Taglib Reference

In This Section

In addition to the information in this topic, see also the description of the class `nl.gx.webmanager.taglib` of the XperienCentral Javadoc.

The CMS Library

The CMS library is used for retrieving information from the XperienCentral content management system.

```
<%@taglib uri="http://www.gx.nl/taglib/wm" prefix="wm" %>
```

---

documentLink

`<wm:documentLink>` - Returns an HTML link to a document and checks for web user authorization. Example usage:

```
<wm:documentLink var="docLink" contentId="${productInfo.document}" dataSource="jdbc/internaldb" />
${docLink}
```

Attribute	Description	Required	Expression*
`contentId`	Sets the content ID of the document to which a link should be returned.	Yes	Yes
`dataSource`	Sets the data source name which can be queried to retrieve more information about the document.	Yes	Yes
`className`	Specifies the class name to be used in the `class="..."` attribute of the resulting HTML link.	No	Yes
`linkText`	The text to be used in building the link. The default value is the title of the document that is referred to.	No	Yes

`mimeTypeOnly`	Specifies whether the MIME type of the document should be returned (set to `true`) or whether to return an HTML link to the document itself (set to `false`). The default value is `false`.	No	No
`mouseOverText`	Specifies the text to be used in the `title="..."` attribute of the resulting HTML link. This text is usually shown as a tooltip if the mouse cursor is hovered over the link. The default is the value of the `linkText` attribute (see above).	No	Yes
`showDocumentTypeIcon`	Specifies whether the `documenttype` icon should be shown to left of the link. The default value is `false` (no icon).	No	No
`showTitleOnly`	Specifies whether the `linkText` (see above) should be shown as an HTML link or as normal text. The default value is `true` (normal text).	No	No
`target`	Specifies the name of the HTML frame to be used in the `target="..."` attribute of the resulting link. If the link is clicked, the document is opened in this frame. If no frame with this name can be found, the link is opened in a new browser window. The default value for this attribute is empty which means the link will open in the same window.	No	Yes
`var`	Specifies the JSP variable to store the result in.	No	No
`webUserGroups`	Sets the web user groups who are granted access to the document.	No	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

---

editable

`<wm:editable>` - Marks editable content areas on the page inside the editor. The inline edit mode in XperienCentral needs to know where editable content exists on the page. This is accomplished by wrapping the editable areas using the `<wm:editable>` tag, thereby indicating where the editable content is located. Example usage:

```
<wm:editable contentHolder="${contentholder}" class="maincontentarea" area="0" tag="div"></wm:editable>
```

This is the most basic and common usage. The class attribute is passed through to the outputted HTML. The following example shows how to wrap the title of a page version:

```
<wm:editable tag="div" contentHolder="${pageVersion}" area="title" class="main_title">
   ${pageVersion.title}
</wm:editable>
```


When wrapping content like a page title, no extra static HTML should appear inside the `<wm:editable>` tag except the page title itself. Any static HTML could be appended to the field being wrapped. Do not use your own element render loop to output the elements content in the tag itself or with a `wm:pagepart` tag.

The following are examples of incorrect usage of the `<wm:editable>` tag:

Incorrect usage of the <wm:editable> tag

```
<wm:editable contentHolder="${contentholder}" class="maincontentarea" area="0" tag="div">
   <wm:render presentationName="freestylepresentation content" object="${contentholder}" startSeparator="0"
endSeparator="1"/>
</wm:editable>
```

Incorrect usage of the <wm:editable> tag

```
<wm:editable contentHolder="${contentholder}" class="maincontentarea" area="0" tag="div"><wm:render
bject="${contentholder}"/></wm:editable>
```

Naming Convention Best Practice

It is best to give areas a meaningful name instead of assigning them a number. Doing so gives you the flexibility to change the order of areas at a later time if you decide to. For example:

```
<wm:editable contentHolder="${contentholder}" class="maincontentarea" area="maincontent" tag="div"></wm:
editable>
```

Back to Top

---

form

<wm:form> - Returns an HTML form tag and a (number of) hidden input tags enclosed in it. Additional inputs can be defined in the body of the <wm:form> tag.

