ORACLE | bea

Web Form Tag Library

Version 3.5

WebForm Tag Library

The WebForm tags are used for interacting with cartridge entities deployed in the runtime environment from a JSP. The following table lists categories of tags are supported in the WebForm tag library and the tags within each category.

Tags	Description
WebForm Tags	The complex functionality of WebForms is exposed as tags present in this category. Request handling, validation of submitted data and display of errors can be performed using tags present in this category
webform	This is root controller tag for WebForms. Responsible for validation and request processing. This the root tag under which all other tags given below should be used.
override	Allows you to override choiceList and defaultValue of a field.
display	Displays the webform.
Action Tag	The action tag is equivalent to the HTML <a href=> tag except that the form is submitted back to the same page and the action parameters the you specified are available in the postback request
button	The button tag is equivalent to the HTML <button> tag except that the form is submitted back to the same JSP page and the action parameters that you specified are available in the postback request.
displayerrors	This tag is used to display top-level errors (errors that are not associated with any field) at the top of the web form.
geterrors	This tag can be used to get the list of errors occurred during processing of data.
adderrors	Add business validations errors to the list of error maintained by the web from. These errors, if applicable to a field, would be displayed next to the field.

clearerrors	Clears business validation errors from the list of errors maintained by the webform.
Summary Tags	The tags in this category can be used for displaying web pages. They also provide support for sorting and paging.
table	This tag takes a list of objects and creates a table to display those objects. With the help of column tags, you simply provide the name of properties (get Methods) that are called against the objects in your list that gets displayed [[reword that...]] This tag works very much like the struts iterator tag, most of the attributes have the same name and functionality as the struts tag.
column	The column tag is used to display values from the list specified in the 'table' tag.
setProperty	There are a number of "default" values and strings used by the table tags to show messages, decide which options to display, etc. You can use the <display:setProperty name=... value=...> tag to override these default values.
Core Tags	These tags in this category are non-UI tags, which can be used to interact with the application layer.
processMessage	This tag submits the validated normalized object to the business tier.
property	This tag should be used in conjunction with the processMessage tag. TransformContext properties that can be specified in webforms.xml can be overridden using this tag.
createNo	This tag is used to create a raw normalized object.
persist	This is tag used to persist a normalized object.
query	This tag used to query data from the persistence manager. The result of the query will be available in 'id' attribute value of query tag.
update	This tag is used to update data to the persistence

	manager.
remove	This tag is used to remove data from the persistence manager. This tag removes the normalized object that is passed by the 'message' attribute.
delete	This tag is used delete data from the persistence manager. This tag deletes the result of a query, the name of which is specified for the attribute 'queryName'.
param	This tag is used to pass parameter values for query tag or delete tag. This tag should be used only with in the query tag or delete tag.

Core Tags

Core tags are non-UI tags, which can be used to interact with the application layer.

The core tags that are supported are,

createNo Tag
persist Tag
query Tag
update Tag
remove Tag
delete Tag
param Tag and
processMessage Tag (@deprecated)

See Also:

WebForm Tags
Summary Tags

createNo Tag

This tag is used to create a raw normalized object.
Sample:

```
<% String format="MT304";  %>
```

```
<volante:createNo id=<%= format + "model" %> format=<%= format %>
scope="session" />
```

Attribute	Description
id	This attribute is used for binding the result of the createNo action, which will be a normalized object. In createNo tag if the scope attribute is not mentioned or scope attribute value is page, in this case based on the id attribute value, a corresponding variable will be created in the page context scope and the normalized object will be bound to that variable. Otherwise variable will be created in the 'scope' attribute value context and the normalized object will be bound to that variable. Scope should be page, session, request, and application. This attribute is mandatory.
format	The value of this attribute should be the name of the internal message an instance of which should be created. The internal message should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
scope	Attribute that specifies the scope for the value of the id attribute. Scope should be page, session, request, and application. This attribute is optional.

See Also:

persist Tag
query Tag
update Tag
remove Tag
delete Tag
param Tag
processMessage Tag (@deprecated)

persist Tag

This is tag used to persist a normalized object. The normalized object can be created using ‘createNo’ tag.

Sample:

```
<% String format=”MT304”;  %>
<volante:persist id="result" format=<%= format %> message=<%= no %>/>
```

Attribute	Description
id	This attribute is used for binding the result of the ‘persist’ action, which will be a normalized object. In ‘persist’ tag the scope attribute not mention or scope attribute value is page, in this case based on the id attribute value, a corresponding variable will be created in the page context scope and the normalized object will be bound to that variable. Otherwise variable will be created in the ‘scope’ attribute value context and the normalized object will be bound to that variable. Scope should be page, session, request, and application. This attribute is mandatory.
format	The value of this attribute should be the name of the internal message, to which the normalized object should be persisted. The format should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
message	The normalized object that is to be persisted. The normalized object should be the type of Internal message name specified in the format attribute. This attribute is mandatory.
scope	Attribute that specifies the scope for the value of the id attribute.

	Scope should be page, session, request, and application. This attribute is optional.

