BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Portal JSP Tags

The WebLogic Workshop Portal extensions provide the following JSP tags.

User/Group Management

Content Management

Interaction Management

Property Sets

Internationalization and Localization

Portal Rendering

Portlet Preferences

Multichannel

Entitlements

Commerce

Utilities

User/Group Management

The following JSP tags are contained in ugm_taglib.jar and profile_taglib.jar.

ugm:addGroupToGroup	Adds a group to a group.
ugm:addUserToGroup	Adds a user to a group.
ugm:createGroup	Creates a group.
ugm:createUser	Creates a user.
ugm:getChildGroupNames	Gets the names of a group's child groups.
ugm:getGroupNamesForUser	Gets the names of groups to which a user belongs.
ugm:getParentGroupName	Gets the name of a group's parent group.
ugm:getTopLevelGroups	Gets the names of all top−level groups.
ugm:getUserNames	Gets user names based on a search expression.
ugm:getUserNamesForGroup	Gets user names within a specific group based on a search expression.
ugm:login	Evaluates the username and password in the HTTP request to provide authentication and sets the authenticated user as the current WebLogic user.
ugm:logout	Ends the current user's WebLogic Server session.
ugm:removeGroup	Deletes a group.

ugm:removeGroupFromGroup	Removes a group from its parent group. Does not delete the group.
ugm:removeUser	Deletes a user from the system.
ugm:removeUserFromGroup	Removes a user from a specified group. Does not delete the user.
ugm:setPassword	Updates a user's password.
profile:createProfile	Creates a profile for a user or group.
profile:getProfile	Gets a specific user or group profile.
profile:getProperty	Gets the value of a property as an Object.
profile:getPropertyAsString	Gets the value of a property as a String.
profile:removeProperty	Removes the specified property from the current session's profile or from the Anonymous User Profile.
profile:setProperty	Updates a property value for either the session's current profile or for the Anonymous User Profile.

Content Management

The following JSP tags are contained in content.jar.

cm:getNode	Retrieves a content node based on an explicit path and stores the node in a variable.
cm:getProperty	Retrieves a property value from a content node and stores it as a variable or prints it in the JSP.
cm:search	Searches for and retrieves content nodes based on a supplied query and stores the results in a variable.

Interaction Management

The following JSP tags are contained in ad_taglib.jar, ph_taglib.jar, pz_taglib.jar, and tracking_taglib.jar.

ad:adTarget	Uses the Ad Service to send an ad query to the content management system.
ad:render	Renders a content node from the Virtual Content Repository in the current JSP.
ph:placeholder	Implements a placeholder defined with the WebLogic Workshop Portal Extensions. Displays personalized Web content in a JSP based on placeholder and Campaign rules and queries defined with the WebLogic Workshop Portal Extensions.
pz:contentQuery	Performs a content attribute search in a content management system and returns an array of content objects.
pz:contentSelector	

	Implements a Content Selector defined with the WebLogic Workshop Portal Extensions. Displays personalized Web content in a JSP based on Content Selector rules and queries defined with the WebLogic Workshop Portal Extensions.
pz:div	Allows a piece of in−line content to be shown if a user belongs to the specific User Segment defined with the WebLogic Workshop Portal Extensions.
BehaviorTracking:clickContentEvent	Generate a behavior event when a user has clicked an image or link. This tag will return a URL query string containing event parameters, which is then used to form the complete URL that hyperlinks the content.
BehaviorTracking:displayContentEvent	Generates a behavior event when a user has received (viewed) any HTML content, such as images, text, PDF files, and Web−compatible multimedia content.

Property Sets

The following JSP tags are contained in ps_taglib.jar.

ps:getPropertyNames	Gets a list of property names in a property set.
ps:getPropertySetNames	Gets a list of property sets for a property set type.
ps:getRestrictedPropertyValues	Gets the pick list of a property's restricted values.

Internationalization and Localization

The following JSP tags are contained in i18n_taglib.jar and l10n_taglib.jar.

i18n:getMessage	Retrieves and displays localized static text or messages from a JspMessageBundle.
i18n:localize	Defines the language, country, variant, and base bundle name to be used throughout a page when accessing resource bundles with the i18n:getmessage tag.
l10n:include	Includes a localized version of a page.
l10n:forward	Forwards to a localized version of a page.
l10n:resolve	Resolves to a localized version of a page.

Portal Rendering

The following JSP tags are contained in render_taglib.jar.

render:beginRender	Used in the portal skeletons for rendering a portal resource. This tag defines the opening HTML tag for a resource.
render:endRender	Used in the portal skeletons for rendering a portal resource. This tag defines the closing HTML tag for a resource.
render:renderChild	Used in the portal skeletons for rendering a portal resource. This tag is used to render portlet titlebars, titlebar buttons, menus (navigation such as page tabs), and table cells within a layout.
render:jspContentUrl	lets you create URLs to windows and set content of those windows.
render:pageUrl	Lets you create links to switch to a page or a book based on labels.
render:standalonePortletUrl	Lets you create URLs to floatable portlets. Use this tag to create links to submit requests to portlets hosted by an external portal, such as floating portlets.
render:postbackUrl	Lets you create URLs to submit GET/POST requests to the portal framework. This tag is necessary because all portlet requests first need to go through the portal, before being delegated to the caller.
render:resourceUrl	Represents a URL to a resource. Resource URLs are typically used for static/dynamic resources such as files or images. Resource URLs cannot be used to submit requests to the portal framework.
render:windowUrl	Lets you create links to windows based on labels and switch a window to a mode, state, or both.
render:toggleButtonUrl	Used in the portal skeletons for rendering a portal resource. This tag lets you build URLs for the toggle buttons in portlet titlebars.
render:param	This tag lets you append query parameters to the URL tags.
render:jspUri	Retrieves the location of the JSP in which this tag is used.
render:getJspUri	Retrieves the location of the JSP in which the tag is used. This helps create correct relative URLs to resources such as images within a JSP. The URL omits the JSP filename.
render:createUri	Creates a URI based on the application's configured skin path.
render:writeUri	Writes the URI based on the application's configured skin path.
render:getSkinPath	

	Returns the path of an image used in the current skin. Every Look & Feel can have a different directory structure as well as different client classifications and localization; this tag resolves the location based on just the file name.
render:createId	Creates an ID for a rendered component.
render:writeId	Writes an ID for a rendered component.
render:writeAttribute	Used in the portal skeletons for rendering a portal resource. This tag sets HTML attributes for a tag.
render:encodeName	Lets you encode names such as HTML form names and JavaScript function names to make them unique such that the encoded names are unique within the context of the a portal page.

Portlet Preferences

The following JSP tags are contained in prefs_taglib.jar.

pref:getPreference	Gets the default value of a portlet preference or sets a default value.
pref:getPreferences	Gets all the values of a given portlet preference or supplies default value.
pref:forEachPreference	Iterates over all the preferences available for a portlet and stores the results in a variable.
pref:ifModifiable	Includes the body of the tag if the given portlet preference is modifiable.
pref:else	Includes the body of the tag if the given portlet preference is not modifiable.

Multichannel

The following JSP tags are contained in client_taglib.jar.

cscm:default	Renders its contents only if the client's classification has been mapped to "default" or if the client is not recognized.
cscm:not−default	Renders its content only if the client classification has been mapped to anything other than "default."
cscm:recognized	Renders its content if the client has been mapped to any classification, even "default."
cscm:not−recognized	Renders its content if the client has not been mapped to any classification.
cscm:when	Renders its content for any client classification listed in this tag's "client" attribute that matches the client classification name in the HTTP request.
cscm:when−not	Renders its content for any client classification not listed

	in this tag's "client" attribute. The client name comes from the HTTP request.

Entitlements

The following JSP tags are contained in auth_taglib.jar.

auth:isAccessAllowed	Provides fine−grained entitlement−setting on application resources for which entitlements are not available by default.
auth:isUserInRole	

Commerce

The following JSP tags are contained in cat_taglib.jar, productTracking_taglib.jar, and eb_taglib.jar.

cat:catalogQuery	Retrieves a set of catalog items based on keywords or a query expression.
cat:catalogSelector	Retrieves a personalized set of catalog items based on a predefined content selector rule.
cat:getProperty	Retrieves a property for display from either a ProductItem or Category.
cat:iterateThroughView	Iterates through a view of one Catalog item at a time until the end of the View is reached.
cat:iterateViewIterator	Iterates through a collection of Categories or ProductItems.
productTracking:clickProductEvent	Generates a behavior event when a user has clicked (through) a product impression.
productTracking:displayProductEvent	Generates a behavior event when a user has received (viewed) a product impression, typically an image.
eb:smnav	Scrollable model navigation. For use with legacy applications that implement Webflow and run in a compatibility domain. Creates "Previous | 10−19 | Next" style navigation for pages with multiple results.

