**TAG LIBRARIES**

The Granite, CQ, and Sling tag libraries give you access to specific functions for use in the JSP script of your templates and components.

**Granite Tag Library**

The Granite tag library contains helpful functions.

When you develop the jsp script of a Granite UI component, it is recommended to include following code at the top of the script:

|  |  |
| --- | --- |
|  | **<%@include file="/libs/granite/ui/global.jsp"%>** |

The global also declares the [Sling library](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#main-pars_par13).

|  |  |
| --- | --- |
|  | **<%@taglib prefix="sling" uri="**[**http://sling.apache.org/taglibs/sling**](http://sling.apache.org/taglibs/sling)**" %>** |

**<ui:includeClientLib>**

The **<ui:includeClientLib>** tag Includes a AEM html client library, which can be a js, a css, or a theme library. For multiple inclusions of different types, for example js and css, this tag needs to be used multiple times in the jsp. This tag is a convenience wrapper around the **[com.adobe.granite.ui.clientlibs.HtmlLibraryManager](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/reference-materials/javadoc/com/adobe/granite/ui/clientlibs/HtmlLibraryManager.html)** service interface.

It has the following attributes:

**categories**

A list of comma-separated client lib categories. This will include all Javascript and CSS libraries for the given categories. The theme name is extracted from the request.

Equivalent to: **com.adobe.granite.ui.clientlibs.HtmlLibraryManager#writeIncludes**

**theme**

A list of comma-separated client lib categories. This will include all theme related libraries (both CSS and JS) for the given categories. The theme name is extracted from the request.

Equivalent to: **com.adobe.granite.ui.clientlibs.HtmlLibraryManager#writeThemeInclude**

**js**

A list of comma-separated client lib categories. This will include all Javascript libraries for the given categories.

Equivalent to: **com.adobe.granite.ui.clientlibs.HtmlLibraryManager#writeJsInclude**

**css**

A list of comma-separated client lib categories. This will include all CSS libraries for the given categories.

Equivalent to: **com.adobe.granite.ui.clientlibs.HtmlLibraryManager#writeCssInclude**

**themed**

A flag that indicates of only themed or non themed libraries should be included. If omitted, both sets are included. Only applies to pure JS or CSS includes (not for categories or theme includes).

The **<ui:includeClientLib>** tag can be used as follows in a jsp:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11** | **<%-- all: js + theme (theme-js + css) --%>**  **<ui:includeClientLib categories="cq.wcm.edit" />**    **<%-- only js libs --%>**  **<ui:includeClientLib js="cq.collab.calendar, cq.security" />**    **<%-- theme only (theme-js + css) --%>**  **<ui:includeClientLib theme="cq.collab.calendar, cq.security" />**    **<%-- css only --%>**  **<ui:includeClientLib css="cq.collab.calendar, cq.security" />** |

**CQ Tag Library**

The CQ tag library contains helpful functions.

To use the CQ Tag Library in your script, the script must start with the following code:

|  |  |
| --- | --- |
|  | **<%@taglib prefix="cq" uri="**[**http://www.day.com/taglibs/cq/1.0**](http://www.day.com/taglibs/cq/1.0)**" %>** |

**Note:**

When the **/libs/foundation/global.jsp** file is included in the script, the taglib is automatically declared.

When you develop the jsp script of a AEM component, it is recommended to include following code at the top of the script:

|  |  |
| --- | --- |
|  | **<%@include file="/libs/foundation/global.jsp"%>** |

It declares the sling, CQ, and jstl taglibs and exposes the regularly used scripting objects defined by the [**<cq:defineObjects />**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#cqdefineObjects) tag. This shortens and simplifies the jsp code of your component.

**<cq:text>**

The **<cq:text>** tag is a convenience tag that outputs component text in a JSP.

It has the following optional attributes:

**property**

Name of the property to use. The name is relative to the current resource.

**value**

Value to use for output. If this attribute is present, it overwrites the use of the **property** attribute.

**oldValue**

Value to use for diff output. If this attribute is present, it overwrites the use of the **property** attribute.

**escapeXml**

Defines whether the characters **<, >, &, '** and **"** in the resulting string should be converted to their corresponding character entity codes. Default value is false. Note that the escaping is applied after the optional formatting.

**format**

Optional **java.text.Format** to use for formatting the text.

**noDiff**

Suppresses the calculation of a diff output, even if a diff info is present.

**tagClass**

CSS class name of an element that will surround a non-empty output. If empty, no element is added.

**tagName**

Name of the element that will surround a non-empty output. It defaults to **DIV**.