See Also:

query Tag
update Tag
remove Tag
delete Tag
param Tag
processMessage Tag (@deprecated)

query Tag

This tag used to query data from the persistence manager. The result of the query will be available in 'id' attribute value of query tag. This id value will contain a 'NormalizedObjectCollection'. If no data matches the query an empty 'NormalizedObjectCollection' will be returned.

Sample:

```
<volante:query queryName="All" id="summaryList" format=<%=format%>
scope="session"/>
```

Using query tag with parameter:

```
<volante:query queryName="ByItem" id="result" format="Invoice" scope="page">
    <volante:param value="ITM1"/>
</volante:query>
```

Attribute	Description
id	This attribute is used for binding the result of the 'query' action, which will be a NormalizedObjectCollection. In 'query' tag if the scope attribute is not mentioned or scope attribute value is page, in this case based on the id attribute value, a corresponding variable will be created in the page context scope and the NormalizedObjectCollection will be bound to that variable. Otherwise variable will be created in the 'scope' attribute value context and the NormalizedObjectCollection will be bound to

	that variable. Scope should be page, session, request, and application. This attribute is mandatory.
queryName	The name of the query that should be already defined in the persistence manager. 'All' is the default query name, which is already defined in the persistence manager. This attribute is mandatory.
format	The value of this attribute should be the name of the internal message, where the query should be looked up. The format should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
scope	Attribute that specifies the scope for the value of the id attribute. Scope should be page, session, request, and application. This attribute is optional.
startRow	Not yet implemented
maxRows	Not yet implemented

See Also:

createNo Tag
persist Tag
update Tag
remove Tag
delete Tag
param Tag
processMessage Tag (@deprecated)

update Tag

This tag is used to update data to the persistence manager.

Sample:

```
<% String format=”MT304”; %>
<volante:update  id="result" format=<%= format %> message=<%= no %>/>
```

Attribute	Description
id	This attribute is used for binding the result of the ‘update’ action, which will be a string message. In ‘update’ tag if the scope attribute is not mentioned or scope attribute value is page, based on the id attribute value, a corresponding variable will be created in the page context scope and the string message will be bound to that variable. Otherwise variable will be created in the ‘scope’ attribute value context and the string message will be bound to that variable. Scope should be page, session, request, and application. This attribute is optional.
format	The value of this attribute should be the name of the internal message, where the normalized object should be updated. The format should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
message	The value of this attribute should be the normalized object to be updated. This attribute is mandatory.
scope	Attribute that specifies the scope for the value of the id attribute. Scope should be page, session, request, and application. This attribute is optional.

See Also:

createNo Tag
persist Tag
query Tag
remove Tag
delete Tag
param Tag
processMessage Tag (@deprecated)

remove Tag

This tag is used to remove data from the persistence manager. This tag removes the normalized object that is passed by the 'message' attribute.

Sample:

```
<% String format="MT304"; %>
<volante:remove message=<%= no %> format= <%= format %> id="result" />
```

Attribute	Description
id	This attribute is used for binding the result of the 'remove' action, which will be a string message. In 'remove' tag if the scope attribute is not mentioned or scope attribute value is page, based on the id attribute value, a corresponding variable will be created in the page context scope and the string message will be bound to that variable. Otherwise variable will be created in the 'scope' attribute value context and the string message will be bound to that variable. Scope should be page, session, request, and application. This attribute is optional.
format	The name of the internal message, from where the data should be removed. The internal message should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
message	The value of this attribute should be the normalized object to be removed.

	This attribute is mandatory.
scope	Attribute that specifies the scope for the value of the ‘id’ attribute. Scope should be page, session, request, and application. This attribute is optional.

See Also:

createNo Tag
persist Tag
query Tag
update Tag
delete Tag
param Tag
processMessage Tag (@deprecated)

delete Tag

This tag is used delete data from the persistence manager.
This tag deletes the result of a query, the name of which is specified for the attribute ‘queryName’.

Sample:

```
<volante:delete queryName="ByItem" id="result" format="Invoice">
      <volante:param name="Id" value="ITM7"/>
</volante:delete>
```

Attribute	Description
id	This attribute is used for binding the result of the ‘delete’ action, which will be a string message. In ‘delete’ tag if the scope attribute not mentioned or scope attribute value is page, based on the id attribute value, a corresponding variable will be created in the page context scope and the string message will be bound to that variable. Otherwise variable will be created in the ‘scope’ attribute value context and the string message will be bound to that variable.

	Scope should be page, session, request, and application. This attribute is not mandatory.
format	The value of this attribute should be the name of the internal message, from where the result of the query is to be deleted. The format should be present. Exception will be thrown if incorrect format is specified. This attribute is mandatory.
queryName	The name of the query that should be already defined in the persistence manager. 'All' is the default query name, which is already defined in the persistence manager. This attribute is mandatory. The result of the query will be removed when this tag is executed.
scope	Attribute that specifies the scope for the value of the id attribute. Scope should be page, session, request, and application. This attribute is optional.

See Also:

createNo Tag
persist Tag
query Tag
update Tag
remove Tag
param Tag
processMessage Tag (@deprecated)

param Tag

This tag is used to pass parameter values for query tag, delete tag. This tag should be used only with in these tags.