Utilities

The following JSP tags are contained in es_taglib.jar.

es:convertSpecialChars	Lets you display special characters as literal strings for display in a browser.
es:counter	Performs Java `for` loop logic.

es:date	Gets a date− and time−formatted String based on the user's time zone preference.
es:forEachInArray	Iterates over an array of returned objects.
es:isNull	Checks if a value is null. For String values, checks to see if the value is empty (zero length).
es:notNull	Checks if a value is not null. For String values, checks to see if the value is greater than zero length.
es:transposeArray	Transposes a standard [row][column] array to a [column][row] array.
es:uriContent	Pulls content from a specified URL.

Related Topics

Using Portal JSP Tags

Creating a JSP Portlet

User/Group Management JSP Tags

Following are the Portal User/Group Management JSP Tags. They are contained in ugm_taglib.jar and profile_taglib.jar.

ugm:addGroupToGroup	Adds a group to a group.
ugm:addUserToGroup	Adds a user to a group.
ugm:createGroup	Creates a group.
ugm:createUser	Creates a user.
ugm:getChildGroupNames	Gets the names of a group's child groups.
ugm:getGroupNamesForUser	Gets the names of groups to which a user belongs.
ugm:getParentGroupName	Gets the name of a group's parent group.
ugm:getTopLevelGroups	Gets the names of all top−level groups.
ugm:getUserNames	Gets user names based on a search expression.
ugm:getUserNamesForGroup	Gets user names within a specific group based on a search expression.
ugm:login	Evaluates the username and password in the HTTP request to provide authentication and sets the authenticated user as the current WebLogic user.
ugm:logout	Ends the current user's WebLogic Server session.
ugm:removeGroup	Deletes a group.
ugm:removeGroupFromGroup	Removes a group from its parent group. Does not delete the group.
ugm:removeUser	Deletes a user from the system.
ugm:removeUserFromGroup	Removes a user from a specified group. Does not delete the user.
ugm:setPassword	Updates a user's password.
profile:createProfile	Creates a profile for a user or group.
profile:getProfile	Gets a specific user or group profile.
profile:getProperty	Gets the value of a property as an Object.
profile:getPropertyAsString	Gets the value of a property as a String.
profile:removeProperty	Removes the specified property from the current session's profile or from the Anonymous User Profile.
profile:setProperty	Updates a property value for either the session's current profile or for the Anonymous User Profile.

<ugm:addGroupToGroup> Tag

The <ugm:addGroupToGroup> tag adds the group corresponding to the provided childGroupName to the group corresponding to the provided groupName. Since a group can only have one parent, any previous relationship with another parent will be destroyed. Both the parent group and the child group must previously exist for proper tag behavior. The tag has no enclosed body.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

childGroupName

Required (String) − The name of the child group. Example: "<%=childGroupName%>"

parentGroupName

Optional (String) − The name of the parent group. Example: "<%=parentGroupName%>"

result

Required (String) − The name of an Integer variable to which the result of the add group to group operation is assigned.

Possible values:

- Success: UserManagementTagConstants.ADD_GROUP_OK.
- Error encountered: UserManagementTagConstants.ADD_GROUP_FAILED

Example

This example shows how to use <ugm:addGroupToGroup> to add a new group of users to an existing group of users.

```
<ugm:addGroupToGroup childGroupName="<%=childGroupName%>"
```

```
parentGroupName="<%=parentGroupName%>"
result="result"/>
```

Related Topics

<ugm:addUserToGroup> Tag

<ugm:addUserToGroup> Tag

The <ugm:addUserToGroup> tag adds the user corresponding to the provided username to the group corresponding to the provided groupName. Both the specified user and the specified group must previously exist for proper tag behavior. The tag has no enclosed body.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

username

Required (String) − The name of the user to be added to the group. Example: "<%=username%>"

groupName

Required (String) − The name of the group to which the user is being added. Example: "<%=groupName%>"

result

Required (String) − The name of an Integer variable to which the result of the add user to group operation is assigned.

Possible values:

- Success: UserManagementTagConstants.ADD_USER_OK.
- Error encountered: UserManagementTagConstants.ADD_USER_FAILED

Example

This example shows how to use <ugm:addUserToGroup> to add a new user to an existing group.

```
<ugm:addUserToGroup userName="<%=userName%>"
groupName="<%=groupName%>"
result="<%=result%>"/>
```

Related Topics

<ugm:addGroupToGroup> Tag

<ugm:createGroup> Tag

The <ugm:createGroup> tag reates a new group in the realm, and a corresponding group profile in the personalization database. This tag has no enclosed body.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

After the group is created, use the <profile:createProfile> tag to create a profile for the new group.

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

groupName

Required (String) − The name of the new group. Example: "<%=groupName%>"

id

Optional (String) − A variable name to which the created Group object is available for the duration of the page's scope.

parentName

Optional (String) − The name of the group to set as the parent of the new group.

result

Required (String) − The name of an Integer variable to which the result of the create group operation is assigned.

Possible Values:

- Success: UserManagementTagConstants.CREATE_GROUP_OK.
- Error encountered: UserManagementTagConstants.CREATE_GROUP_FAILED.
- A group with the specified group name already exists: UserManagementTagConstants.GROUP_EXISTS.

Example

This example shows how to use <ugm:creategroup> to create a new group.

```
<ugm:creategroup groupName="<%=groupName%>" result="result"/>
```

Related Topics

<ugm:createUser> Tag

<ugm:createUser> Tag

The <ugm:createUser> tag creates a new user.

> ***Note***: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Security: In the WebLogic Administration Portal's Service Administration tool, you can determine which roles can perform this action.

After the user is created, use the <profile:createProfile> tag to create a user profile for the new user. If you use the doPostProcess attribute to perform post−user−creation processing (the default), a profile is created automatically.

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

doPostProcess

Optional (Boolean) − Works in conjunction with the fireEvent, login, and saveAnonymous attributes. If this attribute and the fireEvent, login, and saveAnonymous attributes are set to true, all three events−−including the creation of a user profile−−occur after the user is created. If the doPostProcess attribute is set to false, the individual settings on the other three attributes are used.

fireEvent

Optional (Boolean) − If set to true, a UserRegistrationEvent is dispatched to the event service during the post−user−creation process. Defaults to true. If this attribute is set to false, doPostProcess is ignored.

login

Optional (Boolean) − If set to true, the user is logged in during the post−user−creation process. Defaults to true. If true, a user profile is created automatically for the user at login. If this attribute is set to false, doPostProcess is ignored, and you must either create a profile for the user or have a profile created for the user automatically when the user next logs in.

saveAnonymous

Optional (Boolean) − If set to true, any properties the user may have set during the session before registering are added to the new user's properties during the post−user−creation process. Defaults to true. If this attribute is set to false, doPostProcess is ignored.

password

Required (String) − The password for the new user. Example: "<%=password%>"

username

Required (String) − The name of the new user. Example: "<%=username%>"

result

Required (String) − The name of an Integer variable to which the result of the create user operation is assigned.

Possible values:

- Success: UserManagementTagConstants.CREATE_USER_OK.
- Error encountered: UserManagementTagConstants.CREATE_USER_FAILED.
- A user with the specified username already exists: UserManagementTagConstants.USER_EXISTS.

Example

This example shows how to use <ugm:createUser> to create a new user.

```
<ugm:createUser userName="<%=username%>" password="<%=password%>" result="result"/>
```

Related Topics

<ugm:createGroup> Tag

<ugm:getChildGroupNames> Tag

The <ugm:getChildGroupNames> tag returns the names of any groups that are children of the given group.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

groupName

Required (String) − The name of the group to search for child groups.

id

Required (String) − The name of the identifier which will be assigned the String array of child group names. If there are no children, the array will be zero length.