**placeholder**

Default value to use for null or empty text in edit mode, i.e. the placeholder. Please note that the default check is performed after the optional formatting and escaping, i.e. it is written as-is to the output. It defaults to:

**<div><span class="cq-text-placeholder">&para;</span></div>**

**default**

Default value to use for null or empty text. Note that the default check is performed after the optional formatting and escaping i.e. it is written as-is to the output.

Some examples how the **<cq:text>** tag can be used in a JSP:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25** | **<cq:text property="jcr:title" tagName="h2"/>**  **<cq:text property="jcr:description" tagName="p"/>**    **<cq:text value="<%= listItem.getTitle() %>" tagName="h4" placeholder="" />**  **<cq:text value="<%= listItem.getDescription() %>" tagName="p" placeholder=""/>**    **<cq:text property="jcr:title" value="<%= title %>" tagName="h3"/><%**  **} else if (type.equals("link")) {**  **%><cq:text property="jcr:title" value="<%= "\u00bb " + title %>" tagName="p" tagClass="link"/><%**  **} else if (type.equals("extralarge")) {**  **%><cq:text property="jcr:title" value="<%= title %>" tagName="h1"/><%**  **} else {**  **%><cq:text property="jcr:title" value="<%= title %>" tagName="h2"/><%**    **<cq:text property="jcr:description" placeholder="" tagName="small"/>**    **<cq:text property="tableData"**  **escapeXml="false"**  **placeholder="<img src=\"/libs/cq/ui/resources/0.gif\" class=\"cq-table-placeholder\" alt=\"\">"**  **/>**    **<cq:text property="text"/>**    **<cq:text property="image/jcr:description" placeholder="" tagName="small"/>**  **<cq:text property="text" tagClass="text"/>** |

**<cq:setContentBundle>**

The **<cq:setContentBundle>** tag creates an i18n localization context and stores it in the **javax.servlet.jsp.jstl.fmt.localizationContext** configuration variable.

It has the following attributes:

**language**

The language of the locale for which to retrieve the resource bundle.

**source**

The source where the locale should be taken from. It can be set to one of the following values:

* **static**: the locale is taken from the **language** attribute if available, otherwise from the server default locale.
* **page**: the locale is taken from the language of the current page or resource if available, otherwise from the **language**attribute if available, otherwise from the server default locale.
* **request**: the locale is taken from the request locale (**request.getLocale()**).
* **auto**: the locale is taken from the **language** attribute if available, otherwise from the language of the current page or resource if available, otherwise from the request.

If the **source** attribute is not set:

* If the **language** attribute is set, the **source** attribute defaults to **static**.
* If the **language** attribute is not set, the **source** attribute defaults to **auto**.

The "content bundle" can be simply used by standard JSTL **<fmt:message>** tags. The lookup of messages by keys is two-fold:

1. First, the JCR properties of the underlying resource that is currently rendered are searched for translations. This allows you to define a simple component dialog to edit those values.
2. If the node does not contain a property named exactly like the key, the fallback is to load a resource bundle from the sling request (**SlingHttpServletRequest.getResourceBundle(Locale)**). The language or locale for this bundle is defined by the language and source attributes of the **<cq:setContentBundle>** tag.

The **<cq:setContentBundle>** tag can be used as follows in a jsp.

For pages that define their language:

|  |  |
| --- | --- |
| **1**  **2**  **3** | **... %><cq:setContentBundle source="page"/><%  %>**  **<div class="error"><fmt:message key="Hello"/>**  **</div> ...** |

For user personalized pages:

|  |  |
| --- | --- |
| **1**  **2**  **3** | **... %><cq:setContentBundle scope="request"/><% %>**  **<div class="error"><fmt:message key="Hello"/>**  **</div> ...** |

**<cq:include>**

The <cq:include> tag includes a resource into the current page.

It has the following attributes:

**flush**

* A boolean defining whether to flush the output before including the target.

**path**

* The path to the resource object to be included in the current request processing. If this path is relative it is appended to the path of the current resource whose script is including the given resource. Either path and resourceType, or script must be specified.

**resourceType**

* The resource type of the resource to be included. If the resource type is set, the path must be the exact path to a resource object: in this case, adding parameters, selectors and extensions to the path is not supported.
* If the resource to be included is specified with the path attribute that cannot be resolved to a resource, the tag may create a synthetic resource object out of the path and this resource type.
* Either path and resourceType, or script must be specified.

**script**

* The jsp script to include. Either path and resourceType, or script must be specified.