The <wm:form> tag is mostly used to render various types of forms. All these forms are processed by the FormHandler which runs on the frontend and processes forms submitted by users. This `FormHandler` expects several inputs in the form. The <wm:form> tag automatically generates these. What exactly is generated, depends on the handle. Currently the following builtin handles are supported in the <wm:form> tag:

- `form` - Indicates that the form should be processed using the `FormHandler`.
- `formback` - Navigates back one page in the form flow.
- `search` - Generates a search form.
- `poll` - Generates a form for the poll functionality.
- `forum` - Generates a form for the forum functionality.
- `verseonstatus` - Generates a form to get to call the `VerseonFormStatusHandler`.
- `locationservicemappoint` - Used by the Location services functionality in XperienCentral.

Except for the form handle, these are all related to specific XperienCentral features.

Example usage (1):

```
<wm:form handle="search" onSubmit="doSearch();">
   <input type="text" name="keyword" />
</wm:form>
```

The example above displays a search form (`handle="search"`). A search form should specify a keyword, therefore an input is included to set the keyword (using the HTML `input` tag). Also, some JavaScript should be executed if the form is to be submitted. The `onSubmit` parameter takes care of that. The rendering of the form parts is delegated using the `<wm:render>` tag ***. This tag renders the form object of the form element.

Example usage (2):

```
<wm:form handle="form" onSubmit="return checkForm${formElementId}();">
   <input name="dummybutton" type="image" value="" alt="" src="${presentationcontext.website.emptyImage}"
style="position: absolute; border-style: none; padding: 0; spacing: 0; text-indent: 0;" height="0" width="0"/>
   <wm:render object="${formElement.form}" />
</wm:form>
```

Example usage (3):

```
<wm:form handle="formback" />
```

This displays the back button in a form.

Attribute	Description	Required	Expression*
`handle`	Sets the name of the handle that should be used. Valid values: see the description of the available handles above.	Yes	Yes
`extraPassOn`	Specifies a comma-separated list of extra parameters to be used in generating the URL in the form. This is the URL of the page to display when an action is performed in the form. By default the following values are already passed on: `contentid`,`dbid`,`typeofpage`,`step`,`orgurl`,`cfe`, and `forum`.	No	Yes
`target`	Specifies the name of the HTML frame to be used in the `target="..."` attribute of the resulting HTML form tag. The response to the form is displayed in the target frame. If the specified frame does not exist, the response is shown in a new window.	No	Yes
`frame`	If a value (OUTP) for this attribute is specified, an HTML input tag is created as follows in the body of the form tag: `<input type="hidden" name="frame" value="OUTP" />`. This attribute may be used in conjunction with the attribute "`target`" to specify the frame to load in the response to the form.	No	Yes
`onSubmit`	Specifies the name of the JavaScript function to be used in the `onSubmit="..."` attribute of the resulting HTML form tag. This function can be used to, for example, do some error checking on the form data. The default value is empty (no function).	No	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

The following filterHTML tags were introduced in XperienCentral version 10.19.1. These methods rely on the JSoup `clean` *method.*

filterHTMLBasic

`<wmfn:filterHTMLBasic>` - Filters the HTML using a basic filtering configuration. It allows the following range of text nodes: `<a>`, `<b>`, `<blockquote>`, `<br>`, `<cite>`, `<code>`, `<dd>`, `<dl>`, `<dt>`, `<em>`, `<i>`, `<li>`, `<ol>`, `<p>`, `<pre>`, `<q>`, `<small>`, `<span>`, `<strike>`, `<strong>`, `<sub>`, `<sup>`, `<u>`, `<ul>` and `<l>`.

```
${wmfn:filterHTMLBasic(myHtmlToFilter)}
```

Back to Top

---

filterHTMLRelaxed

`<wmfn:filterHTMLRelaxed>` - Filters the HTML using a relaxed filtering configuration. It allows the following range of text and structural body HTML: `<a>`, `<b>`, `<blockquote>`, `<br>`, `<caption>`, `<cite>`, `<code>`, `<col>`, `<colgroup>`, `<dd>`, `<div>`, `<dl>`, `<dt>`, `<em>`, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, `<i>`, `<img>`, `<li>`, `<ol>`, `<p>`, `<pre>`, `<q>`, `<small>`, `<span>`, `<strike>`, `<strong>`, `<sub>`, `<sup>`, `<table>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>`, `<tr>`, `<u>`, `<ul>` and `<l>`. Example usage:

```
${wmfn:filterHTMLRelaxed(myHtmlToFilter)}
```

Back to Top

---

filterHTMLTags

`<wmfn:filterHTMLTags>` - Filters the HTML. Passed tags are allowed — other tags are removed from the HTML. Example usage:

```
<c:set var="imagesFilter" value="figure,img,figcaption" />

<c:set var="allImages" value="${wmfn:filterHTMLTags(myContent, imagesFilter)}" />
```