Sample:

Using param tag with query Tag:

```
<volante:query queryName="ByItem" id="result" format="Invoice" scope="page">
    <volante:param value="ITM1"/>
</volante:query>
```

Using param tag with delete Tag:

```
<volante:delete queryName="ByItem" id="result" format="Invoice">
      <volante:param name="Id" value="ITM7"/>
</volante:delete>
```

Using param tag within input and output tags:

```
<volante:invokeMessageFlow name='<%= name + "WebFlow" %>'
errorRef="processingErrors">
    <volante:input>
        <volante:param value="<%= rawMessage.getBytes() %>"/>
    </volante:input>
    <volante:output>
        <volante:param valueRef="noObj"/>
    </volante:output>
</volante:invokeMessageFlow>
```

Attribute	**Description**
name	The value of this attribute should be the name of parameter that is already defined in the query. If this tag is used within ‘query’, ‘input’ or ‘output’ tags this attribute need not be specified. When used within ‘delete’ tag this attribute must be specified.
value	The value of this attribute should be the value of parameter that is already defined in the query. This attribute is optional. While using the ‘input’ tag either ‘value’ or ‘valueRef’ can be specified. But both attributes cannot be specified.
valueRef	This attribute refers to a variable that contains the value for the parameter. While using the ‘output’ tag it is advisable to use this attribute instead of the ‘value’ attribute.

	This attribute is optional.
scope	Attribute that specifies the scope for the valueRef (session, request, application, page). This attribute is optional. By default the scope is 'Page' scope.

See Also:

createNo Tag
persist Tag
query Tag
update Tag
remove Tag
delete Tag
processMessage Tag (@deprecated)

processMessage Tag (@deprecated)

The Webform tag by itself does not submit the validated normalized object to the business tier. You can perform this by using the processMessage tag. This tag is technically a core tag and can be used independent of the WebForm tag.

This tag submits the specified normalized object to the Business transaction EJB for further processing. The Transform context properties (output.format, output.protocol etc) can be specified in webforms.xml config file. Alternatively, you can specify them as a set of property tags nest under the process tag. The property values you specify, as property tag will override the values specified in the XML.

If there are any exceptions while processing, they are bound to a page attribute with the name specified in errorRef.

Sample:

```
<volante:processMessage errorRef="processingErrors"
format=<%= format %>
message=<%= session.getAttribute(modelName) %> >
 <% if(!"saveAndRelease".equals(request.getParameter("task"))) { %>
       <volante:property name="output.protocol" value="nullprotocol" />
 <% } %>
</volante:processMessage>
```

See Also:

createNo Tag

persist Tag
query Tag
update Tag
remove Tag
delete Tag
param Tag

property Tag (@deprecated)

The property is used must be present inside a processMessage tag. The processMessage tag submits the specified normalized object to the Business transaction EJB for further processing. The Transform context properties (internal.format etc) can be specified in webforms.xml config file. Alternatively, you can specify them as a set of property tags nest under the process tag. The property values you specify, as property tag will override the values specified in the XML.

Sample:

```
<volante:processMessage errorRef="processingErrors"
format=<%= format %>
message=<%= session.getAttribute(modelName) %> >
 <% if(!"saveAndRelease".equals(request.getParameter("task"))) { %>
        <volante:property name="output.protocol" value="nullprotocol" />
 <% } %>
</volante:processMessage>
```

See Also:

createNo Tag
persist Tag
query Tag
update Tag
remove Tag
delete Tag
param Tag

WebForm Tags

Since Web form encapsulates complex functionality it is exposed in the tag library as number of cooperating nested tags. The *webform* tag is main controller tag for Web forms.

Tags	Functionality

webform	This is root controller tag for WebForms. Responsible for validation and request processing. This the root tag under which all other tags given below should be used.
override	Allows you to override some attributes of a field (displayed in the form).
display	Displays the webform.
displayerrors	Displays message level errors in the form.
action	The action tag is equivalent to the HTML <a href=> tag except that the form is submitted back to the same JSP page and the action parameters that you specified are available in the postback request.
button	The button tag is equivalent to the HTML <button> tag except that the form is submitted back to the same JSP page and the action parameters that you specified are available in the postback request.
adderrors	Add business validations errors to the list of error maintained by the web from. These errors, if applicable to a field, would be displayed next to the field.
clearerrors	Clears business validation errors from the list of errors maintained by the webform.
geterrors	Gets the list of errors that have occurred.

See Also:

A sample usage of Webform tag
Core Tags
Summary Tags

Web Form Tag

The webform tag is main controller tag for Web forms. It performs the following functions.

Handles request (submitted data). It converts back and forth between the presentation model and the actual data model (Normalized Object). In case of errors it populates the form with data entered by the user.

Validates the submitted form. These validations include mandatory fields verification and type conversion checks (UI to model). Application level validations must be done in the EJB layer.

Exposes some variables that help you decide whether the form is completed and the model (NormalizedObect) is ready for use (processing in EJB layer)

In case of business validation errors (in EJB layer), you can pass these errors back to the WebForm, which would display the errors next to the correct field.

Allow you to override the choice list, the default value and other attributes of a field. Using this you can for instance fill up the sender and recipient combo boxes with values fetched from the Database.

