse Strings to numbers**
  – Tedious and time consuming to use try/catch blocks with Double.parseDouble, Integer.parseInt, etc.
- **Solution**
  – Make bean properties be Integer, Double, etc. JSF will convert automatically
    - Wrapper types (Integer, Double, etc.) are usually preferable to primitive types (int, double, etc.) so that textfield can be initially empty, which happens only when getter returns null or empty String.
- **Example**

```
public Double getBidAmount() { return(bidAmount); }
public void setBidAmount(Double bidAmount) {
   this.bidAmount = bidAmount; }
```

Second Problem with Previous Approach

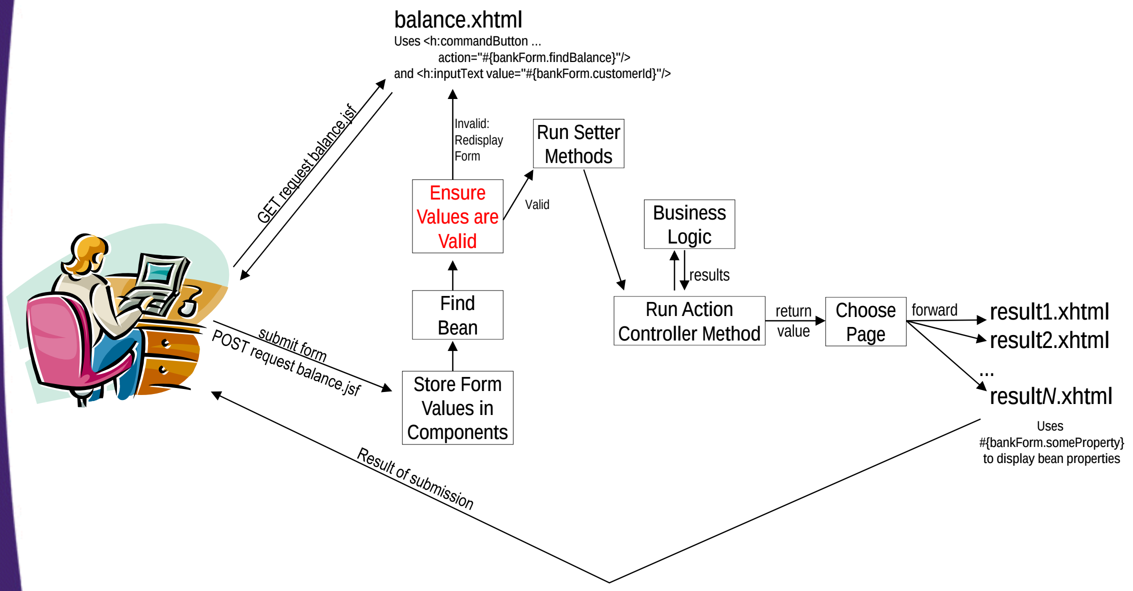
- **Illegal entries resulted in default values**
  – If user entered “blah” or another non-numeric value, exception was thrown from parsing code, and number field remained at its initial value (0 in this case). In most cases, this behavior would be unexpected to user.
- **Solution**
  – Once you use numeric bean properties, JSF will automatically redisplay form if parsing fails. You need to set and display converterMessage so user understands the problem.
- **Example**

```
<h:inputText value="#{someBean.someNumericProperty}"
             required="true" requiredMessage="…"
             converterMessage="Amount must be a number"
             id="amount"/>
<h:message for="amount"/>
```

Importance of converterMessage

- **If converterMessage supplied**
  - That exact text will be displayed as the error message
  - Easy to localize (internationalize) error messages
    - converterMessage="#{messages.yourErrorName}“
      - See example in second tutorial section on validation
- **If converterMessage omitted**
  - A standard error message will be displayed
    - Usually cumbersome and hard to read
    - Not customized for your field
  - You can override the error messages from the builtin Messages.properties file by loading a properties file with certain names (e.g., here, javax.faces.component.UIInput.CONVERSION), but then you get the same error message for all failed conversions
- **Bottom line**
  - *Always* supply converterMessage when you have numeric fields

JSF Flow of Control (Simplified)



“Valid” now means non-empty (if “required” is set) and able to be converted to the type that the setter method expects.