Example

This example shows how to use <ugm:getChildGroupNames> to retrieve the names of a child group for the group SomeGroup.

```
<ugm:getChildGroupNames groupName="SomeGroup" id="childrenOfSomeGroup" />
```

Related Topics

<ugm:getGroupNamesForUser> Tag

<ugm:getParentGroupName> Tag

<ugm:getTopLevelGroups> Tag

<ugm:getGroupNamesForUser> Tag

The <ugm:getGroupNamesForUser> tag retrieves a String array that contains the group names corresponding to groups to which the provided user immediately belongs. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

username

Required (String) − The name of the user whose matching groups are sought. Example: "<%=username%>"

id

Required (String) − A variable name to which the resultant group names are assigned. Example: "myGroups"

Example

This example shows how to use <ugm:getGroupNamesForUser> to retrieve a group name to apply to a user.

```
<ugm:getGroupNamesForUser userName="<%=username%>" id="myGroups"/>
```

Related Topics

<ugm:getChildGroupNames> Tag

<ugm:getParentGroupName> Tag

<ugm:getTopLevelGroups> Tag

<ugm:getParentGroupName> Tag

The <ugm:getParentGroupName> tag retrieves the name of the parent of the group associated with the provided groupName. The information is taken from the realm. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

groupName

Required (String) − The name of the group whose parent group name is sought. Example: "<%=groupName%>"

id

Required (String) − A variable name to which the name of the parent is available for the duration of the page's scope. Example: "parentGroupName". If there is no parent, the variable value is null.

Example

This example shows how to use <ugm:getParentGroupName> to retrieve a parent group name.

```
<ugm:getParentGroupName groupName="<%=groupName%>" id="parentGroupName"/>
```

Related Topics

<ugm:getChildGroupNames> Tag

<ugm:getGroupNamesForUser> Tag

<ugm:getTopLevelGroups> Tag

<ugm:getTopLevelGroups> Tag

The <ugm:getTopLevelGroups> tag retrieves an array of group names, each of which has no parent group. The information is taken from the realm. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

id

Required (String) − A variable name to which the top−level Group names, stored as a String, are available for the duration of the page's scope. Example: "topLevelGroups".

Example

```
<ugm:getTopLevelGroups id="topLevelGroups"/>
```

Related Topics

<ugm:getChildGroupNames> Tag

<ugm:getGroupNamesForUser> Tag

<ugm:getParentGroupName> Tag

<ugm:getUserNames> Tag

The <ugm:getUsernames> tag retrieves a String array that contains the usernames matching the provided search expression. The search expression supports only the asterisk (*) wildcard character, and is case insensitive. As many asterisks as desired may be used in the search expression. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

searchExp

Optional (String) − The search expression to apply to the user name search. Defaults to `*'. Example: "t*".

userLimit

Optional (String, representing an int) − The maximum number of users to be returned from the search. (String which has a particular Integer.valueOf.) Defaults to 100. If user count exceeds userLimit, the length of the array in id is truncated to the length of userLimit. Example: "500".

id

Required (String) − A variable name to which the resultant user names are assigned. Example: "myUsers".

result

Optional (String) − The name of an Integer variable to which the result of the getUsernames operation is assigned.

Possible values:

- Success: UserManagerTagConstants.USER_SEARCH_OK.
- General error (such as a database connection failure that occurred during the search): UserManagerTagConstants.
  USER_SEARCH_FAILED.

If no users match the search criteria, then the result will not be equal to UserManagerTagConstants.USER_SEARCH_FAILED, but the length of the array returned in "id" will be zero.

Note: The USER_SEARCH_FAILED value is returned only when a general error occurs while searching for the user, such as a database connection failure. If no user matches the search criteria, the result will not be equal to UserManagementTagConstants.USER_SEARCH_FAILED, but the length returned by the array in id will be zero.

Example

This example shows how to use <ugm:getUsernames> to retrieve a and print a username.

```
<ugm:getUsernames userLimit="500" searchExp="t*" id="myUsers"/>
<%System.out.println("I found " + myUsers.length + " users.");%>
```

Related Topics

<ugm:getUserNamesForGroup> Tag

<ugm:getUserNamesForGroup> Tag

The <ugm:getUsernamesForGroup> tag retrieves a String array that contains the usernames matching the provided search expression and correspond to members of the provided group. The search expression supports only the asterisk (*) wildcard character, and is case insensitive. As many asterisks as desired may be used in the search expression. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

searchExp

Optional (String) − The search expression to apply to the user name search. Defaults to " *". Example: "t*".

groupName

Required (String) − The name of the group whose matching members are sought. Example: "engineering".

userLimit

Optional (String, representing an int) − The maximum number of users to be returned from the search. (String which has a particular Integer.valueOf.) Defaults to 100. If user count exceeds userLimit, the length of the array in id is truncated to the length of userLimit. Example: "500".

id

Required (String) − A variable name to which the resultant user names are assigned. Example: "myUsers".

result

Optional (String) − The name of an Integer variable to which the result of the get usernames for group operation is assigned.

Possible values:

- Success: UserManagementTagConstants.USER_SEARCH_OK.

- General error: UserManagementTagConstants.USER_SEARCH_FAILED.

Note: The USER_SEARCH_FAILED value is returned only when a general error occurs while searching for the user, such as a database connection failure. If no user matches the search criteria, the result will not be equal to UserManagementTagConstants.USER_SEARCH_FAILED, but the length returned by the array in id will be zero.

Example

This example shows shows how to use <ugm:getUsernamesForGroup> to retrieve and display usernames that match the provided search expression and correspond to members of the provided group.

```
<ugm:getUsernamesForGroup groupName="engineering" userLimit="500" searchExp="t*" id="myUsers"/>
<%System.out.println("I found " + myUsers.length + " users in my group.");%>
```

Related Topics

<ugm:getUserNames> Tag

<ugm:login> Tag

The <ugm:login> tag provides weak authentication (username, password) against the current security realm, and sets the authenticated user as the current WebLogic user. This tag has no enclosed body.

The login tag requires a username parameter and a password parameter to be present in the HTTP request.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

result

Required (String) − The name of an Integer variable to which the result of the login operation is assigned.

Possible values:

- Success: UserManagementTagConstants.LOGIN_OK.
- General error when performing authentication: UserManagementTagConstants.LOGIN_ERROR.
- Authentication failed because of invalid username/password combination: UserManagementTagConstants.LOGIN_FAILED.

Example

```
<ugm:login result="result" />
```

This tag is also used in the Login to Portal Portlet sample.

Related Topics

<ugm:logout> Tag

<ugm:logout> Tag

The <ugm:logout> tag ends the current user's WebLogic Server session. This tag should be used in combination with the <ugm:login> tag.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

None.

Example

```
<ugm:logout />
```

This tag is also used in the Login to Portal Portlet sample.

Related Topics

<ugm:login> Tag

<ugm:removeGroup> Tag

The <ugm:removeGroup> tag removes the group corresponding to the provided groupName. This tag has no enclosed body.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

groupName

Required (String) − The name of the group to be removed. Example: "<%=groupName%>".

result

Required (String) − The name of an Integer variable to which the result of the remove group operation is assigned.

Possible Values:

- Success: UserManagementTagConstants.REMOVE_GROUP_OK.
- Error encountered: UserManagementTagConstants.REMOVE_GROUP_FAILED.

Example

This example shows how to use <ugm:removeGroup> to remove a group that corresponds to the provided groupName.

```
<ugm:removeGroup groupName="<%=groupName%>" result="result"/>
```

Related Topics

<ugm:removeGroupFromGroup> Tag

<ugm:removeUserFromGroup> Tag

<ugm:removeUser> Tag

<ugm:removeGroupFromGroup> Tag

The <ugm:removeGroupFromGroup> tag removes a child group from a parent group.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

childGroupName

Required (String) − The name of the child group to remove from its parent.

parentGroupName

Required (String) − The name of the parent group from which the child group will be removed.

result

Required (String) − The name of an Integer variable to which the result of the remove group from group operation is assigned.

Possible values:

- Success: UserManagementTagConstants.REMOVE_GROUP_OK.
- Failure: UserManagementTagConstants.REMOVE_GROUP_FAILED.

Example

This example shows how to use <ugm:removeGroupFromGroup> to remove a child group from a parent group.

```
<ugm:removeGroupFromGroup childGroupName="<%=salesteam%>" parentGroupName="<%=humanresources%>
result="result" />
```

Related Topics

<ugm:removeUser> Tag

The <ugm:removeUser> tag removes the user corresponding to the provided username. It can remove any type of extended user that has its profileType set in the database. This tag has no enclosed body.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

username

Required (String) − The username of the user to be removed. Example: "<%=username%>".

result

Required (String) − The name of an Integer variable to which the result of the remove user operation is assigned.

Possible values:

- Success: UserManagementTagConstants.REMOVE_USER_OK.
- Error encountered: UserManagementTagConstants.REMOVE_USER_FAILED.

Example

This example shows how to use <ugm:removeUser> to remove a user.

```
<ugm:removeUser username="<%=username%>" result="result"/>
```

Related Topics

<ugm:removeUserFromGroup> Tag

<ugm:removeGroupFromGroup> Tag

<ugm:removeGroup> Tag

<ugm:removeUserFromGroup> Tag

The <ugm:removeUserFromGroup> tag removes a user from a group.