See Also:

Attributes of webform Tag
Variables exposed by webform Tag
Overriding values specified in Designer
Navigation Tags
Displaying the form
Displaying top level error messages
Adding Business validation errors to the form
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Attributes of webform Tag

Attribute	Description
name	The name of the webform, which is to be displayed. The name should be defined in the cartridge. This attribute is optional.

	If this attribute is not specified then webform with name ‘Default’ will be looked up and displayed.
format	The internal format’s name as specified in the Designer. The internal message should have a web form added and the Jars files for the internal message should have been deployed. This attribute is mandatory.
modelRef	The value of this attribute (in the specified scope) should hold the normalized object. You can use this or use the model property to specify the NormalizedObject. This attribute is optional.
model	The NormalizedObject to be used. You can use either the model or the modelRef property. This attribute is optional.
scope	Attribute that specifies the scope for the modelRef (session, request, application, page). This attribute is optional.
requestURI	The URL for the page where the WebForm tag is present should be specified as value for this property. This information is needed because WebForm submits back (postback) to the same page. By default, the web tag tries to figure out what the URL is for the page it is on by calling the request.getRequestURI() method, but this will not always return a correct answer in environment where you are forwarding a request around before arriving at the JSP that is to be displayed (like struts). In those cases, you need to specify the URL via the "requestURI" attribute. This attribute is optional.
errorStyle	The style in error messages should be displayed. It can be one of “text”, “tooltip” or “tooltippopup”. In the “text” is specified, the validation messages are displayed as plain text in the form. If you specify “tooltip” an error icon is displayed next to the field with the error message as the tooltip (mouse over image).If “tooltippopup” is selected an error icon will be displayed next to the error field. If the icon is clicked the error message will be

	displayed in a pop up window. Even in this case the error message will also be displayed as tool tip when mouse is moved over the image. This attribute is optional.
errorBackground	This attribute specifies the background in which the error fields are highlighted. By default the error fields are highlighted in a background of pink. HTML color syntax can be used to specify the color (e.g. #FFFFFF) This attribute is optional.
checkMandatoryFields	By default the webform validates the presence of mandatory fields before sending data to the EJB/Application layer. However since the validation is anyway likely to be performed in the EJB/Application layer the value of this attribute can be set to false. If this attribute is set to false it will not perform the validation. This provides a flexible approach when you want to save a partially filled page. For example if you want to save the data in draft mode this attribute has to be set to false. By default the value of this attribute is true. This attribute is optional.
readOnly	Specifies whether the message should be displayed in edit or read only mode. If it is displayed in read only mode all the fields are disabled. The allowed values (boolean) are true and false. This attribute is optional.
hideNullFields	If set to true, all fields and sections with null value are hidden. It provides a convenient way of viewing sparsely populated form. This should normally be used only in readOnly mode. This attribute is optional.
page	Sets the current page for the webform. This attribute is optional.

```
<volante:webform  format=<%= format %>
  errorStyle="text"
```

```
  errorBackground="#FFE3E7"
  modelRef=<%= modelName %>
  requestURI=<%= requestURI%>
  readOnly=<%= readOnly %>
>
```

See Also:

Variables exposed by webform Tag

Variables exposed by webform Tag

Variable Name	Description
formpage	The current page number (1 .. formPages)
formpages	Total number of pages in the form
completed	A boolean value, if true means that the form has been submitted and there are no validation errors. This would mean that the normalized object is ready to be passed to the business tier (EJB).
errors	True if the form to be displayed has errors
pagenames	A String[] containing the names of all pages in the form (as specified during design time).
currentpage	Name of the current page to be displayed.

These page level variables are available inside the webform tag. These variables provide enough information for you to build navigational buttons (Previous/Next) or a tab like interface in your JSP page.

Note:

You can add HTML tags, Java code and nested tags inside the webform tag. They behave, as you would expect.

See Also:

Attributes of webform Tag

Overriding values specified in Designer

You can override the attributes of a field specified in webforms in Designer by using the <volante:override> tag. For example, using the 'choiceList' attribute of the <volante:override> tag, you can override the list of values to be displayed for a field. Using this you can, for instance, fill up the sender and recipient combo boxes with values fetched from the Database. Please note that, in this you should have specified the renderer for the field as Choice in the Designer for choiceList override to work.
The following code fragments overrides the values specified for field 'Sender' with choice renderer in Designer. The values displayed in the combo in webforms are the list of values specified using the override tag and not the values specified in Designer.

```
<%
List senders = Arrays.asList( new String[] {"DAVID LEAN","WILLIAM PRICE"});
List recipients =Arrays.asList( new String[] {"BANK OF NEW YORK",
"ABN AMRO"});
%>
<volante:webform  format=<%= format %>   ….. >
      ...
<volante:override fieldName="Sender" choiceList="<%= senders%>"
label="New Sender" labelStyle="DefaultLabelStyle"
rendererStyle="DefaultRendererStyle" />
      ...
<volante:override fieldName="Sender" choiceList="<%= senders%>"
visible="false" />
...
```

Attribute	Description
fieldName	Qualified name (as specified in Designer) of a field whose attributes you want to override. This is a mandatory attribute.
choiceList	List of string values. You should have specified the renderer for the field as *Choice* in the Designer for the choiceList override to work. The specified values replace the values specified in Designer (typically you would have left it empty in the Designer). This is an optional attribute.
defaultValue	The default value for the field. This overrides the value you specified in the Designer.