Example: Search Engine Advertiser Keywords Form

- **Bean code**
  - Make two non-String bean properties
    - bidAmount is Double
    - bidDuration is Integer
      - Use wrapper types so that textfields are blank initially. If you use double or int, *some* number must be in textfield
- **Input page**
  - Add converterMessage to bid amount and bid duration fields
- **Results page**
  - Same as previous example except for bean name

Managed Bean: Properties

```
public class BidBean2 {
  private String userId, keyword;
  private Double bidAmount;
  private Integer bidDuration;

  // Same accessors for userId and keyword as before

  public Double getBidAmount() { return(bidAmount); }

  public void setBidAmount(Double bidAmount) {
    this.bidAmount = bidAmount;
  }

  public Integer getBidDuration() { return(bidDuration); }

  public void setBidDuration(Integer bidDuration) {
    this.bidDuration = bidDuration;
  }
```

Managed Bean: Action Controller

```
private void doSomeBusinessLogicWithBid() {
  // Update database with bid, etc.
}

public String doBid() {
  doSomeBusinessLogicWithBid();
  return("show-bid-2");
}
```

Input Form: Top (enter-bid-3.xhtml)

```
<h:form>
<h:panelGrid columns="3" styleClass="formTable">
  User ID:
  <h:inputText value="#{bidBean2.userId}"
               required="true"
               requiredMessage="You must enter a user ID"
               id="userId"/>
  <h:message for="userId"/>

  Keywords:
  <h:inputText value="#{bidBean2.keyword}"
               required="true"
               requiredMessage="You must enter a keyword"
               id="keyword"/>
  <h:message for="keyword"/>
```

Input Form: Continued (enter-bid-3.xhtml)

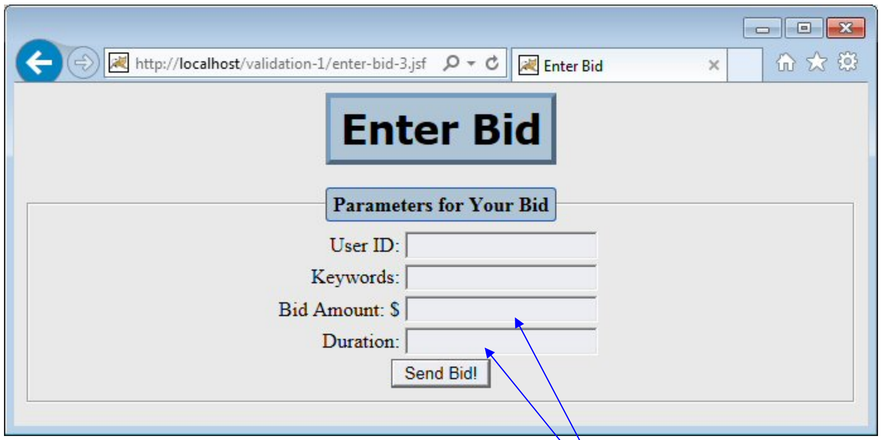
```
  Bid Amount: $
  <h:inputText value="#{bidBean2.bidAmount}"
               required="true"
               requiredMessage="You must enter an amount"
               converterMessage="Amount must be a number"
               id="amount"/>
  <h:message for="amount"/>

  Duration:
  <h:inputText value="#{bidBean2.bidDuration}"
              required="true"
              requiredMessage="You must enter a duration"
              converterMessage="Duration must be a whole number"
              id="duration"/>
  <h:message for="duration"/>
</h:panelGrid>
  <h:commandButton value="Send Bid!"
                   action="#{bidBean2.doBid}"/>
</h:form>
```

Results Page (show-bid-2.xhtml)

```
...
<h1 class="title">Bid Accepted</h1>
<h2>You have bid successfully.</h2>
<ul>
  <li>User ID: #{bidBean2.userId}</li>
  <li>Keywords: #{bidBean2.keyword}</li>
  <li>Bid Amount: $#{bidBean2.bidAmount}</li>
  <li>Duration: #{bidBean2.bidDuration}</li>
</ul>
...
```

This results page is shared by all remaining examples in this tutorial section

Results (Initial Form)


http://localhost/validation-1/enter-bid-3.jsf

Enter Bid

Parameters for Your Bid

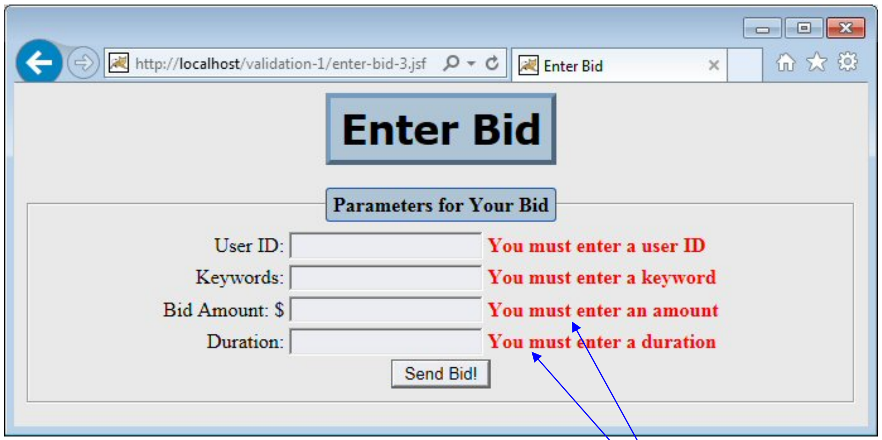
User ID:	
Keywords:	
Bid Amount: $	
Duration:	

Send Bid!