Execution of this tag requires the logged in user to have administrator rights.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

username

Required (String) − The username of the user to remove from the given group.

groupName

Required (String) − The name of the group from which the given user will be removed.

result

Required (String) − The name of an Integer variable to which the result of the remove user from group operation is assigned.

Possible values:

- Success: UserManagementTagConstants.REMOVE_USER_OK.
- Failure: UserManagementTagConstants.REMOVE_USER_FAILED.

Example

This example shows how to use <ugm:removeUserFromGroup> to remove a specified user from a specified group.

```
<ugm:removeUserFromGroup username="UserToRemove" groupName="SomeGroup" result="myResult" />
```

Related Topics

<ugm:setPassword> Tag

The <ugm:setPassword> tag updates the password for the user corresponding to the provided username. Users must be logged in to change their passwords.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

atnProvider

Optional (String) − The name of the authentication provider to use. The default setting is the WebLogic Server DefaultAuthenticator. The authentication provider must provide write access. See Using Multiple Authentication Providers in Portal Development for more information.

username

Required (String) − The username of the user whose password is to be changed.

password

Required (String) − The new user password.

result

Required (Integer) − The name of an Integer variable to which the result of the set password operation is assigned.

Possible values:

- Success: UserManagementTagConstants.SET_PASSWORD_OK.
- Failure: UserManagementTagConstants.SET_PASSWORD_FAILED.

Example

The following example resets a user's password.

```
<ugm:setPassword username="<%userName>" password="<%password>" result="<%result>" />
```

<profile:createProfile> Tag

The <profile:createProfile> tag creates a new user or group profile, along with any designated successors for property inheritance. Use this tag in conjunction with the <ugm:createUser> tag to create a profile for a new user.

You can also create a profile for a user or group in an authentication provider that does not allow read access. When the user logs in, the user is paired with the profile.

Note: It is possible (but not recommended) to store an identical username or group name in more than one authentication provider. For example, user "foo" can reside in the default WebLogic Server LDAP provider and in an external RDBMS provider. In that case, WebLogic Portal uses only one user profile for user "foo."

Syntax

<tagName *attribute*="*value*" />

Attributes

profileKey

Required (String) − The key (name of a user or group) for the profile to create. Example: "<%=username%>".

successorKey

Optional (String) − The key for the profile's successor. Example: "<%=defaultGroup%>".

scope

Optional (String) − The HTTP scope of the new profile. Pass "request" or "session" as the values. Defaults to session.

groupOnly

Optional (Boolean) − When set to true, a group profile named by the profileKey is created rather than a user profile. No successor will be created when this value is true. Defaults to false.

profileId

Optional (String) − A variable name from which the new profile is available, as a com.bea.p13n.usermgmt.profile.ProfileWrapper.

successorId

Optional (String) − The ID of the profile's successor, as a com.bea.p13n.usermgmt.profile.ProfileWrapper.

result

Optional (Integer) − A variable name for the result of the operation.

Possible values:

- ProfileManagementTag.CREATE_PROFILE_OK
- ProfileManagementTag.CREATE_PROFILE_FAILED

Example

```
<profile:createProfile profileKey="<%=username%>" profileId="profileId" />
```

Related Topics

<ugm:createUser> Tag

<profile:getProfile> Tag

<profile:getProfile> Tag

The <profile:getProfile> tag retrieves the profile corresponding to the provided profile key and profile type. The tag has no enclosed body. The retrieved profile can be treated as a com.bea.p13n.usermgmt.profile.ProfileWrapper. Along with the profile key and profile, an explicit successor key and successor type can be specified, as specified by the profileType attribute. This successor will then be used, along with the retrieved profile, in subsequent invocations of the <profile:getProperty> tag to ensure property inheritance from the successor. If no successor is retrieved, standard ConfigurableEntity successor search patterns will apply to retrieved properties.

Syntax

<tagName *attribute*="*value*" />

Attributes

profileKey

Required (String) − A unique identifier that can be used to retrieve the profile which is sought. Example: "<%=username%>".

successorKey

Optional (String) − A unique identifier that can be used to retrieve the profile successor. Example: "<%=defaultGroup%>".

scope

Optional (String) − The HTTP scope of the retrieved profile. Pass "request" or "session" as the values. Defaults to session.

groupOnly

Optional (String) − Specifies to retrieve a group profile named by the profileKey, rather than a user profile. No successor will be retrieved when this value is true. Defaults to false.

profileId

Optional (String) − A variable name from which the retrieved profile is available for the duration of the JSP's page scope.

successorId

Optional (String) − A variable name from which the retrieved successor is available for the duration of the JSP's page scope.

result

Optional (String) − A variable name from which the result of the operation is available.

Possible values:

- Success: UserManagementTagConstants.GET_PROFILE_OK.
- Error encountered:
  - ♦ UserManagementTagConstants.GET_PROFILE_FAILED.
  - ♦ UserManagementTagConstants.NO_SUCH_PROFILE.
  - ♦ UserManagementTagConstants.NO_SUCH_SUCCESSOR.

Example

```
<%@ page import="com.bea.p13n.usermgmt.servlets.jsp.taglib.UserManagementTagConstants" %>
<%@ page import="com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants" %>

<% String pname = request.getPrincipalName(); %>
<profile:getProfile profileKey="<%=pname%>" result="rc"/>

<% if (rc.intValue() == UserManagementTagConstants.GET_PROFILE_OK)
out.println("GET_PROFILE_OK");
else if (rc.intValue() ==
UserManagementTagConstants.GET_PROFILE_FAILED)
out.println("GET_PROFILE_FAILED");
else if (rc.intValue()==UserManagementTagConstants.NO_SUCH_PROFILE)
out.println("NO_SUCH_PROFILE");
%>

<profile:getProperty propertySet="ldap" propertyName="cn" id="myProperty" />
<% if (myProperty.toString().equalsIgnoreCase("roy hobbs")) {
out.println(myProperty);
}
else {
out.println(myProperty + " failed here");
}
%>
```

Related Topics

<profile:getProperty> Tag

<profile:getPropertyAsString> Tag

<profile:getProperty> Tag

The <profile:getProperty> tag retrieves the property value for a specified property set−property name pair. The tag has no enclosed body. The value returned is an Object. In typical cases, this tag is used after the <profile:getProfile> tag is invoked to retrieve a profile for session use. The property to be retrieved is retrieved from the session profile. If the <profile:getProfile> tag has not been used upon invoking the <profile:getProperty> tag, the specified property value is retrieved from the Anonymous User Profile.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt.servlets.jsp.tags.UserManagementTag−Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySet

Optional (String) − The Property Set from which the property's value is to be retrieved. Example: "Demographics". If no property set is provided, the property is retrieved from the profile's default (unscoped) properties.

propertyName

Required (String) − The name of the property to be retrieved. Example: "Date_of_Birth".

id

Optional (String) − If the id attribute is supplied, the value of the retrieved property will be available in the variable name to which id is assigned. Otherwise, the value of the property is inlined.

Example

```
<%@ page import="com.bea.p13n.usermgmt.servlets.jsp.taglib.UserManagementTagConstants" %>
<%@ page import="com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants" %>

<% String pname = request.getPrincipalName(); %>
<profile:getProfile profileKey="<%=pname%>" result="rc"/>

<% if (rc.intValue() == UserManagementTagConstants.GET_PROFILE_OK)
out.println("GET_PROFILE_OK");
else if (rc.intValue() ==
UserManagementTagConstants.GET_PROFILE_FAILED)
out.println("GET_PROFILE_FAILED");
else if (rc.intValue()==UserManagementTagConstants.NO_SUCH_PROFILE)
out.println("NO_SUCH_PROFILE");
%>

<profile:getProperty propertySet="ldap" propertyName="cn" id="myProperty" />
```

```
<% if (myProperty.toString().equalsIgnoreCase("roy hobbs")) {
out.println(myProperty);
}
else {
out.println(myProperty + " failed here");
}
%>
```

Related Topics

<profile:getProfile> Tag

<profile:getPropertyAsString> Tag

<profile:getPropertyAsString> Tag

The <profile:getPropertyAsString> tag retrieves the property value for a specified property set−property name pair. The tag has no enclosed body. The value returned is a String. In typical cases, this tag is used after the <profile:getProfile> tag is invoked to retrieve a profile for session use. The property to be retrieved is retrieved from the session profile. If the <profile:getProfile> tag has not been used upon invoking the <profile:getPropertyAsString> tag, the specified property value is retrieved from the Anonymous User Profile.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySet

Optional (String) − The Property Set from which the property's value is to be retrieved. Example: "Demographics". If no property set is provided, the property is retrieved from the profile's default (unscoped) properties.

propertyName

Required (String) − The name of the property to be retrieved. Example: "Date_of_Birth".

id

Optional (String) − If the id attribute is supplied, the value of the retrieved property will be available in the variable name to which id is assigned. Otherwise, the value of the property is inlined.