**ignoreComponentHierarchy**

* A boolean controlling whether the component hierarchy should be ignored for script resolution. If true, only the search paths are respected.

**Example:**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7** | **<%@taglib prefix="cq" uri="**[**http://www.day.com/taglibs/cq/1.0**](http://www.day.com/taglibs/cq/1.0)**" %><%**  **%><div class="center">**  **<cq:include path="trail" resourceType="foundation/components/breadcrumb" />**  **<cq:include path="title" resourceType="foundation/components/title" />**  **<cq:include script="redirect.jsp"/>**  **<cq:include path="par" resourceType="foundation/components/parsys" />**  **</div>** |

 Should you use **<%@ include file="myScript.jsp" %>** or **<cq:include script="myScript.jsp" %>** to include a script?

* The **<%@ include file="myScript.jsp" %>** directive informs the JSP compiler to include a complete file into the current file. It is as if the contents of the included file were pasted directly into the original file.
* With the **<cq:include script="myScript.jsp">** tag, the file is included at runtime.

Should you use **<cq:include>** or **<sling:include>**?

* When developing AEM components, Adobe recommends that you use **<cq:include>**.
* **<cq:include>** allows you to directly include script files by their name when using the script attribute. This takes component and resource type inheritance into account, and is often simpler than strict adherence to Sling's script resolution using selectors and extensions.

**<cq:includeClientLib>**

**Caution:**

**<cq:includeClientLib>** has been deprecated since AEM 5.6. [**<ui:includeClientLib>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#main-pars_title_1565813984) should be used instead.

The **<cq:includeClientLib>** tag Includes a AEM html client library, which can be a js, a css or a theme library. For multiple inclusions of different types, for example js and css, this tag needs to be used multiple times in the jsp. This tag is a convenience wrapper around the **com.day.cq.widget.HtmlLibraryManager** service interface.

It has the following attributes:

**categories**

A list of comma-separated client lib categories. This will include all Javascript and CSS libraries for the given categories. The theme name is extracted from the request.

Equivalent to: **com.day.cq.widget.HtmlLibraryManager#writeIncludes**

**theme**

A list of comma-separated client lib categories. This will include all theme related libraries (both CSS and JS) for the given categories. The theme name is extracted from the request.

Equivalent to: **com.day.cq.widget.HtmlLibraryManager#**writeThemeInclude

**js**

A list of comma-separated client lib categories. This will include all Javascript libraries for the given categories.

Equivalent to: **com.day.cq.widget.HtmlLibraryManager#writeJsInclude**

**css**

A list of comma-separated client lib categories. This will include all CSS libraries for the given categories.

Equivalent to: **com.day.cq.widget.HtmlLibraryManager#writeCssInclude**

**themed**

A flag that indicates of only themed or non themed libraries should be included. If omitted, both sets are included. Only applies to pure JS or CSS includes (not for categories or theme includes).

The **<cq:includeClientLib>** tag can be used as follows in a jsp:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11** | **<%-- all: js + theme (theme-js + css) --%>**  **<cq:includeClientLib categories="cq.wcm.edit" />**    **<%-- only js libs --%>**  **<cq:includeClientLib js="cq.collab.calendar, cq.security" />**    **<%-- theme only (theme-js + css) --%>**  **<cq:includeClientLib theme="cq.collab.calendar, cq.security" />**    **<%-- css only --%>**  **<cq:includeClientLib css="cq.collab.calendar, cq.security" />** |

**<cq:defineObjects>**

The **<cq:defineObjects>** tag exposes the following, regularly used, scripting objects which can be referenced by the developer. It also exposes the objects defined by the [**<sling:defineObjects>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#slingdefineObjects) tag.

**componentContext**

* the current component context object of the request (com.day.cq.wcm.api.components.ComponentContext interface).

**component**

* the current AEM component object of the current resource (com.day.cq.wcm.api.components.Component interface).

**currentDesign**

* the current design object of the current page (com.day.cq.wcm.api.designer.Design interface).

**currentPage**

* the current AEM WCM page object (com.day.cq.wcm.api.Page interface).

**currentStyle**

* the current style object of the current cell (com.day.cq.wcm.api.designer.Style interface).

**designer**

* the designer object used to access design information (com.day.cq.wcm.api.designer.Designer interface).

**editContext**

* the edit context object of the AEM component (com.day.cq.wcm.api.components.EditContext interface).

**pageManager**

* the page manager object for page level operations (com.day.cq.wcm.api.PageManager interface).

**pageProperties**

* the page properties object of the current page (org.apache.sling.api.resource.ValueMap).