Because we use Double and Integer for the methods, the fields can be initially blank (because value is null). If you use double or int, getter method will never return null, and textfield will never be empty when form comes up.

And remember that ever since Java 5, autoboxing lets you assign Integer to int and vice versa without explicit conversion (and same with Double/double).

Results (Missing Input)


http://localhost/validation-1/enter-bid-3.jsf

Enter Bid

Parameters for Your Bid

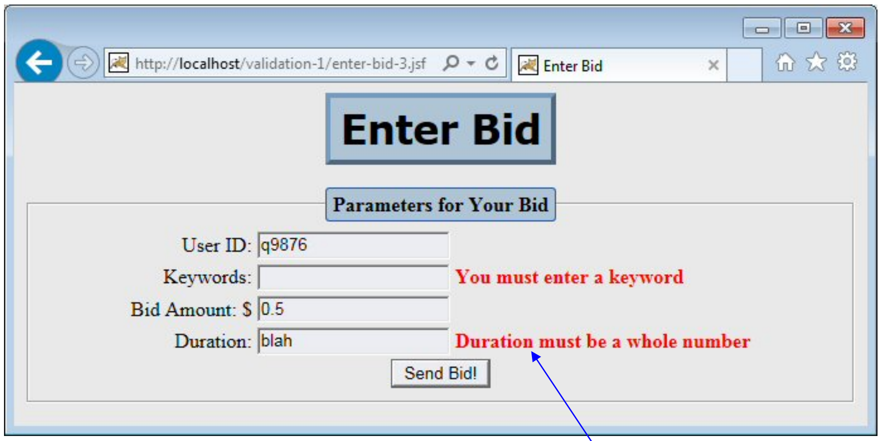
User ID:		**You must enter a user ID**
Keywords:		**You must enter a keyword**
Bid Amount: $		**You must enter an amount**
Duration:		**You must enter a duration**

Send Bid!

Since tests for required attributes take precedence over tests for proper types, the requiredMessage attribute is displayed here.

The error messages are bold and red due to the definition of the formTable CSS class in styles.css.

Results (Malformed Input)


http://localhost/validation-1/enter-bid-3.jsf

Enter Bid

Parameters for Your Bid

User ID: q9876

Keywords: You must enter a keyword

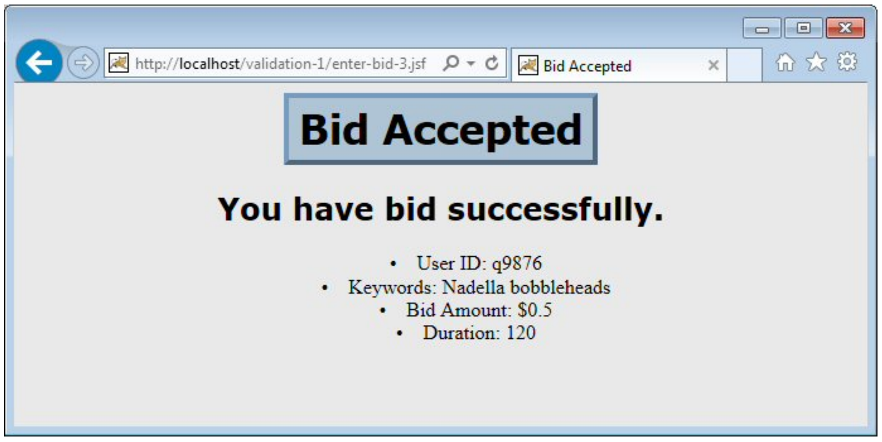
Bid Amount: $ 0.5

Duration: blah Duration must be a whole number

Send Bid!

Since the value passes the required test, type conversion is attempted, and if it fails, the converterMessage is shown

Results (Good Input)


http://localhost/validation-1/enter-bid-3.jsf

Bid Accepted

You have bid successfully.

- User ID: q9876
- Keywords: Nadella bobbleheads
- Bid Amount: $0.5
- Duration: 120