	This is an optional attribute.
label	The title to be used for a field. This overrides the value you specified in the Designer. This is an optional attribute.
enabled	The enabled or disabled state of a field in a form. Use the value “false” to disable a field and the value “true” to enable a field. This is an optional attribute.
visible	The visible or hidden state a field in a form. Use the value “false” to hide a field and the value “true” to show a field. This is an optional attribute.
labelStyle	The style to be applied to the title/label of a field. It should be one of the styles available under the Style Manager of the corresponding Web Form. This is an optional attribute.
rendererStyle	The style to be applied to the renderer of a field. It should be one of the styles available under the Style Manager of the corresponding Web Form. This is an optional attribute.

See Also:

Web Form Tag
Navigation Tags
Displaying the form
Displaying top level error messages
Adding Business validation errors to the form
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Navigation Tags

By default, (for maximum flexibility) the webform does not display navigational buttons (for submitting, moving to next page etc). This has to be done in the JSP by the programmer. Webforms provides two tags for navigation action tag and button tag.

See Also:

Action Tag
Button tag
Web Form Tag
Overriding values specified in Designer
Displaying the form
Displaying top level error messages
Adding Business validation errors to the form
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Action Tag

The action tag is equivalent to the HTML <a href=> tag except that the form is submitted back to the same page and the action parameters the you specified are available in the postback request. The actions supported are “Submit”, “nextPage” and “PreviousPage”. Any additional parameters you pass will be available in the post back request as parameters.

```
<volante:action name="Submit" params="task=save">
      <IMG src="images/save.gif" border="0"/>
</volante:action>
```

These are the attributes supported by the *action* tag.

Attribute	Description
name	The webform action that you want to perform. The allowed values are “Submit”, “NextPage” and “PreviousPage”. This is a mandatory attribute.
params	Your action specific parameters (name value pairs). These values would be appended to the link and the values would be accessible when the form is posted back. This is an optional attribute.
style	The CSS class that should be applied to the button This is an optional attribute.

See Also:

Button tag

Button tag

Instead of using action tag the other possibility is to add buttons to the form. This can be done using the button tag.

This tag is equivalent to the HTML <button> tag except that the form is submitted back to the same JSP page and the action parameters that you specified are available in the postback request.

```
<volante:button name="Submit" style="sbttn" label=" Save "
               params="task=save"/>
```

These are the attributes supported by the *button* tag.

Attribute	Description
name	The webform action that you want to perform. The allowed values are “Submit”, “NextPage” and “PreviousPage”. This is a mandatory attribute.
params	Your action specific parameters (name value pairs). These values would be appended to the link and the values would be

	accessible when the form is posted back. This is an optional attribute.
label	Label of the button This is a mandatory attribute.
style	The CSS class that should be applied to the button This is an optional attribute.

The reason for providing these two tags is to ensure that form is posted back as per the requirement. If the control should come back to the form, always use one of these two tags instead of their HTM equivalents.

See Also:

Action Tag

Displaying the form

Display part is exposed as an independent tag because the user may not always want to display the form. Note that WebForm uses a postback model hence when the form is submitted the same page gets control. The WebForm tag validates the submitted data and if there were no errors it would send it to the Business Tier (EJB). And if no error occurs in business tier also, the form would not be displayed again. The user would forward to some other page. So it is very likely that the user would use the 'display' tag conditionally.

```
<span width="100%" class="rectangle-border">
<volante:display/>
</span>
```

See Also:

Web Form Tag
Overriding values specified in Designer
Navigation Tags
Displaying top level error messages
Adding Business validation errors to the form
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Displaying top level error messages

Top level (message level) error messages that are not associated with any field can be displayed at the top of the webform using the 'displayerrors' tag.

```
<span width="100%" class="rectangle-border">
<volante:displayerrors/>
</span>
```

See Also:

Web Form Tag
Overriding values specified in Designer
Navigation Tags
Displaying the form
Adding Business validation errors to the form
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Adding Business validation errors to the form

The validations done by the Webform are only mandatory field verification and type conversion checks. Rest of the validations has to be done in the Application layer while processing the message. If errors are encountered during processing, they have to be displayed in the Web form (next to the field). You do this by adding the business validation errors to the web form.

```
<volante:adderrors errorRef="processingErrors" scope="session"/>
```

The errorRef should refer to an attribute that has a list of validation errors

Attribute	Description
errorRef	Reference to a list of errors.
error	List of errors.
scope	Attribute that specifies the scope for the errorRef (session, request, application, page). This attribute is optional.

	By default the scope is ‘Page’ scope.

See Also:

Web Form Tag
Overriding values specified in Designer
Navigation Tags
Displaying the form
Displaying top level error messages
Removing Business validation errors
Getting List of Errors
A sample usage of Webform tag

Removing Business validation errors

The list of business validation stored in a session variable (or in other scope) can be removed using the clear errors tag.

```
<volante:clearerrors errorRef="externalErrors"/>
```

The errorRef should refer to an attribute that has a list of validation errors. All the errors present in the list will be cleared.