**properties**

* the properties object of the current resource (org.apache.sling.api.resource.ValueMap).

**resourceDesign**

* the design object of the resource page (com.day.cq.wcm.api.designer.Design interface).

**resourcePage**

* the resource page object (com.day.cq.wcm.api.Page interface).
* It has the following attributes:

**requestName**

* inherited from sling

**responseName**

* inherited from sling

**resourceName**

* inherited from sling

**nodeName**

* inherited from sling

**logName**

* inherited from sling

**resourceResolverName**

* inherited from sling

**slingName**

* inherited from sling

**componentContextName**

* specific to wcm

**editContextName**

* specific to wcm

**propertiesName**

* specific to wcm

**pageManagerName**

* specific to wcm

**currentPageName**

* specific to wcm

**resourcePageName**

* specific to wcm

**pagePropertiesName**

* specific to wcm

**componentName**

* specific to wcm

**designerName**

* specific to wcm

**currentDesignName**

* specific to wcm

**resourceDesignName**

* specific to wcm

**currentStyleName**

* specific to wcm

**Example**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4** | **<%@page session="false" contentType="text/html; charset=utf-8" %><%**  **%><%@ page import="com.day.cq.wcm.api.WCMMode" %><%**  **%><%@taglib prefix="cq" uri="**[**http://www.day.com/taglibs/cq/1.0**](http://www.day.com/taglibs/cq/1.0)**" %><%**  **%><cq:defineObjects/>** |

**Note:**

When the **/libs/foundation/global.jsp** file is included in the script, the **<cq:defineObjects />** tag is automatically included.

**<cq:requestURL>**

The **<cq:requestURL>** tag writes the current request URL to the JspWriter. The two tags [**<cq:addParam>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#cqaddParam) and [**<cq:removeParam>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#cqremoveParam) and may be used inside the body of this tag to modify the current request URL before it is written.

It allows you to create links to the current page with varying parameters. For example, it enables you to transform the request:

**mypage.html?mode=view&query=something** into **mypage.html?query=something**.

The use of **addParam** or **removeParam** only changes the occurrence of the given parameter, all other parameters are unaffected.

**<cq:requestURL>** does not have any attribute.

Examples:

|  |  |
| --- | --- |
| **1** | **<a href="<cq:requestURL><cq:removeParam name="language"/></cq:requestURL>">remove filter</a>** |
| **1** | **<a title="filter results" href="<cq:requestURL><cq:addParam name="language" value="${bucket.value}"/></cq:requestURL>">${label} (${bucket.count})</a>** | |

**<cq:addParam>**

The **<cq:addParam>** tag adds a request parameter with the given name and value to the enclosing [**<cq:requestURL>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#cqrequestURL) tag.

It has the following attributes:

**name**

* name of the parameter to be added

**value**

* value of the parameter to be added

**Example:**

|  |  |
| --- | --- |
|  | **<a title="filter results" href="<cq:requestURL><cq:addParam name="language" value="${bucket.value}"/></cq:requestURL>">${label} (${bucket.count})</a>** |

**<cq:removeParam>**

The **<cq:removeParam>** tag removes a request parameter with the given name and value from the enclosing [**<cq:requestURL>**](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/taglib.html#cqrequestURL) tag. If no value is provided all parameters with the given name are removed.

It has the following attributes:

**name**

* name of the parameter to be removed

Example:

|  |  |
| --- | --- |
| **1** | **<a href="<cq:requestURL><cq:removeParam name="language"/></cq:requestURL>">remove filter</a>** |

**Sling Tag Library**

The Sling tag library contains helpful Sling functions.

When you use the Sling Tag Library in your script, the script must start with the following code:

|  |  |
| --- | --- |
|  | **<%@ taglib prefix="sling" uri="**[**http://sling.apache.org/taglibs/sling/1.0**](http://sling.apache.org/taglibs/sling/1.0)**" %>** |

**Note:**

When the **/libs/foundation/global.jsp** file is included in the script, the sling taglib is automatically declared.

**<sling:include>**

The **<sling:include>** tag includes a resource into the current page.

It has the following attributes:

**flush**

* A boolean defining whether to flush the output before including the target.

**resource**

* The resource object to be included in the current request processing. Either resource or path must be specified. If both are specified, the resource takes precedence.

**path**

* The path to the resource object to be included in the current request processing. If this path is relative it is appended to the path of the current resource whose script is including the given resource. Either resource or path must be specified. If both are specified, the resource takes precedence.