Checking that Field Values are in Right Range or Match Regex

Problem with Previous Approach

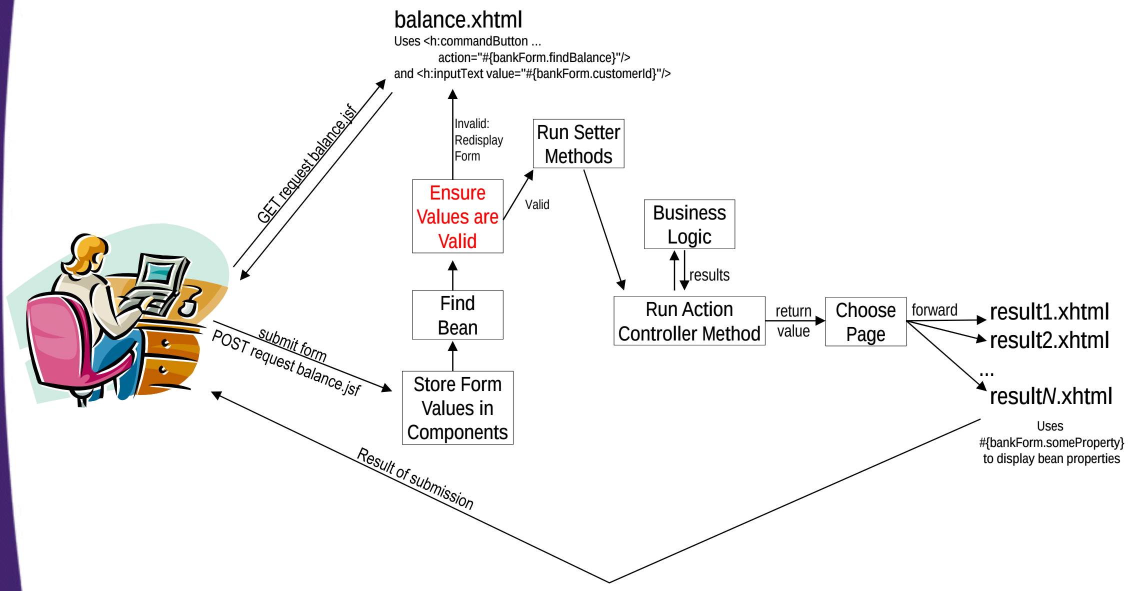
- **Any number was accepted for bid amount and bid duration**
  – Small, zero, and even negative values were accepted. A clever advertiser would enter a negative amount for the bid amount and get paid instead of charged.
- **Solution**
  – Put <f:validateBlah …/> between start and end tags of input element. Add validatorMessage.
- **Example**

```
<h:inputText value="#{bidBean2.bidAmount}"
             required="true" requiredMessage="…" converterMessage="…"
             validatorMessage="Amount must be 0.10 or greater" id="amount">
      <f:validateDoubleRange minimum="0.10"/>
 </h:inputText>
 <h:message for="amount"/>
```

Importance of validatorMessage

- **If validatorMessage supplied**
  - That exact text will be displayed as the error message
  - Easy to localize (internationalize) error messages
    - converterMessage="#{messages.yourErrorName}“
      - See example in second tutorial section on validation
- **If validatorMessage omitted**
  - A standard error message will be displayed
    - Usually cumbersome and hard to read
    - Not customized for your field
  - You can override the error messages from the builtin Messages.properties file by loading a properties file with certain names (e.g., javax.faces.validator.DoubleRangeValidator.NOT_IN_RANGE and many similar ones), but then you get the same error message for all failed conversions of a given type
- **Bottom line**
  - *Always* supply validatorMessage when using f:validate*Blah*

JSF Flow of Control (Simplified)



“Valid” now means non-empty (if “required” is set), able to be converted to the type that the setter method expects, and passing the rules of the explicit validator tags.

Validators and Their Attributes

- **f:validateLength**
  - minimum
  - maximum
- **f:validateLongRange**
  - minimum
  - maximum
- **f:validateDoubleRange**
  - minimum
  - maximum
- **f:validateRegex**
  - pattern
    - Bean property must be String (not double, Integer, etc.)

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

From Randall Munroe and xkcd.com

Example: Search Engine Advertiser Keywords Form

- **Bean code**
  - No changes from previous example
- **Input page**
  - Enforce that user ID is 5 or 6 characters long
  - Enforce that keyword is 3 or more characters long
  - Enforce that bid amount is at least 10 cents
  - Enforce that bid duration is at least 15 days
    - Use converterMessage in each of the four cases
- **Results page**
  - No changes from previous example

Input Form: User ID (enter-bid-4.xhtml)

```
<h:panelGrid columns="3" styleClass="formTable">
  User ID:
  <h:inputText value="#{bidBean2.userId}"
               required="true"
               requiredMessage="You must enter a user ID"
               validatorMessage="ID must be 5 or 6 chars"
               id="userID">
    <f:validateLength minimum="5" maximum="6"/>
  </h:inputText>
  <h:message for="userID"/>
```

No converterMessage because property type is String, so conversion cannot fail.

Input Form: Keyword (enter-bid-4.xhtml)

```
Keywords:
<h:inputText value="#{bidBean2.keyword}"
        required="true"
        requiredMessage="You must enter a keyword"
        validatorMessage="Keyword must be at least 3 chars"
        id="keyword">
  <f:validateLength minimum="3"/>
</h:inputText>
<h:message for="keyword"/>
```

No converterMessage because property type is String, so conversion cannot fail.