Back to Top

---

filterHTMLTagsAndAttributes

`<wmfn:filterHTMLTagsAndAttributes>` - Filters the HTML. Passed tags and attributes are allowed — other tags and attributes are removed from the HTML. Example usage:

```
Create an overview of only the a links on an html page:
${wmfn:filterHTMLTagsAndAttributes(myHtmlToFilter, 'a', 'href')}

Extract all images from an html page:
${wmfn:filterHTMLTagsAndAttributes(myHtmlWithImages, 'figure,img,figcaption,a', 'src,width,height,href')}

Remove any styling from the forms in content:
${wmfn:filterHTMLTagsAndAttributes(myStyledForms, 'form,input,fieldset,textarea,label,select,option', 'action,
method,type,value,name,for')}
```

Back to Top

---

formPart

`<wm:formPart>` - This tag retrieves a form part using its alias as the search criteria. Example usage:

```
<wm:formPart
var="userNameFormPart" alias="username" />
${userNameFormPart}
```

The fragment above will look up the form part with the alias "username".

Attribute	Description	Required	Expression*
`alias`	Specifies the alias of the form part to search for. This is the alias that is used in the XSL of the form handlers to identify the form part.	Yes	Yes
`var`	Specifies the JSP variable in which to store the result.	Yes	No

** The attribute can support EL (Expression Language) values.*

Back to Top

---

getMediaItems

`<wm:getMediaItems>` - This tag retrieves media items from the Content Repository and sorts the results. Example usage:

```
<wm:formPart var="userNameFormPart" alias="username" /> ${userNameFormPart}
```

The example above retrieves all media items that match a specified term.

Attribute	Description	Required	Expression*
`andnotterms`	Specifies a comma separated list of terms the returned media item doesn't have. This includes terms present in other web initiatives (XperienCentral R25 and higher).	No	Yes
`andterms`	Specifies a comma separated list of terms the returned media item must have. This includes terms present in other web initiatives (XperienCentral R25 and higher).	No	Yes
`contenttypes`	Specifies the content type of the media item (article, image, etc.).	No	Yes
`createdsince`	Specifies the date after which the returned media items must have been created.	No	Yes
`excludemediaitemids`	Specifies the returned media item IDs that need to be excluded.	No	Yes
`orterms`	Specifies the term(s) that the returned media items must have at least one of. This includes terms present in other web initiatives (XperienCentral R25 and higher).	No	Yes
`publicationdatefrom`	Specifies the beginning of the date range in which the returned media item was published.	No	Yes
`publicationdateto`	Specifies the end of the date range in which the returned media item was published.	No	Yes
`referenceddatabaseentity`	Specifies the database entity the returned media item is linked to.	No	Yes
`referencedresourceinstance`	Specifies the resource instance the returned media item is linked to.	No	Yes
`resultrange`	Specifies the number of returned media items to display.	No	Yes

`sortascending`	Specifies whether to sort the returned media items in ascending order.	No	Yes
`sortoptions`	Specifies how to sort the returned media items. The options are: • `creationdate` • `mostviewed` • `mostreactions` • `publicationdate` • `modificationdate` • `voteaverage` • `webtitle` • `lastpost`	No	Yes
`statusid`	Specifies the media item's workflow state.	No	Yes
`websiteid`	Specifies the ID of the website from which the media items are retrieved.	No	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

---

image

`<wm:image>` - Creates an Image object from its URL. Example usage:

```
<wm:image var="photo" url="http://www.gx.nl/images/picture.jpg" alternativeText="a picture" />
${image.htmlTag}
```

Attribute	Description	Required	Expression*
`alias`	Specifies the alias of the form part to search for. This is the alias that is used in the XSL of the form handlers to identify the form part.	Yes	Yes
`var`	Specifies the JSP variable in which to store the result.	Yes	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

includeUrl

`<wm:includeUrl>` - This tag evaluates the URL to which the given parameter points to and writes the result to the JSP page. External URLs should start with `http`. Internal URLs should be relative to the context path (i.e. start with /). Use this instead of `jsp:include` to pass on the presentation context. Example usage:

```
<wm:includeUrl url="/WEB-INF/presentation/jsp/shared/include/top.jspf" />
```