**resourceType**

* The resource type of the resource to be included. If the resource type is set, the path must be the exact path to a resource object: in this case, adding parameters, selectors and extensions to the path is not supported.
* If the resource to be included is specified with the path attribute that cannot be resolved to a resource, the tag may create a synthetic resource object out of the path and this resource type.

**replaceSelectors**

* When dispatching, the selectors are replaced with the value of this attribute.

**addSelectors**

* When dispatching, the value of this attribute is added to the selectors.

**replaceSuffix**

* When dispatching, the suffix is replaced by the value of this attribute.

**Note:**

The resolution of the resource and the script that are included with the **<sling:include>** tag is the same as for a normal sling URL resolution. By default, the selectors, extension, etc. from the current request are used for the included script as well. They can be modified through the tag attributes: for example **replaceSelectors="foo.bar"** allows you to overwrite the selectors.

Examples:

|  |  |
| --- | --- |
|  | **<div class="item"><sling:include path="<%= pathtoinclude %>"/></div>** |
|  | **<sling:include resource="<%= par %>"/>** |

|  |  |
| --- | --- |
|  | **<sling:include addSelectors="spool"/>** |
|  | **<sling:include resource="<%= par %>" resourceType="<%= newType %>"/>** |

|  |  |
| --- | --- |
|  | **<sling:include resource="<%= par %>" resourceType="<%= newType %>"/>** |
|  | **<sling:include replaceSelectors="content" />** |

**<sling:defineObjects>**

The **<sling:defineObjects>** tag exposes the following, regularly used, scripting objects which can be referenced by the developer:

**slingRequest**

* SlingHttpServletRequest object, providing access to the HTTP request header information - extends the standard HttpServletRequest - and provides access to Sling-specific things like resource, path info, selector, etc.

**slingResponse**

* SlingHttpServletResponse object, providing access for the HTTP response that is created by the server. This is currently the same as the HttpServletResponse from which it extends.**request**
* The standard JSP request object which is a pure HttpServletRequest.**response**
* The standard JSP response object which is a pure HttpServletResponse.

**resourceResolver**

* The current ResourceResolver object. It is the same as slingRequest.getResourceResolver()

.**sling**

* A SlingScriptHelper object, containing convenience methods for scripts, mainly sling.include('/some/other/resource') for including the responses of other resources inside this response (eg. embedding header html snippets) and sling.getService(foo.bar.Service.class) to retrieve OSGi services available in Sling (Class notation depending on scripting language).

**resource**

* the current Resource object to handle, depending on the URL of the request. It is the same as slingRequest.getResource().

**currentNode**

* If the current resource points to a JCR node (which is typically the case in Sling), this gives direct access to the Node object. Otherwise this object is not defined.

**log**

* Provides an SLF4J Logger for logging to the Sling log system from within scripts, eg. log.info("Executing my script").
* It has the following attributes:

**requestName**

**responseName**

**nodeName**

l**ogName resourceResolverName**

**slingName**

**Example:**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4** | **<%@page session="false" %><%**  **%><%@page import="com.day.cq.wcm.foundation.forms.ValidationHelper"%><%**  **%><%@taglib prefix="sling" uri="**[**http://sling.apache.org/taglibs/sling/1.0**](http://sling.apache.org/taglibs/sling/1.0)**" %><%**  **%><sling:defineObjects/>** |

**JSTL Tag library**

The [JavaServer Pages Standard Tag Library](http://www.oracle.com/technetwork/java/index-jsp-135995.html) contains a lot of useful and standard tags. The core, formatting and functions taglibs are defined by the **/libs/foundation/global.jsp** as shown in the following snippet.

**Extract of /libs/foundation/global.jsp**

|  |  |
| --- | --- |
| **1**  **2**  **3** | **<%@taglib prefix="c" uri="**[**http://java.sun.com/jsp/jstl/core**](http://java.sun.com/jsp/jstl/core)**" %>**  **<%@taglib prefix="fmt" uri="**[**http://java.sun.com/jsp/jstl/fmt**](http://java.sun.com/jsp/jstl/fmt)**" %>**  **<%@taglib prefix="fn" uri="**[**http://java.sun.com/jsp/jstl/functions**](http://java.sun.com/jsp/jstl/functions)**" %>** |

After importing the **/libs/foundation/global.jsp** file as described before, you can use the **c**, **fmt** and **fn** prefixes to access to those taglibs. The official documentation of the JSTL is available at [The Java EE 5 Tutorial - JavaServer Pages Standard Tag Library](http://docs.oracle.com/javaee/5/tutorial/doc/bnakc.html).