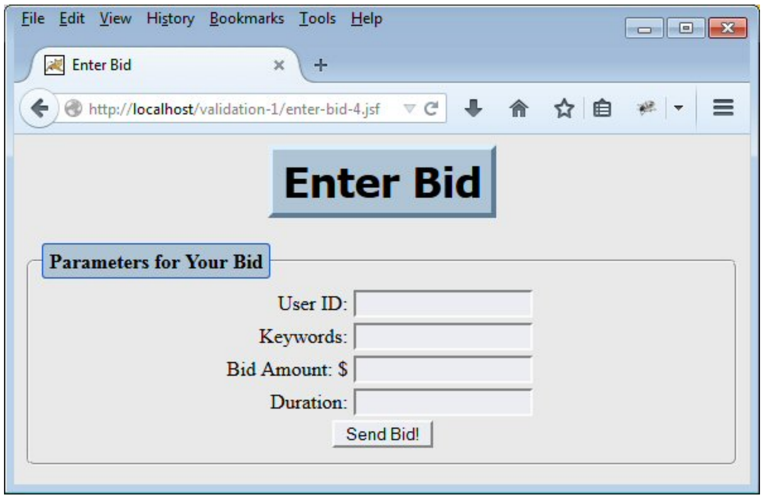
Input Form: Bid Amount (enter-bid-4.xhtml)

```
Bid Amount: $
<h:inputText value="#{bidBean2.bidAmount}"
          required="true"
          requiredMessage="You must enter an amount"
          converterMessage="Amount must be a number"
          validatorMessage="Amount must be 0.10 or greater"
          id="amount">
  <f:validateDoubleRange minimum="0.10"/>
</h:inputText>
<h:message for="amount"/>
```

Input Form: Bid Duration (enter-bid-4.xhtml)

```
  Duration:
  <h:inputText value="#{bidBean2.bidDuration}"
          required="true"
          requiredMessage="You must enter a duration"
          converterMessage="Duration must be a whole number"
          validatorMessage="Duration must be 15 days or more"
          id="duration">
    <f:validateLongRange minimum="15"/>
  </h:inputText>
  <h:message for="duration"/>
</h:panelGrid>

  <h:commandButton value="Send Bid!"
                   action="#{bidBean2.doBid}"/>
</h:form>
```

Results (Initial Form)

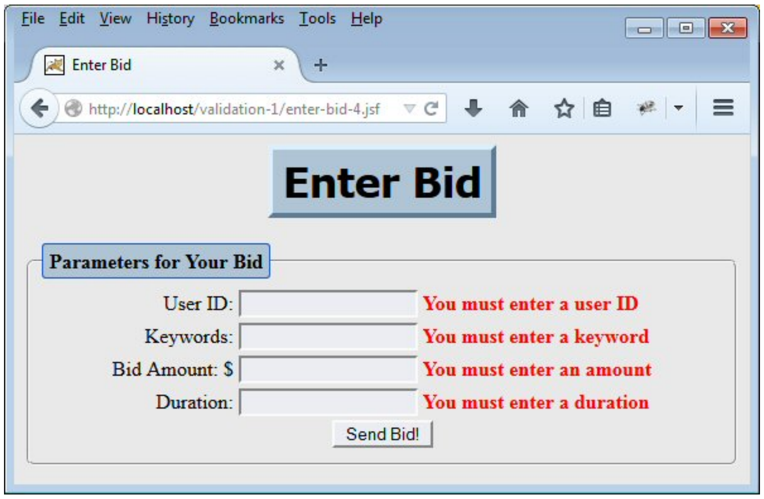

File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-4.jsf

Enter Bid

Parameters for Your Bid

User ID:
Keywords:
Bid Amount: $
Duration:
Send Bid!

Results (Missing Data)


File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-4.jsf

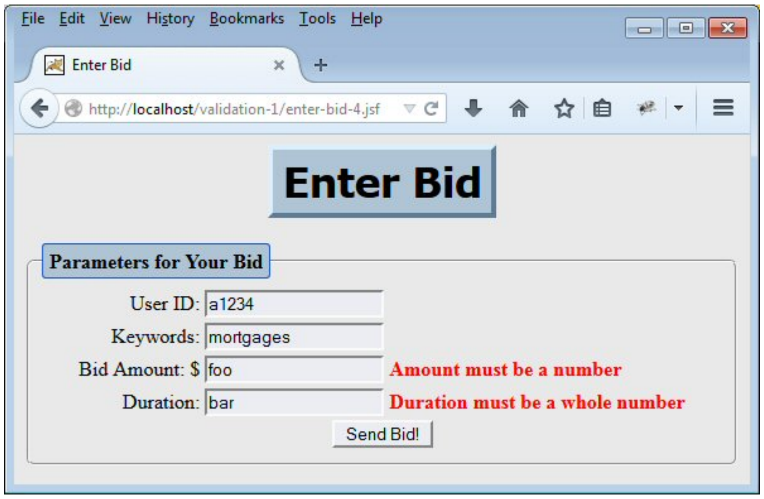
Enter Bid

Parameters for Your Bid

User ID: You must enter a user ID
Keywords: You must enter a keyword
Bid Amount: $ You must enter an amount
Duration: You must enter a duration
Send Bid!