Attribute	Description	Required	Expression*

`url`	Specifies the URL indicating the resource to retrieve the HTML from. To include references to other JSP files, use the relative path found on the “Presentation” tab in XperienCentral.	Yes	Yes
`connectionTimeout`	Sets the connection timeout in milliseconds for external URLs. The default value is `5000` (= 5 sec.).	No	Yes
`followRedirects`	Specifies whether a redirect should be followed if the URL returns a redirect to another location. The default value is `true` (= follow the redirect).	No	Yes
`readTimeout`	Sets the read timeout in milliseconds for external URLs. Input. The default value is `10000` (= 10 sec.).	No	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

---

isActive

`<wm:isActive>` - This tag specifies whether the object that is specified comes from a component that is currently active. For example, it returns `false` when invoked on an element that is contained by a plugin that is in the RESOLVED state (currently not running). Example usage:

```
<wm:isActive var="componentLoaded" object="${componentObject}"/>
<c:if test="componentLoaded">
...
</c:if>
```

Attribute	Description	Required	Expression*
`object`	Specifies the Java object to check.	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	No	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

link

`<wm:link>` - Using this tag, link objects can be constructed that refer to pages, downloads, etc. This tag supports dynamic attributes (i.e. you can specify your own attributes) which are passed on to the link object that is created.

Example (1):

```
<wm:link var="aLinkObject" linkText="clickme" foo="bar" target="_top" />
${aLinkObject.htmlTag}
```

In the example above, a new link object is created referencing the current page. It is assigned to `aLinkObject`. Using dynamic attributes a custom attribute (foo) is defined. The link is opened in the `_top` section of the window.

Example (2):

```
<wm:link var="homepage" reference="${presentationcontext.website.homePage}" linkText="to the homepage" />
Click here to go ${homepage.htmlTag}.
```

Attribute	Description	Required	Expression*
`var`	Specifies the JSP variable in which to store the result in.	Yes	No
`className`	Specifies the class name to be used in the `class="..."` attribute of the resulting HTML link.	No	Yes
`linkText`	Sets the text to use when displaying the HTML link. The default value is the title of the page that is linked to.	No	Yes
`mouseOverText`	Specifies the text to be used in the `title="..."` attribute of the resulting HTML link. This text is usually shown as a tooltip if the mouse cursor is hovered over the link.	No	Yes
`passOn`	Specifies a comma separated list of all URL parameters of the current request that need to be included in the link that is built.	No	Yes
`passOnAllParameters`	Specifies whether all URL parameters of the current request are included in the link that is built. The default value is `false` (don't include).	No	Yes
`presentationName`	Specifies an XperienCentral presentation object. The link is rendered by this presentation.	No	Yes
`reference`	The page to link to. If not set, the current page will be used.	No	Yes
`target`	Specify the target to be used in the `target="..."` attribute of the resulting HTML link. Using this attribute, links can be opened in a new browser window. The default value is empty which means the link will open in the current window.	No	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

---

object

`<wm:object>` - Retrieves an object from the content database which is filled with objects that have an ID and a type. For XperienCentral objects that are exposed through the Java API, wrapper classes are available that encapsulate the data that is stored in the content database. The public API of these wrapper classes is defined in interfaces. The `<wm:object>` tag can be used as a factory for creating wrapper objects based on the type and object ID. The type is the fully qualified class name of the wrapper interface. If no wrapper object can be created (either the ID or the type is wrong), the value null is used. Example usage:

```
<wm:object var="message" objectId="${param.message}" objectType="nl.gx.webmanager.cms.forum.ForumMessage"/>
<wm:render object="${message}" />
```

The example above assigns a `ForumMessage` object to the variable `message`. Some tags like `<wm:render>` have an attribute object. The variable may be used in such tags.

Attribute	Description	Required	Expression*
`hideError`	*The* `hideError` *attribute was introduced in XperienCentral version R24.* When set to `true`, specifies that XperienCentral should not log an error if the specified `objectId` cannot be found.	No	No
`objectId`	Specifies the numerical ID of the object to retrieve.	Yes	Yes
`objectType`	Specifies the Java class of the object to retrieve. The `objectType` should be a fully qualified classname. Note there are some deprecated abbreviations for some classes which are still supported. You should always use the fully qualified classname. In future releases, support for the abbreviations may be dropped.	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	Yes	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

pagePart

`<wm:pagePart>` - The label defines which JSPF is used to generate the content (looked up in the presentation mappings that XperienCentral creates by reading the presentation descriptor files). An XperienCentral page can contain content consisting of multiple elements, each with its own presentation JSPs. To show these elements, the page presentation JSP should contain a `wm` tag that renders these elements.