Attribute	Description
errorRef	Reference to a list of errors to be cleared. If used in conjunction with the invokeMessage tag the errorRef you specify here would be same as the errorRef you specified in invokeMessage tag.
scope	Attribute that specifies the scope for the errorRef (session, request, application, page). By default the scope is ‘Page’ scope.

See Also:

Web Form Tag
Overriding values specified in Designer
Navigation Tags
Displaying the form
Displaying top level error messages
Adding Business validation errors to the form
Getting List of Errors
A sample usage of Webform tag

Getting List of Errors

The list of errors occurred during processing of data can be obtained using the 'geterrors' tag. This tag returns a list of **com.tplus.transform.runtime.webforms.ValidationInfo** objects. Details about each error such as ErrorCode, Message etc. can be obtained from the ValidationInfo object.

88

```
<volante:geterrors errorType="message" errorRef="topLevelErrors"
scope="session"/>
```

Attribute	Description
ErrorType	This attributes specifies the type of error that is to be returned. It can be one of two values 'all' or 'message'. If value specified is 'all' all errors that occur during processing are returned. If value specified is 'message' only top level errors (errors that are not bound to any field) are returned. By default the errorType is 'message'.
ErrorRef	Reference to the list of errors that are returned by the geterrors tag. The list contains ValidationInfo objects. Details about the error can be obtained from ValidationInfo object.
Scope	Attribute that specifies the scope for the errorRef (session, request, application, page). By default the scope is 'Page' scope.

Methods in **com.tplus.transform.runtime.webforms.ValidationInfo**
The ValidationInfo object obtained using the geterrors tag can be used to get details about the error using the following methods present in it.

Method	Description
getFieldId()	Returns the id of the field with which this exception is associated. If the exception is not associated with any field 'null' is returned.
GetMessage()	Returns the error message.

getExceptionObject()	Returns the original exception thrown from the application layer or null if this validation was generated by web form itself. If the exception is not null exception properties such as ErrorCode, Error Phase etc. can be obtained from it.
getFieldValue()	Returns the value for the error field. In cases where the value cannot be set for the field (e.g. type mismatch) null will be returned.

See Also:

Web Form Tag
Overriding values specified in Designer
Navigation Tags
Displaying the form
Displaying top level error messages
Adding Business validation errors to the form
Removing Business validation errors
A sample usage of Webform tag

A sample usage of Webform tags

```
<volante:webform name="<%= webformName %>" format="<%= format %>"
checkMandatoryFields="false" errorStyle="tooltippopup" modelRef="<%=
modelName%>" requestURI="<%= requestURI%>"
readOnly="<%= readOnly%>" hideNullFields="<%= readOnly%>">
   <!—Override values specified in designer --!>
   <% if(newMessage) { %>
      <volante:override fieldName="Sender" label="New Sender"/>
  <% } else { %>
      <volante:override fieldName="Sender" enabled="false" />
  <% } %>
  <volante:override fieldName="Recipient" />
  <!— Check whether form has been completed (with no errors) --!>
  <% if (!completed.booleanValue()) { %>
     <volante:adderrors errorRef="externalErrors" scope="session"/>
  <% }
  <volante:clearerrors errorRef="externalErrors" scope="session"/>
  <volante:clearerrors errorRef="processingErrors" scope="session"/>
   <!— Send message for processing to EJB layer--!>
```

```
   <volante:invokeMessageFlow name='<%= format + "WebFlow" %>'
errorRef="processingErrors" scope="session">
    <volante:input>
         <volante:param value="<%= session.getAttribute(modelName)%>"/>
         <volante:param value='<%= request.getParameter("task") %>'/>
     </volante:input>
   </volante:invokeMessageFlow>
   <% if (session.getAttribute("processingErrors") == null) { %>
      <jsp:forward page="<%= returnPage%>"/>
   <% }%>
   <% } %>
   <!— Add business validation errors --!>
   <volante:adderrors errorRef="processingErrors" scope="session"/>
   <!--- Using table for laying out buttons, tabs, error messages & form --->
   <table width="760" cellspacing="0" cellpadding="0" >
   <!--- display button at top --->
       <tr><td>
       <%@ include file="WebFormButtons.jsp" %>
       </td></tr>
       <!--- display top level errors --->
       <tr><td>
          <volante:displayerrors/>
       </td></tr>
       <!--- display a row of tabs--->
        <tr><td>
        <br>
        <%@ include file="WebFormTabs.jsp" %>
         </td></tr>
         <!--- display the  form --->
         <tr><td style="BORDER:#000000 1px solid;">
         <volante:display/>
         </td></tr>
         <!--- display button at bottom --->
          <tr><td>
            <br>
              <%@ include file="WebFormButtons.jsp" %>
          </td></tr>
        </table>
    </volante:webform>
```

See Also:

WebForm Tags

Summary Tags

Summary tags are based on an open source tag library named “display”.

Main Features:

Displays a list of Normalized objects (in fact it can a collection of any bean)
Support for sorting a column
Support for paging
Allows customization of columns (links, form fields etc).
Based on the HTML table tag.
Works well with the core tags such as *query* etc. that produce a list of normalized objects.