Example

This example shows how to retrieve a String of information from a user profile.

```
<profile:getPropertyAsString
id="myBirthDate"
propertySet="Demographics"
propertyName="Date_of_Birth"/>My birthday is <%=myBirthDate%>.
```

This tag is used in the Portal Search portlet.

Related Topics

<profile:getProperty> Tag

<profile:removeProperty> Tag

The <profile:removeProperty> tag removes the specified property from the current session's profile or from the Anonymous User Profile. Subsequent calls to getProperty for a removed property would result in the default value for the property as prescribed by the property set as defined with the WebLogic Workshop Portal Extensions.

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySet

Optional (String) − The Property Set from from which the property's value is to be removed. The property is removed from the profile's default (unscoped) properties if no property set is provided. Property Sets are defined with the WebLogic Workshop Portal Extensions.

propertyName

Required (String) − The name of the property to be removed. Properties are defined in Property Sets with the WebLogic Workshop Portal Extensions.

Example

```
<profile:removeProperty propertySet="personal" propertyName="annualIncome" />
```

Related Topics

<profile:setProperty> Tag

<ps:getPropertyNames> Tag

<ps:getPropertySetNames> Tag

<ps:getRestrictedPropertyValues> Tag

<profile:setProperty> Tag

The <profile:setProperty> tag updates a property value for either the session's current profile, or for the Anonymous User Profile. This tag has no enclosed body.

Note: All User Management tags send results to the same file. If you are checking for results, include this import statement at the top of the page: <%@ page import="com.bea.p13n.usermgmt. servlets.jsp.tags.UserManagementTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySet

Optional (String) − The Property Set in which the property's value is to be set. Example: "Demographics".

The property is set for the profile's default (unscoped) properties if no property set is provided.

propertyName

Required (String) − The name of the property to be set. Example: "Gender".

value

Required (Object or String) − The new property value.

result

Optional (String) − The name of an Integer object to which the result of the set property operation is assigned.

Possible values:

- Success: UserManagementTagConstants.SET_PROPERTY_OK.
- Error encountered: UserManagementTagConstants.SET_PROPERTY_FAILED.

Example

This example shows a that a property called "Gender" will get an updated value.

```
<% String myGender = request.getParameter("gender"); %>
<profile:setProperty propertySet="Demographics" propertyName="Gender" value="<%=myGender%>"/>
```

This tag is used in the Portal Search portlet's edit JSP.

Related Topics

Interaction Management JSP Tags

Following are the Portal Interaction Management JSP Tags. The are contained in ad_taglib.jar, ph_taglib.jar, pz_taglib.jar, and tracking_taglib.jar.

ad:adTarget	Uses the Ad Service to send an ad query to the content management system.
ad:render	Renders a content node from the Virtual Content Repository in the current JSP.
ph:placeholder	Implements a placeholder defined with the WebLogic Workshop Portal Extensions. Displays personalized Web content in a JSP based on placeholder and Campaign rules and queries defined with the WebLogic Workshop Portal Extensions.
pz:contentQuery	Performs a content attribute search in a content management system and returns an array of content objects.
pz:contentSelector	Implements a Content Selector defined with the WebLogic Workshop Portal Extensions. Displays personalized Web content in a JSP based on Content Selector rules and queries defined with the WebLogic Workshop Portal Extensions.
pz:div	Allows a piece of in−line content to be shown if a user belongs to the specific User Segment defined with the WebLogic Workshop Portal Extensions.
BehaviorTracking:clickContentEvent	Generate a behavior event when a user has clicked an image or link. This tag will return a URL query string containing event parameters, which is then used to form the complete URL that hyperlinks the content.
BehaviorTracking:displayContentEvent	Generates a behavior event when a user has received (viewed) any HTML content, such as images, text, PDF files, and Web−compatible multimedia content.

<ad:adTarget> Tag

The <ad:adTarget> uses the Ad Service to send an ad query to the content management system. Unlike the <ph:placeholder> tag, the query in the <ad:adTarget> tag does not compete with other queries in an ad placeholder.

Use this tag if you need to make sure that a given ad displays to customers in a specific location. Depending on how narrowly you construct the query, you might have to remove or modify this tag when you want to display a different ad.

If the ad query returns more than one ad, the Ad Service uses the adWeight attribute of each ad to determine which ad to display.

Syntax

<tagName *attribute*="*value*" />

Attributes

query

Required (String) − Contains a query that the Ad Service uses to find content. Use the query syntax described in the Portal Javadoc for com.bea.content.expression.ExpressionHelper.

height

Optional (int) − The optional height (in pixels) of the rendered content.

width

Optional (int) − The optional width (in pixels) of the rendered content.

Example

The following example picks one of the ads in the ad group "Car" and renders it in a space measuring 200 x 400 pixels.

```
<ad:adTarget query="group == 'Car'" width="400" height="200" />
```

Related Topics

<ph:placeholder> Tag

<ad:render> Tag

<ad:render> Tag

The <ad:render> tag renders a binary property of a content node from the Virtual Content Repository in the current JSP.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The script variable name that has the com.bea.content.Node object to render. This will be used if ad is not specified.

ad

The com.bea.content.Node object reference to render. If this is specified, id will be ignored.

height

Optional (int) − The optional height (in pixels) of the rendered content.

width

Optional (int) − The optional width (in pixels) of the rendered content.

Example

```
<cm:getNode path="/BEA Repository/graphics/hr/open_enrollment" id="node" />

<ad:render id="node" height="200" width="400" />
```

Related Topics

<ad:adTarget> Tag

<ph:placeholder> Tag

<cm:getNode> Tag

<ph:placeholder> Tag

The <ph:placeholder> tag implements a placeholder, which describes the behavior for a location on a JSP page. You use the placeholder designer to define a placeholder.

Multiple placeholder tags can refer to the same placeholder. Each instance of a placeholder tag invokes its placeholder definition separately. If the placeholder definition specifies multiple queries, each placeholder tag instance can display different ads, even though each instance shares the same definition.

When WebLogic Portal receives a request for a JSP that contains an ad placeholder, the placeholder tag contacts the Ad Service, a session EJB that invokes business logic to determine which ad to display.

Syntax

<tagName *attribute*="*value*" />

Attributes

name

Required (String) − A string that refers to a placeholder definition.

height

Optional (int) − Specifies the height (in pixels) that the placeholder uses when generating the HTML that the browser requires to display a document. The placeholder uses this value only for content types to which display dimensions apply and only if other attributes have not already defined dimensions for a given document. If you do not specify this value and other attributes have not already been defined, the browser behavior determines the height of the document.

width

Optional (int) − Specifies the width (in pixels) that the placeholder uses when generating the HTML that the browser requires to display a document. The placeholder uses this value only for content types to which display dimensions apply and only if other attributes have not already defined dimensions for a given document. If you do not specify this value and other attributes have not already been defined, the browser behavior determines the height of the document.

Example

This example displays the ad specified in the MainPageBanner placeholder.

```
<ph:placeholder name="/placeholders/MainPageBanner.pla"/>
```

Related Topics

Creating Placeholders

Creating Campaigns

<ad:adTarget> Tag

<pz:contentQuery> Tag

The <pz:contentQuery> tag performs a content attribute search for content in the Repository Manager, returning an array of Node objects from the Repository Manager. If the useCache attribute is set to true, the results of a content management query will be cached. The tag only has a begin tag and does not have a body or end tag.

Use the following tags in conjunction with <pz:*> tags to access and display the returned content:

- <es:forEachInArray> to iterate over the Node objects in the array.
- <cm:*> tags to access metadata attributes in the content, retrieve content, and put it into the HTTP response output stream.

Syntax

<tagName *attribute*="*value*" />

Attributes

max

Optional (String, long) − Limits the maximum number of content nodes returned. If not present, it returns all of the content nodes found.

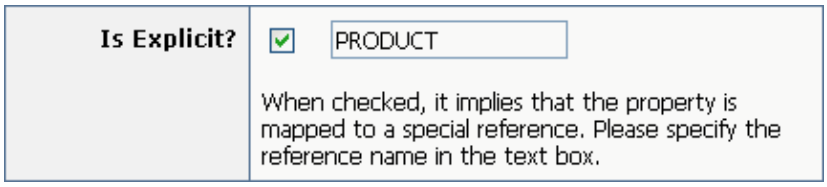
sortBy

Optional (String) − Use this attribute to sort the retrieved content in ascending or descending order. Enter the content attributes on which you want to sort the content.