Example 1:

```
<div id="column1">
   <wm:pagePart label="Content column 1" />
</div>
```

This tells XperienCentral that a pagePart with the label “Content column 1” should be displayed here. The “Content column 1” label is defined in the presentation descriptor file of the content JSPF (i.e. `content.xml`). The `<wm:pagePart>` tag effectively tells XperienCentral to execute that JSPF and include the results.

Example 2:

```
<wm:pagePart label="WM content" var="result"/>
${result}
```

Create custom pageParts by defining a .JSPF and an XML descriptor file.

Attribute	Description	Required	Expression*
`label`	Specifies the label of the page part to render. This must be an exact match of the value of the `<name>` defined in the descriptor XML file and the `<scope>` in that file must be “pagepart”.	Yes	Yes
`object`	Specifies the object to render. The default value is `${presentationcontext.baseObject}` (the object that is rendered by the jsp this tag is in).	No	Yes
`var`	Specifies the JSP variable to store the result in.	No	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

pageMetaData

`<wm:pageMetaData>` - Retrieves a `pagemetadata` object of the requested type for the current page version. Example usage:

```
<wm:pageMetaData var="keywords" type="keywords" />
<meta name="keywords" value="${keywords}" />
```

Attribute	Description	Required	Expression*
`type`	Specifies the type of metadata. The valid values are `keywords` and `description`.	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	Yes	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

presentationProperty

`<wm:presentationProperty>` - Looks up a presentation property defined in the presentation context and returns its value in a JSP variable. Several presentation property types in addition to String and Boolean are supported:

- `string`
- `boolean`
- `image`
- `font`
- `button`
- `color`
- `layout`

The type for the presentation properties is defined in the presentation descriptor. The values for the properties can be defined in the descriptor files by creating presentation variants in XperienCentral or by relying on the style information that is defined in XperienCentral. Example usage:

```
<wm:presentationProperty var="bulletImage" label="bullet" />
${bulletImage.htmlTag}
```

In the example above, the presentation property named “bullet” is retrieved and its value is returned in the variable `bulletImage`.

Attribute	Description	Required	Expression*
`label`	Specifies the property label to retrieve.	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	Yes	No
`page`	Specifies the page to used to retrieve a style property from. For example, this is used for page images in a navigation menu. This attribute is only used for properties of type `image`, `button`, `color` and `font`. For other property types this attribute is ignored.		
`useTargetPageStyle`	Specifies whether the style of the target page is used to retrieve a style property instead of the style of the current page. This attribute is only used for properties of type `image`, `button`, `color` and `font`. For other property types this attribute is ignored.		

**The attribute can support EL (Expression Language) values.*

Back to Top

---

render

`<wm:render>` - renders an object using a presentation.

Example (1):

```
<wm:render var="pollForm" presentationName="WM pollelement form"/>
<wm:render var="pollResult" presentationName="WM pollelement result" />
```

In the example above, a poll element object is rendered using two different presentations. In this case the default presentation of the poll element is not used because the optional `presentationName` attribute is provided in the `<wm:render>` tag. Additionally, no object is specified, which means that the object which is being rendered is the same as the object that is rendered by the JSP this code is in. The example renders both the form and the results into separate variables. Depending on whether the user has submitted the poll, either the form for doing so or the poll results can be displayed. You can also explicitly specify an object and no presentation.

Example (2):

```
<wm:render object="${element}" />
```

The `<wm:render>` tag renders the object `${element}` using whatever presentation is associated with the object in XperienCentral.

Attribute	Description	Required	Expression*
`object`	Specifies the object to render. The default value is `${presentationcontext.baseObject}` (the object that is rendered by the JSP this tag is in).	Yes	Yes
`presentationName`	Sets the presentation that the object should be rendered with. The default value is empty, which means the object is rendered using the presentation assigned in XperienCentral.	Yes	No
`var`	Specifies the JSP variable to store the result in.	No	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

sizedImage

`<wm:sizedImage>` - Resizes image objects. If the aspect ratio (height/width) of the resized image is different than the source image, the image will be cropped from the top and bottom or left and right to avoid the displaying of black bars. It returns the resized image object. Example usage:

```
<wm:image var="bullet" url="http://www.gxsoftware.com/images/bullet.jpg "/>
<wm:sizedImage var="bulletSmall" image="${bullet}" width="20" height="20" />
${bulletSmall.htmlTag}
```

In the example above the image stored in the variable `bullet` is resized to 20x20 pixels and returned in the variable `bulletSmall`.