The “required” rule has first precedence.

Results (Type Conversion Errors)


File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-4.jsf

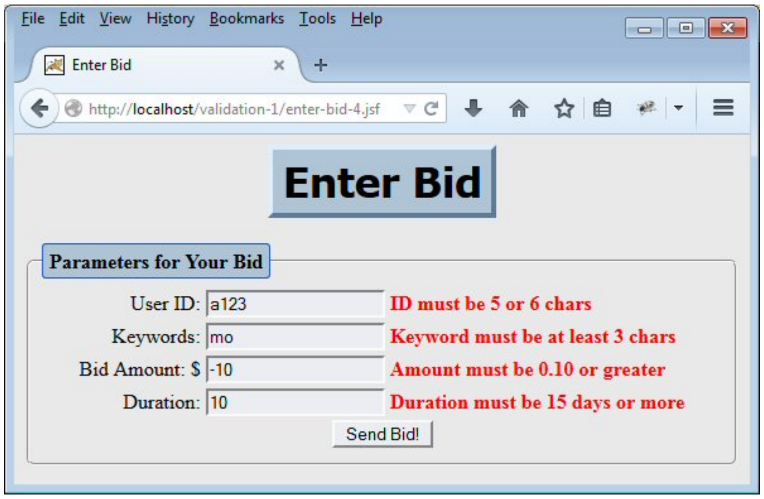
Enter Bid

Parameters for Your Bid

User ID:	a1234	
Keywords:	mortgages	
Bid Amount: $	foo	Amount must be a number
Duration:	bar	Duration must be a whole number

Second precedence is conversion to the types expected by the setter methods.

Results (Range Validation Errors)


File Edit View History Bookmarks Tools Help
Enter Bid
http://localhost/validation-1/enter-bid-4.jsf

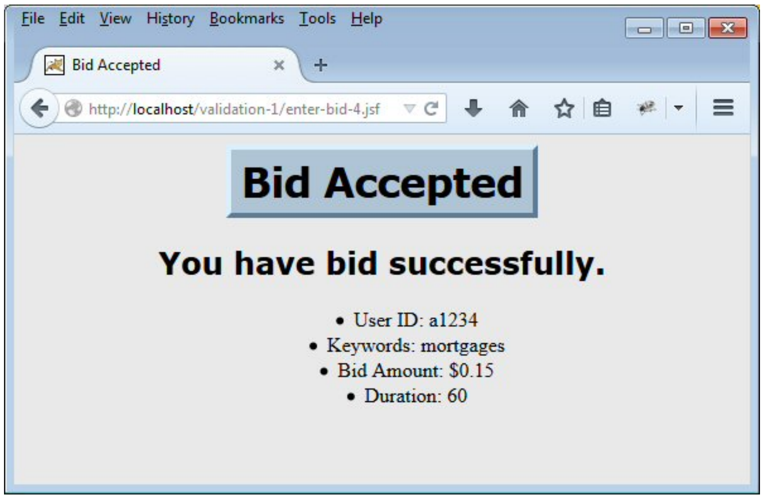
Enter Bid

Parameters for Your Bid

User ID:	a123	ID must be 5 or 6 chars
Keywords:	mo	Keyword must be at least 3 chars
Bid Amount: $	-10	Amount must be 0.10 or greater
Duration:	10	Duration must be 15 days or more

Third precedence is checking the rules of the explicit validator tags.

Results (Good Input)


File Edit View History Bookmarks Tools Help
Bid Accepted
http://localhost/validation-1/enter-bid-4.jsf

Bid Accepted

You have bid successfully.

- User ID: a1234
- Keywords: mortgages
- Bid Amount: $0.15
- Duration: 60

Alerting Users of Problems Below

Problem with Previous Approaches (with Long Forms)

- **Error messages at top**
  - By the time user scrolls down to offending field, the exact error is forgotten
- **Error messages by fields**
  - The user might be looking at top of form, and not be aware that there is a problem further down
- **Solution**
  - Put error messages next to fields
  - Put “Fix Problems Below” warning at top. Do this only if there is at least one error.
  - Implement this with conditional rendering that checks #{not empty facesContext.messageList}