However, you can sort content only on content properties that are explicit, meaning the values are stored in their own database column. System properties (see com.bea.content.expression.Search) are explicit by default. But the content properties you create in the WebLogic Administration portal (properties contained within the content "types" you create) are implicit. If you want to sort on those properties, you must make them explicit by doing the following:

1. In your database, create a column in CM_NODE table of type Integer to store the explicit property values you want to sort on. Name the column whatever you like (within the naming contraints).
2. In the WebLogic Administration Portal, set the property you want to sort on to be explicit, and enter the name of the database column you created.

   For example, if you have a property called "myproducts," and you created a database column called "PRODUCT" for it, the following is how you would define the "myproducts" property: select the "Is Explicit?" option and enter "PRODUCT" as the special reference.



Any content nodes that use this property will have the property value stored in the PRODUCT database column, and the retrieved content can be sorted by that property.

Use ASC and DESC to sort retrieved content in ascending or descending order.

Examples:

sortBy= myproducts ASC

sortBy="cm_modifiedDate DESC" (This is a system attribute that is explicit by default.)

query

Required (String) − A content query string used to search for content.

Example:

query= "cm_contentType contains `text' && author=`Proulx'".

id

Required (String) − The array variable name that contains the Node objects found. If no content is found, the variable is assigned an empty array (not null) of Node objects.

useCache

Optional (String, Boolean) − Determines whether Nodes are cached. This attribute can have one of two values:

- False (default value): NodeCache is not used. If false (not specified), the cacheId, cacheScope and cacheTimeout settings are ignored.
- True: NodeCache is used.

cacheId

Optional (String) − The identifier name used to cache the Node. Internally, the cache is implemented as a Map; this will become the key. If not specified, the id attribute of the tag is used.

cacheTimeout

Optional (String, long) − The time, in milliseconds, for which the cached Node is valid. If more than this amount of time has passed since the Node was cached, the cached Node will be cleared, retrieved, and placed back into the cache.

Use −1 for no−timeout (always use the cached Node). Default = −1.

cacheScope

Optional (String) − Specifies the lifecycle scope of the content cache. Similar to <jsp:useBean>.

Possible values:

- application: Any JSP page in the web application that any customer requests can access the cache.
- session (the default): Any JSP in the web application that the current customer requests can access the cache.
- page: Only the current JSP that any customer requests can access the cache.
- request: Only the current user request can access the cache. If a customer re−requests the page, the content selector re−runs the query and recreates the cache.

Example

This example queries the content management system for an author and prints the title of the author's work.

```
<pz:contentQuery id="docs" query="author = 'Hemingway'" />
<ul>
<es:forEachInArray array="<%=docs%>" id="aDoc"type="com.bea.content.Node">
<li>The document title is: <cm:getProperty id="aDoc" name="Title" conversionType="html" />
</es:forEachInArray>
</ul>
```

Related Topics

<es:forEachInArray> Tag

<cm:getProperty> Tag

<cm:getNode> Tag

<cm:search> Tag

<pz:contentSelector> Tag

The <pz:contentSelector> tag implements a Content Selector defined in the Content Selector designer.

A content selector rule first evaluates a set of conditions that you define in the Content Selector designer (for example, whether or not a user fits a specified classification). If the conditions evaluate to true, all matching content is retrieved from the Virtual Content Repository based on a content query defined in the Content Selector designer.

To cache the results of the content selector rule, set the useCache attribute to true. If the cache has not timed out, subsequent calls to the <pz:contentSelector> tag will return the cached results without re−evaluating the content query.

The <pz:contentSelector> tag is empty (attributes only) and returns an array of Node objects from the Virtual Content Repository as a result of executing the content query.

Use the following tags in conjunction with <pz:*> tags to access and display the returned content:

- <es:forEachInArray> to iterate over the Node objects in the array.
- <cm:*> tags to access metadata attributes in the content, retrieve content, and put it into the HTTP response output stream.

Content Selectors can display text content (content property values and binary text content such as HTML files) and non−text binary content (such as graphics). To render non−text binary content, you must use the ShowBinary servlet, as shown in Example 2.

Syntax

<tagName *attribute*="*value*" />

Attributes

rule

Required (String) − The name of the content selector in the content selector definitions created in the Content Selector designer.

max

Optional (String, long) − Limits the maximum number of content items returned. If not present, or if equal to −1L, it returns all of the content items found.

sortBy

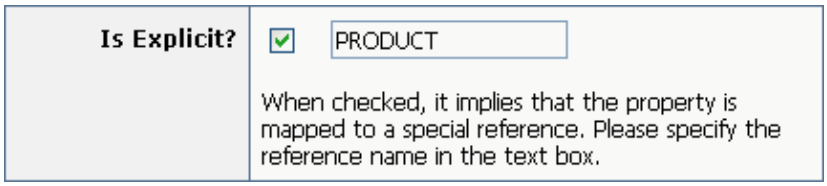
Optional (String) − Use this attribute to sort the retrieved content in ascending or descending order. Enter the content attributes on which you want to sort the content.

However, you can sort content only on content properties that are explicit, meaning the values are stored in their own database column. System properties (see com.bea.content.expression.Search) are explicit by default.

But the content properties you create in the WebLogic Administration portal (properties contained within the content "types" you create) are implicit. If you want to sort on those properties, you must make them explicit by doing the following:

1. In your database, create a column in CM_NODE table of type Integer to store the explicit property values you want to sort on. Name the column whatever you like (within the naming contraints).
2. In the WebLogic Administration Portal, set the property you want to sort on to be explicit, and enter the name of the database column you created.

   For example, if you have a property called "myproducts," and you created a database column called "PRODUCT" for it, the following is how you would define the "myproducts" property: select the "Is Explicit?" option and enter "PRODUCT" as the special reference.



   Any content nodes that use this property will have the property value stored in the PRODUCT database column, and the retrieved content can be sorted by that property.

Use ASC and DESC to sort retrieved content in ascending or descending order.

Examples:

sortBy= myproducts ASC

sortBy="cm_modifiedDate DESC" (This is a system attribute that is explicit by default.)

query

Optional (String) − A content query string used to search for content. This query overrides any query that was defined for the Content Selector in the Content Selector designer.

Example:

query="cm_contentType contains 'text' && author='Salinger'".

appendQuery

Optional (String) − A content query string to append to the resulting content query from the rule. If the content selector rule conditions do not match, this value will be ignored. If this value starts || (two vertical bars), it will be logically OR'ed with the content query. If it starts with && (two ampersands), it will be logically AND'ed with the content query. If it doesn't start with || or &&, it will be logically AND'ed with the content query.

id

Required (String) − The array variable name that contains the Node objects found. If no content is found, the variable is assigned an empty array (not null) of Node objects.

useCache

Optional (String, Boolean) − Determines whether content is cached. This attribute can have one of two values:

False (default value): The content cache is not used. If false (not specified), the cacheId, cacheScope and cacheTimeout settings are ignored.

True: Content cache is used.

cacheId

Optional (String) − The identifier name used to cache the content. Internally, the cache is implemented as a Map, which will become the key. If not specified, the id attribute of the tag is used.

cacheTimeout

Optional (String, long) − The time, in milliseconds, for which the cached content is valid. If more than this amount of time has passed since the content was cached, the cached content will be cleared, retrieved, and placed back into the cache.

Use −1 for no−timeout (always use the cached content). Default = −1.

cacheScope

Optional (String) − Specifies the lifecycle scope of the content cache. Similar to <jsp:useBean>.

Possible values:

- application: Any JSP page in the web application that any customer requests can access the cache.
- session (the default): Any JSP in the web application that the current customer requests can access the cache.
- page: Only the current JSP that any customer requests can access the cache.
- request: Only the current user request can access the cache. If a customer re−requests the page, the content selector re−runs the query and recreates the cache.