Attribute	Description	Required	Expression*
`height`	Sets the height of the resized image in pixels.	Yes	Yes

`image`	Specifies the image object to resize.	Yes	No
`var`	Specifies the JSP variable to store the result in.	Yes	No
`width`	Sets the width of the resized image in pixels.	Yes	Yes

**The attribute can support EL (Expression Language) values.*

Back to Top

---

specialPages

`<wm:specialPages>` - Returns the special pages for a specific label. In XperienCentral special pages can be defined for several types of labels. For example, a standard media repository page can be selected or RSS feed pages, etc. Example usage:

```
<wm:specialPages var="pages" label="footerPages" />
<c:forEach items="${pages}" var="page">
   <wm:link reference="${page}" />
</c:forEach>
```

In the example above all footer Pages (defined in the XperienCentral Configuration menu) are returned in the variable `pages`. A link to each of these pages is created.

Attribute	Description	Required	Expression*
`label`	Specifies the special page label as defined in XperienCentral	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	Yes	No

**The attribute can support EL (Expression Language) values.*

Back to Top

---

text

`<wm:text>` - This tag looks up the text value defined for a language label as configured in XperienCentral. The configuration can be found in the Configure > Language Labels menu in XperienCentral. If the `var` attribute is set, the result is bound to this variable, otherwise the result of the lookup is printed. Example usage:

```
<wm:text var="submitText" label="wm_language.form_submit"/>
<input type="submit" name="${submitText}" />
```

In the example above the text defined for the label "wm_language.form_submit" is retrieved for the current language. A button using this text is displayed.

Attribute	Description	Required	Expression*
`label`	Specifies the text label as defined in XperienCentral	Yes	Yes
`var`	Specifies the JSP variable to store the result in.	No	No

** The attribute can support EL (Expression Language) values.*

Back to Top

The Functions Library

The Functions library contains functions for manipulating data in XperienCentral. Example usage:

```
<%@taglib uri="http://www.gxsoftware.com/taglib/functions" prefix="wmfn"%>
```

Back to Top

---

function

```
<script type="text/javascript">
alert('page title is ${wmfn:escapeToJavascript(pageVersion.title)}');
</script>
```

Function	Description

`addLineFeed`
 	
`: string`
	Retrieves a line feed (usually line feeds in JSP pages are ignored).

`contains`
 	
`(object, object[]) : boolean`
	Returns `true` if the array contains the instance.

`convertToJavaScript`
 	
`(string) : string`
	Converts HTML code to JavaScript document.write statements.

`escapeToJavaScript`
 	
`(string) : string`
	Replaces `\\` with `\\\\` and `'` with `\\'` in a string so that it can be used in a JavaScript statement.

`escapeToText`
 	
`(string) : string`
	Surrounds the text part with a `<pre>` tag so that while processing this part should be taken literally and so that line feeds should be preserved.

`formatExternalUrl`
 	
`(string) : string`
	Prepends `http://` before an external URL if it doesn't start with `http://`, `https://`, `ftp://`, `file://`, `javascript:`, `mailto:`, `/` or `#`.

`formSignatureParameterName`
 	
`(servletcontext) : string`
	If secure forms are enabled, this function retrieves the name of the secure form request parameter. This function returns the name of the form parameter which should be used for sending the form signature.

`indexOf`
 	
`(object, object[]): int`
	Returns the index of the object in the array or `-1` if the array doesn't contain the object.

`instanceOf`
 	
`(object, string): boolean`
	This is more flexible `instanceOf` implementation than the built-in Java `instanceOf` operator.

`replaceNewLines` `(string) : string`	Replaces new lines with ` `.

`signFormParameters`
 	
`(servletcontext, servletrequest, string) : string`
	Computes a signature for a set of names for the use of secure forms in SiteWorks.

`signFragment`
 	
`(servletcontext, servletrequest, string) : string`
	Generates a secure forms signature for the given HTML fragment.

`sortRows`
 	
`(object[][], int, boolean) : object[]`
	Sorts a two dimensional array (an array of arrays) by the objects in a column.

`sortMaps`
 	
`(sortedmap[], string, boolean) : sortedmap`
	Sorts an array of `SortedMaps` by the elements at a specified key.

`subArray`
 	
`(object[], int, int) : object[]`
	Returns a sub array of the original array.

`urlEncode`
 	
`(string) : string`
	The URL encodes a UTF-8 string.

Back to Top