OptionalValidationFramework

Optional Validation Framework

Authors: Mike Kienenberger, Alexander Jesse

An optional validation framework is available as module [OptionalValidator] in svn for the jsf-comp [http://jsf-comp.svn.sourceforge.net/viewvc/jsf-comp/trunk/OptionalValidator/] project.

```
Note: I recommend evaluating the use of the sandbox subForm or ADF subform partial form validation components
as an alternative to this framework -- Mike kienenberger
```

This optional validator framework allows you to change the behavior of validators in a form. JSF allows you limited two-stage control of validation using the immediate attribute. However, this is only sufficient for trivial use cases. It does not work well if you have multiple actions requiring multiple validation configurations on the same form. It also does not provide a warning-only mode where validation errors are reported in a non-fatal manner.

OptionalValidator currently supports three modes: hard (default), soft, and none.

- "none" means that the validator will not be validated.
- "soft" means that the validator will be executed, and the FacesMessages generated, but the lifecycle will continue past the processValidations phase.
- "hard" is the standard JSF behavior, and the validator will perform as normal.

There is also an OptionalConverter for performing the same operations on conversion errors (in none or soft mode, after a conversion error, the converter will always return the original submitted value for getAsString and getAsObject).

Because the required validation of a component is handled separately from normal validation, you must not use the required attribute. Instead, there is a RequiredValidator that needs to be used, and a OptionalValidatorWrappingRequiredValidatorChecker component that is added to bottom of your UIForm in order to trigger the RequiredValidators (which may also be wrapped in OptionalValidators).

There is also a SubmittedValueCollectorWalker component that is added to the top of your UIForm to preserve submitted values for components which failed validation and to mark components invalid at the start of the model-update phase.

The behavior of the optional validators is determined by the setting of the NET_SF_JSFC_OPT_VDTR_MODE request parameter. Different UICommands can submit different request parameter options.

It is the responsibility of the end-user developer to know whether any particular component contains valid input for a given action, but it should now be possible to check the isValid method for a component.

For assistance, send email to mkienenb@gmail.com

Example

```
<html [...] xmlns:jsfcomp="http://sf.net.jsfcomp.validator">
[...]
    <h:form id="form">
        <jsfcomp:submittedValueCollectorWalker/>
        [...]
          <t:inputCalendar value="#{dateValue}">
            <jsfcomp:optionalConverter delegateConverterId="net.sf.jsfcomp.validator.CalendarDateTimeConverter"
/>
            <jsfcomp:optionalValidator id="openDateOV">
                <f:validator validatorId="net.sf.jsfcomp.validator.RequiredValidator"/>
            </jsfcomp:optionalValidator>
        </t:inputCalendar>
        [...]
        <h:inputText value="#{value}">
            <jsfcomp:optionalValidator delegateValidatorId="net.sf.jsfcomp.validator.RequiredValidator"/>
            <jsfcomp:optionalValidator id="openDateOV">
                <myValidatorId/>
            </jsfcomp:optionalValidator>
            </h:inputText>
        [...]
        <h:commandLink value="Optional action"  action="#{optionalAction}">
            <f:param name="NET_SF_JSFC_OPT_VDTR_MODE" value="none"/>
        </h:commandLink>
        <h:commandLink value="Refresh action"  action="#{refreshAction}">
            <f:param name="NET_SF_JSFC_OPT_VDTR_MODE" value="soft"/>
        </h:commandLink>
        <h:commandLink value="Update action"  action="#{refreshAction}">
            <f:param name="NET_SF_JSFC_OPT_VDTR_MODE" value="hard"/>
        </h:commandLink>
        [...]
        <jsfcomp:optionalValidatorWrappingRequiredValidatorChecker/>
    </h:form>
[...]
</html>
```

In progress

- Allow a set of converters to be wrapped by OptionalConverter.
- Allow separate mode settings for different groups of validators.
- Change how OptionalConverter reports errors (use message bundles?).

Implementation notes

- The OptionalValidatorWrappingRequiredValidatorChecker component must be placed after all other validating components in the MyFaces implementation in order to mark those components invalidated. Safest bet is to put it right before the closing </h:form> tag. This is probably also true for other JSF implementations.
- The SubmittedValueCollectorWalker component must be placed before all other validating components in the MyFaces implementation in order to preserve submitted values and update invalidated component status before the update-model phase. Safest bet is to put it right after the <h:form> tag. This is probably also true for other JSF implementations.

Using only the required validator parts of this framework

If all you want is to use a required validator instead of the required attribute on a component, do something like this. The required validator parts of this framework should work with vanilla JSF 1.1, and have been relatively-well tested.

```
<h:form>
     <jsfcomp:submittedValueCollectorWalker/>

     <uiinput>
         <jsfcomp:requiredValidator/>
     <uiinput/>

     <uiinput>
         <jsfcomp:requiredValidator/>
     <uiinput/>

     <jsfcomp:requiredValidatorChecker/>
</h:form>
```