Example 1

This example shows retrieval and display of text−based content from the Virtual Content Repository. The code identifies a user to target and uses a Content Selector rule created in the Content Selector designer called "PremierCustomerSpotlight" to run a query and print the title of a document retrieved from the query. If the value of the "name" attribute in the <cm:getProperty> tag is a text−based binary file, such as an HTML file, that text will be rendered on the page.

```
<profile:getProfile profileKey="bob" profileId="myProfile" scope="session"/>
<pz:contentSelector rule="PremierCustomerSpotlight" id="docs" />
<ul>
<es:forEachInArray array="<%=docs%>" id="aDoc"
type="com.bea.content.Node">
```

```
<li>The document title is: <cm:getProperty id="aDoc" name="Title" conversionType="html" />
</es:forEachInArray>
</ul>
```

Example 2

This example shows retrieval and display of non−text binary content (graphics) from the Virtual Content Repository. Notice in the HTML <img> tag the required use of the getContextPath method, the ShowBinary servlet, and the getPath method to render the graphics.

```
<html>
<head><title>Here's Your Content</title></head>
<body>
<p>Here's your content:</p>
<p>
<pz:contentSelector id="nodes" rule="modern" />
    <es:forEachInArray array="<%=nodes%>" id="node" type="com.bea.content.Node" >
        <img src=<%=request.getContextPath() + "/ShowBinary" + node.getPath()%> >
    </es:forEachInArray></p>

<pz:contentSelector id="nodes" rule="classic" />
    <es:forEachInArray array="<%=nodes%>" id="node" type="com.bea.content.Node" >
        <img src=<%=request.getContextPath() + "/ShowBinary" + node.getPath()%> >
    </es:forEachInArray></p>

</body>
</html>
```

Related Topics

Creating Content Selectors

<es:forEachInArray> Tag

<cm:getProperty> Tag

<cm:getNode> Tag

<cm:search> Tag

<pz:div> Tag

The <pz:div> tag allows a piece of in−line content to be shown if a user belongs to the specific User Segment defined in the User Segment designer. This tag has a begin tag, a body, and an end tag. The tag returns a list of User Segment objects to which the user belongs.

Syntax

<tagName *attribute*="*value*" >
</tagName>

Attributes

rule

Required (String) − The name of the User Segment defined in the User Segment designer.

id

Optional (String) − A collection that contains the User Segment objects that apply to the user.

Example

This example displays an HTML−formatted paragraph to users belonging to the PremierCustomer segment.

```
<profile:getProfile profileKey="bob" profileId="myProfile" scope="session"/>
<pz:div id= classifications  rule="PremierCustomer">
<%
//if the user is classified as a Premier Customer, a list with one entry should be returned//
java.util.Iterator iterator=classifications.iterator();
while (iterator.hasNext())
{
com.bea.p13n.user. Classification classification=(Classification) iterator.next();
out.println (classification.getName());
}
%>
<p>Please check out our new Premier Customer bonus program.<p>
</pz:div>
```

Related Topics

Creating User Segments

<BehaviorTracking:clickContentEvent> Tag

The <BehaviorTracking:clickContentEvent> tag is used to generate a behavior event when a user has clicked (through) an image or link. This tag will return a URL query string containing event parameters, which is then used to form the complete URL that hyperlinks the content.

Syntax

<tagName *attribute*="*value*" />

Attributes

documentId

Optional (String) − The path of the item that is displayed, if applicable (that is, an image URL or banner image ID).

documentType

Optional (String) − Type or category of the item that is displayed (if applicable).

id

Optional (String) − Page variable which will hold the output of this tag.

userId

Optional (String) − Name of the user for which that content was retrieved. If the optional value is not provided, it will be set to the value of the request.getRemoteUser().

Example

```
<BehaviorTracking:clickContentEvent documentId="<%= documentId %>" documentType="<%= documentTy
userId="<%= userId %>" id="outputFromTag" />
```

Related Topics

<BehaviorTracking:displayContentEvent> Tag

<BehaviorTracking:displayContentEvent> Tag

The <BehaviorTracking:displayContentEvent> tag generates a behavior event when a user has received (viewed) any HTML content, such as images, text, PDF files, and Web−compatible multimedia content.

Syntax

<tagName *attribute*="*value*" />

Attributes

documentId

Optional (String) − The path of the item that is displayed, if applicable.

documentType

Optional (String) − Type or category of the item that is displayed, if applicable.

Example

This example shows a code snippet of processing that would follow the retrieval of content from a content management system. For each document returned but not displayed in this example, the <BehaviorTracking:displayContentEvent> tag generates an event and passes the document's ID and type.

```
<es:forEachInArray id="nextRow" array="<%=headlines%>" type="com.bea.content.Node">
<es:notNull item="<%=nextRow%>">
<BehaviorTracking:displayContentEvent documentId="<%=nextRow.getPath()%>"
documentType="<%=headingProp%>"/>
</es:notNull>
</es:forEachInArray>
```

Related Topics

<BehaviorTracking:clickContentEvent> Tag

<es:forEachInArray> Tag

<es:notNull> Tag

Content Management JSP Tags

Following are the Portal Content Management JSP Tags. The are contained in content.jar.

cm:getNode	Retrieves a content node based on an explicit path and stores the node in a variable.
cm:getProperty	Retrieves a property value from a content node and stores it as a variable or prints it in the JSP.
cm:search	Searches for and retrieves content nodes based on a supplied query and stores the results in a variable.

<cm:getNode> Tag

The <cm:getNode> tag retrieves a content node (item) or a hierarchy node (directory) based on an explicit path and stores the node in a variable. Use this tag to retrieve content if you know the exact path of the content node.

Syntax

<tagName *attribute*="*value*" />

Attributes

path

Required (String) − The repository path of the node to retrieve. The path must start with /<REPOSITORY_NAME>/. For example:
/BEA Repository/graphics/hr/open_enrollment retrieves a content item called "open_enrollment."

id

Required (String) − The variable in which the content node is placed.

failOnError

Optional (boolean) − This attribute can have one of two values:

- false (default value): Handles JSP processing errors gracefully and prints nothing if an error occurs. Returns null on an error.
- true: Throws an exception. You can handle the exception in the code, let the page proceed to the normal error page, or let the application server handle it less gracefully.

useCache

Optional (boolean) − Set to true to cache the results. (The default is "false.") A different cache is used than what is set in the Administration Portal. If you are already using application cache, as determined in the Administration Portal's Server Administration page, do not set this cache attribute. Using more than one cache can result in unreliable data retrieval.

cacheScope

Optional (String) − The scope in which the cache is used: "application," "page," "request," or "session." The default is "session." See the description of the ***useCache*** attribute for guidance.

cacheId

Optional (String) − The variable in which the cached items are stored. See the description of the ***useCache*** attribute for guidance.

cacheTimeout

Optional (long) − The number of milliseconds to store the items in cache. See the description of the ***useCache*** attribute for guidance.

Example

```
<cm:getNode path="/BEA Repository/graphics/hr/open_enrollment" id="node" />
<div>
    <%= node %>
</div>
```

Related Topics

<cm:getProperty> Tag

<cm:search> Tag

<cm:getProperty> Tag

The <cm:getProperty> tag retrieves a property value from a content node. Properties are contained in "types" in the BEA Virtual Content Repository. You can store the property value in a variable or print it. Use this tag for text−only property values or text−based binary files (such as HTML, XML, or TXT). Do not use this tag to retrieve non−text binaries, such as graphics.

If a property format is binary, but the file stored in that property is a text file, the entire text file is considered to be the property value.

Syntax

<tagName *attribute*="*value*" />

Attributes

node

Optional (Node) − The content node from which to retrieve the property. If this attribute is left blank or if the node is not found using this value, the ***id*** value is used.

name

Optional (String) − The name of the property to retrieve. If you select a system property (createdBy, createDate, modifiedBy, modifiedDate, and path), the value of that node's property is retrieved. If you leave this attribute empty, the content node's primary property is used, if defined.

id

Optional (String) − The pageContext attribute name to use for the node.

resultId

Optional (String) − The variable in which the property value is stored. Use with the ***id*** attribute. If this attribute is not used, the value of the property is printed in the JSP instead of stored as a variable. Many of the following properties can be used to format the property value printed in the JSP when this attribute is not set.

isMultiple

Optional (boolean) − If the property contains multiple values, set to "true." The ***resultId*** variable is treated as a Collection. If the property has a single value, set to "false." The ***resultId*** is treated as an Object that can be cast to the Java type specified in the ***resultType*** attribute. The default is "true."

resultType

Optional (String) − The Java type to which the single property value is cast. You cannot use this property if the ***isMultiple*** attribute is set to "true," the default.

default

Optional (String) − If the ***resultId*** is not set, this specifies the text to print if the property does not exist for the content object.

conversionType

Optional (String) − If the ***resultId*** is not set, this setting ("html," "url," or "none") tells the JSP how to render the property value. The "html" setting converts HTML special characters to their entity form. The "url" setting converts text using java.net.URLEncoder. The "none" setting performs no conversion or encoding. The default is "none."

maxLength

Optional (int) − If the ***resultId*** is not set, this sets a maximum character length of the the printed text.

dateFormat

Optional (String) − If the ***resultId*** is not set and the property is a java.util.Date value, this determines a date format. Use java.util.SimpleDateFormat notation.

numFormat

Optional (String) − If the resultId is not set and the property is a number, this determins the number format. Use java.util.NumFormat notation.

blockSize

Optional (int) − Sets the size of the blocks in bytes in which the property value is read. It is used to read in text binary property values (such as HTML files) to determine the chunk size that is read from the InputStream and written to the PageContext output writer.

startIndex

Optional (int) − The index (from 0) in bytes to start printing the content.

endIndex

Optional (int) − The index (from 0) in bytes to stop printing the content.

baseHref

Optional (String) − Use this attribute only if the property value (usually a text−based binary file) contains relative links to other documents stored in the BEA Virtual Content Repository. This attribute resolves references to repository−based content by setting the base reference to the ShowBinaryServlet. The value should usually be <%=request.getContextPath() + "/ShowBinary"%>.

failOnError

Optional (boolean) − This attribute can have one of two values:

- false (default value): Handles JSP processing errors gracefully and prints nothing if an error occurs. Returns null on an error.