Example: Search Engine Advertiser Keywords Form

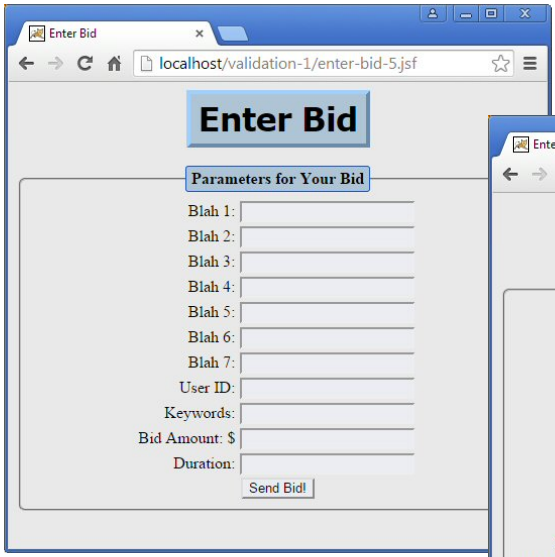
- **Bean code**
  - No changes from previous example
- **Input page**
  - Same per-field checks as previous example
  - Added dummy fields to make the field long
    - The warning is only needed when form is long enough that error message could potentially be scrolled out of view when user looks at top of form
  - Add a “Please Fix Problems Below” warning to top of form that is rendered only if there is at least one error
- **Results page**
  - No changes from previous example

Input Form: Top (enter-bid-5.xhtml)

```
<h:form>
<h:outputText value="Fix Errors Marked Below" styleClass="error"
              rendered="#{not empty facesContext.messageList}"/>
<h:panelGrid columns="3" styleClass="formTable">
  Blah 1: <h:inputText id="blah1"/> <h:message for="blah1"/>
  Blah 2: <h:inputText id="blah2"/> <h:message for="blah2"/>
  Blah 3: <h:inputText id="blah3"/> <h:message for="blah3"/>
  Blah 4: <h:inputText id="blah4"/> <h:message for="blah4"/>
  Blah 5: <h:inputText id="blah5"/> <h:message for="blah5"/>
  Blah 6: <h:inputText id="blah6"/> <h:message for="blah6"/>
  Blah 7: <h:inputText id="blah7"/> <h:message for="blah7"/>

  User ID:
  ...  (same fields and checks as last example)
```

Eclipse (at least as of latest Luna version at end of 2014) incorrectly gives a warning that the not empty test always evaluates to true. Ignore this bogus warning.

Results


Enter Bid

localhost/validation-1/enter-bid-5.jsf

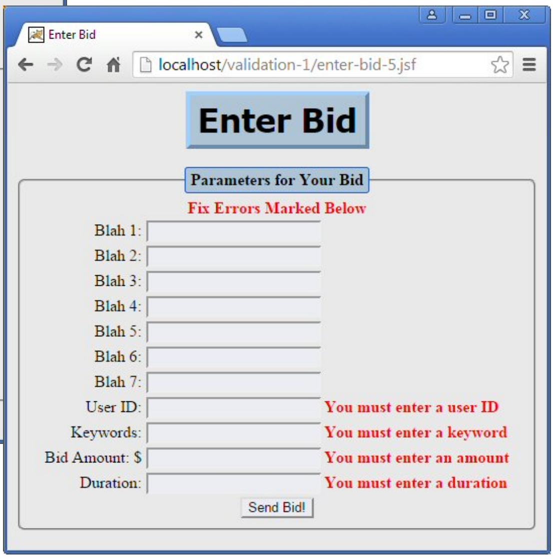
Enter Bid

Parameters for Your Bid

Blah 1:	
Blah 2:	
Blah 3:	
Blah 4:	
Blah 5:	
Blah 6:	
Blah 7:	
User ID:	
Keywords:	
Bid Amount: $	
Duration:	

Send Bid!

No errors on initial display, so warning is not shown


Enter Bid

localhost/validation-1/enter-bid-5.jsf

Enter Bid

Parameters for Your Bid

Fix Errors Marked Below

Blah 1:		
Blah 2:		
Blah 3:		
Blah 4:		
Blah 5:		
Blah 6:		
Blah 7:		
User ID:		You must enter a user ID
Keywords:		You must enter a keyword
Bid Amount: $		You must enter an amount
Duration:		You must enter a duration

Send Bid!

If redisplayed with at least one error, warning is shown

Making Resuable “Fix Errors” Composite Component

- **Composite components**
  - Composite components let you put a chunk of JSF functionality in a separate file and then reuse it
  - There are several later sections on composite components, but very quick preview will be shown here
    - Do not worry about the details of this example: you can get those from the later tutorial section. Just realize that the warning at top could be reused many times, so is prime candidate for being moved into a composite component.