See Also:

Table Tag
Column Tag
setProperty Tag
Core Tags
WebForm Tags

Table Tag

This tag takes a list of objects and creates a table to display those objects. With the help of column tags, you simply provide the name of properties (get Methods) that are called against the objects in your list that gets displayed. This tag works very much like the struts iterator tag, most of the attributes have the same name and functionality as the struts tag.

Attribute	Description
Id	Uniquely identifies a table tag. This is useful in cases where multiple table tags are defined in a single page.
name	The value of this attribute should be the list of objects that are to be displayed in the table.
property	The property to be fetched from the list specified in the ‘name’ attribute should be given here.
list	The list of objects to be displayed can be directly passed to the table using this attribute. In such a case ‘name’ attribute need

	not be used.
decorator	A "decorator" is a design pattern where one object provides a layer of functionality by wrapping or "decorating" another object. You can write your own wrapper class extending the decorator and specify that class as value for this attribute.
scope	Attribute that specifies the scope for the values of this table
length	In case where you want to show only a subset of the data present in the list you can use the length attribute. The number of data display will be the value specified for this attribute.
offset	This attribute can be used to skip data present in the list. For example you can display first 2 items in the list, then skip the next 5 items using the offset attribute. This attribute can be used together with the length attribute.
pagesize	This attribute can be used to split the data in the list into pages. The number of items displayed in the table will be the value specified for this property. If the number of items in the list is greater than the value specified the remaining items will be automatically paged.
requestURI	The URL for the page where the table tag is present should be specified as value for this property. By default, the table tag tries to figure out what the URL is for the page it is on by calling the request.getRequestURI() method, but this will not always return a correct answer in environment where you are forwarding a request around before arriving at the JSP that is to be displayed (like struts). In those cases, you need to tell the table tag what it's URL is via the "requestURI" attribute
width	Width of the table.
styleClass	Style class name to apply to the table
border	Width of border to be drawn around cells
cellspacing	Amount of space between cells
cellpadding	Amount of space between cell border and contents
align	Aligns within the text flow – You can use stylesheets instead

background	Background image
bgcolor	Background color of table – You can use stylesheets instead
frame	To draw borders around the table (IE only)
height	Specifies height of the table
hspace	Number of pixels to left/right of an aligned table
Rules	To draw borders within the table (IE only)
summary	Like alt, provides a summary of the table for non-display browsers
vspace	Number of pixels above/below an aligned table

An example for table tag is given below

```
<volante:table id="1" name="summaryList"
requestURI=<%= requestURI %>
pagesize="15"decorator="com.tplus.transform.runtime.webforms.summary.CustomData
ObjectWrapper" border="0" bgcolor="#CCCCCC">
```

See Also:

Column Tag
setProperty Tag

Column Tag

The column tag is used to display values from the list specified in the ‘table’ tag.

Attribute	Description
Property	This attribute specifies what getXXX method is called on each item in the list. The value returned by the method is set for the column.
Value	The value for the column can be directly specified using this property instead of using ‘property’.
Title	The title for the column.

Nulls	This attribute can be used suppress "null" values that might be returned.
decorator	A "decorator" is a design pattern where one object provides a layer of functionality by wrapping or "decorating" another object. You can write your own wrapper class extending the decorator and specify that class as value for this attribute.
Sort	If this attribute is “true” then the column data will be sorted. When the user clicks on the column title the rows will be sorted in ascending order and redisplayed on the page. If the user clicks on the column title again, the data will get sorted in descending order and redisplayed. Only the rows being shown on the page are sorted and resorted, so if you use this attribute along with the pagesize attribute, it will not resort the entire list.
Group	This attribute is used to group columns. If multiple columns are group the grouping order should be specified in this property.
Autolink	If you have email addresses or web URLs in the data that you are displaying in columns of your table, then you can set the autolink="true". The data will be automatically displayed as hyperlinks.
Href	This is strut-like attribute that can be set to create a dynamic link. It is the base URL used to construct the dynamic link.
paramId	The name of the parameter that gets added to the URL specified in the href property.
paramName	Name of the bean that contains the data the user wants to tack on the URL (typicall null, indicating the current object in the List)
paramProperty	Attribute to call on the object specified in the paramName property to return the value that gets tacked onto the URL.
paramScope	Specific scope where the databean lives, typically null

maxLength	This attribute can be used to restrict the size of a column, so that it fits within a certain size of table.
maxWords	This attribute can be used to restrict the number of words to be displayed in a column. "..." will be appended at the end if value of the column is greater than the value specified for this property.
Width	Width of the cell
styleClass	Style class name to apply to the cell
Align	Aligns within the cell
Valign	Specifies vertical alignment within the cell
background	Background image
Bgcolor	Background color of cell – you can use stylesheets instead
Height	Specifies height of the cell
nowrap	Indicates that the cell should not wrap text

An example for 'column' tag is given below

```
<volante:column property=<%=fieldName%> title=<%=fieldName%> sort="true"
decorator="com.tplus.transform.runtime.webforms.summary.LinkColumnWrapper"  />
```

See Also:

Table Tag
setProperty Tag

setProperty Tag

There are a number of "default" values and strings used by the table tags to show messages, decide which options to display, etc. You can use the <display:setProperty name=... value=...> tag to override these default values. This is useful if you want to change the behavior of the tag a little (for example, don't show the header etc), or if you need to localize some of the default messages and banners.