Component File (resources/utils/warnIfError.xhtml)

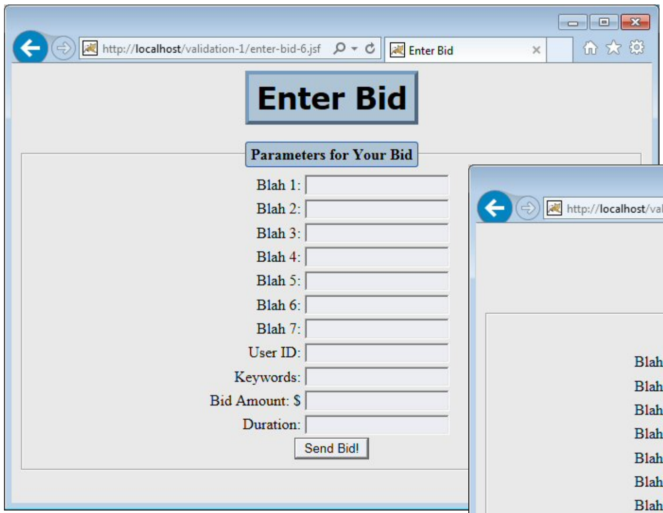
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:cc="http://xmlns.jcp.org/jsf/composite">

<cc:interface>
  <cc:attribute name="message"
                default="Fix Errors Marked Below"/>
  <cc:attribute name="styleClass" default="error"/>
</cc:interface>
<cc:implementation>
<h:outputText value="#{cc.attrs.message}"
              styleClass="#{cc.attrs.styleClass}"
              rendered="#{not empty facesContext.messageList}"/>
</cc:implementation>

</html>
```

This design lets the author of the calling page override the warning message and the CSS class name.
This approach is explained in detail in the first lecture on composite components.

Input Form: Top (enter-bid-6.xhtml)

```
...
<html ...
   xmlns:utils="http://xmlns.jcp.org/jsf/composite/utils">
<h:form>
<utils:warnIfError/>
<h:panelGrid columns="3" styleClass="formTable">
  Blah 1: <h:inputText id="blah1"/> <h:message for="blah1"/>
  Blah 2: <h:inputText id="blah2"/> <h:message for="blah2"/>
  Blah 3: <h:inputText id="blah3"/> <h:message for="blah3"/>
  Blah 4: <h:inputText id="blah4"/> <h:message for="blah4"/>
  Blah 5: <h:inputText id="blah5"/> <h:message for="blah5"/>
  Blah 6: <h:inputText id="blah6"/> <h:message for="blah6"/>
  Blah 7: <h:inputText id="blah7"/> <h:message for="blah7"/>

  User ID:
  ...  (same fields and checks as last example)
```

Results (Same Functionality as Previous Example)



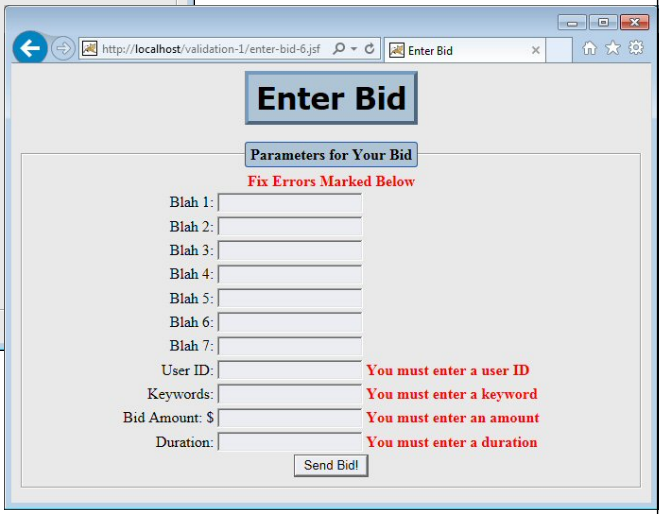
http://localhost/validation-1/enter-bid-6.jsf — Enter Bid

Enter Bid

Parameters for Your Bid

Blah 1:
Blah 2:
Blah 3:
Blah 4:
Blah 5:
Blah 6:
Blah 7:
User ID:
Keywords:
Bid Amount: $
Duration:

No errors on initial display, so warning is not shown



http://localhost/validation-1/enter-bid-6.jsf — Enter Bid

Enter Bid

Parameters for Your Bid

Fix Errors Marked Below

Blah 1:
Blah 2:
Blah 3:
Blah 4:
Blah 5:
Blah 6:
Blah 7:
User ID: You must enter a user ID
Keywords: You must enter a keyword
Bid Amount: $ You must enter an amount
Duration: You must enter a duration

If redisplayed with at least one error, warning is shown

Wrap-Up

Summary

- **Example code**

```
<h:inputText value="..." required="true" requiredMessage="..."
             converterMessage="..." validatorMessage="..." id="someId">
   <f:validateBlah .../>
</h:inputText>
<h:message for="someId"/>
```

- **Strategies**
  - Always supply custom error messages
    - Use requiredMessage when required="true"
    - Use converterMessage with non-String bean properties
    - Use validatorMessage when using f:validate*Blah*
  - Use h:panelGrid to keep error messages next to fields
  - Put “See Below” warning at top if form is long

Questions?

More info:
http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial
http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial
http://courses.coreservlets.com/jsf-training.html – Customized JSF and PrimeFaces training courses
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training