Attribute	Description
name	The name of the property to override.
value	Value for the property.

An example for ‘setProperty’ tag is given below

```
<volante:setProperty name="sort.behavior" value="all" />
```

The various properties that are currently available are listed along with their default values and a brief explanation.

Property	Default	Valid Values	Descriptions
basic.show.header	true	true, false	Indicates if you want the header to appear at the top of the table, the header contains the column names, and any additional action banners that might be required (like paging, export, etc...)
basic.msg.empty_list	Nothing found to display	Any string	The message that is displayed with the list that this table is associated with is either null, or empty.
sort.behavior	page	page, list	Describes the behavior that happens when a user clicks on a sortable column in a table that is showing just a portion of a long list. The default behavior (page), just resorts the elements currently being displayed and leaves the user showing the same subset. If you change this value to (list) then the entire list will be resorted, and the user will be taken back to the first subset of the newly sorted list.
export.banner	Export options: {0}	Any string in a message format with 1 placehold	Contains the string that is displayed in the table footer when the user indicates that they want to enabled the export function. The placeholder is replaced with links to the various export formats

		er	that are support.
export.sepchar	|	Any string	Used to seperate the valid export type (typically would be a bar a comma, or a dash).
export.csv	true	true, false	Should the tag present the option to export data in comma seperated format (csv).
export.csv.label	CSV	Any string	The label on the link that the user clicks on to export the data in CSV format.
export.csv.mimetype	text/csv	Any valid mime-type	The MIME type that is used when sending CSV data back to the user's web browser. If you want to launch a specific program to deal with the data on the user's system you can change this property.
export.csv.include_header	false	true, false	If set to true, then the first line of the export will contain the column titles as displayed on the HTML page. By default this is set to false, so the header is not included in the export.
export.excel	true	true, false	Should the tag present the option to export data in Excel format (tab seperated values).
export.excel.label	Excel	Any string	The label on the link that the user clicks on to export the data in Excel format.
export.excel.mimetype	application/vnd.ms-excel	Any valid mime-type	The MIME type that is used when sending Excel data back to the user's web browser. I can't think of many reasons you would want to change this, but perhaps you want to launch a specific program to deal with the data on the user's system.
export.excel.include_header	false	true, false	If set to true, then the first line of the export will contain the column titles as displayed on the HTML page. By default this is set to false, so the header is not

			included in the export
export.xml	true	true, false	Should the tag present the option to export data in XML format.
export.xml.label	XML	Any string	The label on the link that the user clicks on to export the data in XML format.
export.xml.mimetype	text/xml	Any valid mime-type	The MIME type that is used when sending XML data back to the user's web browser. I can't think of many reasons you would want to change this, but perhaps you want to launch a specific program to deal with the data on the user's system.
export.amount	list	page, list	Indicates how much data should be sent down to the user when they ask for a data export. By default, it sends the entire list, but you can instruct the table tag to only send down the data that is currently being shown on the page.
export.decorated	true	true, false	Should the data be "decorated" as it is exported. The default value is true, but you might want to turn off any decoratation that is HTML specific for example when exporting the data.
paging.banner.placement	top	top, bottom, both	When the table tag has to show the header for paging through a long list, this option indicates where that header should be shown in relation to the table
paging.banner.item_name	item	Any string	What the various objects in the list being displayed should be refered to as (singular).
paging.banner.items_name	items	Any string	What the various objects in the list being displayed should be refered to as (plural).
paging.banner.no_items_found	No {0} found.	Any string in a message	What is shown in the pagination header when no objects are available in the list to be displayed. The single placeholder

		format with 1 placeholder	is replaced with the name of the items in the list (plural).
paging.banner.one_items_found	1 {0} found.	Any string in a message format with 1 placeholder	What is shown in the pagination header when one object is available in the list to be displayed. The single placeholder is replaced with the name of the items in the list (singular
paging.banner.all_items_found	{0} {1} found, showing all {2}	Any string in a message format with 3 placeholders	What is shown in the pagination header when all the objects in the list are being shown. {0} and {2} are replaced with the number of objects in the list, {1} is replaced with the name of the items {plural}.
paging.banner.some_items_found	{0} {1} found, displaying {2} to {3}	Any string in a message format with 4 placeholders	What is shown in the pagination header when a partial list of the objects in the list are being shown. {0} indicates the total number of objects in the list, {1} is replaced with the name of the items (plural}, {2} and {3} are replaced with the start and end index of the objects being shown respectively.
paging.banner.include_first_last	false	true, false	Should the banner contain a "First" and "Last" link to instantly jump to the start and end of the list. The default behavior is to not include those links
paging.banner.first_label	First	Any string	Label for the link that takes the person to the first page of objects being shown in the list.
paging.banner.last_label	Last	Any string	Label for the link that takes the person to the last page of objects being shown in the list.
paging.banner.prev_label	Prev	Any string	Label for the link that takes the person to the previous page of objects being

			shown in the list.
paging.banner.next_label	Next	Any string	Label for the link that takes the person to the next page of objects being shown in the list.
paging.banner.group_size	8	Any reasonable number	The number of pages to show in the header that this person can instantly jump to.

See Also:

Table Tag
Column Tag