- true: Throws an exception. You can handle the exception in the code, let the page proceed to the normal error page, or let the application server handle it less gracefully.

transactionTimeout

Optional (int) − The time, in seconds, before a transaction will timeout. Set this attribute if you need a transaction timeout longer than WebLogic default settings. The default value for this attribute is −1, which indicates that the transaction timeout default to the WebLogic default settings.

Example

```
<p>
        <cm:getProperty node="<%=images>" name="demographic" default="No content node found." /
</p>
```

Related Topics

<cm:getNode> Tag

<cm:search> Tag

<cm:search> Tag

The <cm:search> tag searches for and retrieves content nodes based on a supplied query and stores the results in a variable.

Syntax

<tagName *attribute*="*value*" />

Attributes

query

Optional (String) − The search query to execute based on the query syntax described in the Portal Javadoc for com.bea.p13n.expression.ExpressionHelper. Use declarative syntax such as "color='blue'". An expression is constructed with the query value. This is used if the ***expression*** attribute is not used.

expression

Optional (Expression) − The search expression to execute as a com.bea.p13n.expression.Expression. This value can be set programmatically.

id

Required (String) − The variable in which the search results are placed.

max

Optional (String) − The maximum number of content nodes returned by the search.

sortBy

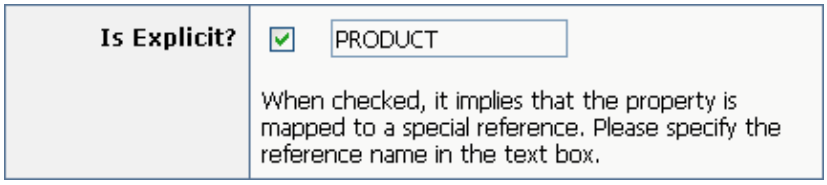
Optional (String) − Use this attribute to sort the retrieved content in ascending or descending order. Enter the content attributes on which you want to sort the content.

However, you can sort content only on content properties that are explicit, meaning the values are stored in their own database column. System properties (see com.bea.content.expression.Search) are explicit by default. But the content properties you create in the WebLogic Administration portal (properties contained within the content "types" you create) are implicit. If you want to sort on those properties, you must make them explicit by doing the following:

1. In your database, create a column in CM_NODE table of type Integer to store the explicit property values you want to sort on. Name the column whatever you like (within the naming contraints).
2. In the WebLogic Administration Portal, set the property you want to sort on to be explicit, and enter the name of the database column you created.

   For example, if you have a property called "myproducts," and you created a database column called "PRODUCT" for it, the following is how you would define the "myproducts" property:

select the "Is Explicit?" option and enter "PRODUCT" as the special reference.



Any content nodes that use this property will have the property value stored in the PRODUCT database column, and the retrieved content can be sorted by that property.

Use ASC and DESC to sort retrieved content in ascending or descending order.

Examples:

sortBy= myproducts ASC

sortBy="cm_modifiedDate DESC" (This is a system attribute that is explicit by default.)

failOnError

Optional (boolean) − This attribute can have one of two values:

- false (default value): Handles JSP processing errors gracefully and prints nothing if an error occurs. Returns null on an error.
- true: Throws an exception. You can handle the exception in the code, let the page proceed to the normal error page, or let the application server handle it less gracefully.

contextParams

Optional (Map) − The context parameters of the search.

useCache

Optional (boolean) − Set to true to cache the search results at the servlet container layer. (The default is "false.") A different cache is used than what is set in the Administration Portal. You do not need to set this cache attribute if you are already using application cache.

cacheScope

Optional (String) − The scope in which the cache is used: "application," "page," "request," or "session." The default is "session." See the description of the ***useCache*** attribute for guidance.

cacheId

Optional (String) − The variable in which the cached items are stored. See the description of the ***useCache*** attribute for guidance.

cacheTimeout

Optional (long) − The number of milliseconds to store the items in cache. See the description of the ***useCache*** attribute for guidance.

Example

```
<cm:search id="docs" query="author = 'Hemingway'" />
<ul>
<es:forEachInArray array="<%=docs>" id="aDoc" type="com.bea.content.Node">
    <li>The document title is: <cm:getProperty id="aDoc" name="Title" conversionType="html" />
</es:forEachInArray>
</ul>
```

Related Topics

<cm:getNode> Tag

<cm:getProperty> Tag

<es:forEachInArray>

Property Set JSP Tags

Following are the Portal Property Set JSP Tags. The are contained in ps_taglib.jar.

ps:getPropertyNames	Gets a list of property names in a property set.
ps:getPropertySetNames	Gets a list of property sets for a property set type.
ps:getRestrictedPropertyValues	Gets the pick list of a property's restricted values.

<ps:getPropertyNames> Tag

The <ps:getPropertyNames> tag is used to get a list of property names in a property set.

Note: All Property Sets tags send results to the same file. If you are checking for results, include this import directive at the top of the page: <%@ page import="com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySetName

Required (String) − The name of the property set to search.

propertySetType

Required (String) − Type of property set to search.

id

Required (String) − The id of the variable to hold the list of property names, as a String array.

result

Optional (String) − The identifier of an Integer variable that will be created and initialized with the result of the operation.

Possible values:

- Query is successful: PropertySetTagConstants.PROPERTY_SEARCH_OK.
- Problem getting the list of property names: PropertySetTagConstants.PROPERTY_SEARCH_FAILED.
- Property set named by propertySetName and propertySetType could not be found: PropertySetTagConstants.INVALID_PROPERTY_SET.

Example

This example retrieves and lists properties from the "Demographics" property set.

```
<%@ page import= "com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants" %>
.
.
.
<% String myPropertySet="Demographics"; %>
<p> ------- <b>ps:getPropertyNames</b> ------------
<br>
```

```
<ps:getPropertyNames propertySetName="<%= myPropertySet %>"
propertySetType="USER" id="propertyNames" result="myResult"/>
<% for (int i=0; i < propertyNames.length; i++)
out.println(propertyNames[i] + "<br>");
%>
```

Related Topics

<ps:getPropertySetNames> Tag

<ps:getRestrictedPropertyValues> Tag

<ps:getPropertySetNames> Tag

The <ps:getPropertySetNames> tag is used to get a list of property sets given a property set type.

Note: All Property Sets tags send results to the same file. If you are checking for results, include this import directive at the top of the page: <%@ page import="com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySetType

Required (String) − Type of property set to search.

id

Required (String) − The identifier of the variable to hold the list of property set names, as a String array.

result

Optional (String) − The identifier of an Integer variable that will be created and initialized with the result of the operation.

Possible values:

- Query is successful: PropertySetTagConstants.PROPERTY_SET_SEARCH_OK.
- Problem getting the list of property names: PropertySetTagConstants.PROPERTY_SET_SEARCH_FAILED.

Example

This example shows how to use retrieve property sets of type "USER."

```
<ps:getPropertySetNames propertySetType="USER"
id="userPropertySets" result="myResult" />
```

Related Topics

<ps:getPropertyNames> Tag

<ps:getRestrictedPropertyValues> Tag

<ps:getRestrictedPropertyValues> Tag

The <ps:getRestrictedPropertyValues> tag returns a list of restricted values (for example, a list of languages from a "PreferredLanguage" property) for a specific property definition, converted into Strings. These values will be returned as an array of Strings.

Note: All Property Sets tags send results to the same file. If you are checking for results, include this import directive at the top of the page: <%@ page import="com.bea.p13n.property.servlets.jsp.taglib.PropertySetTag− Constants" %>

Syntax

<tagName *attribute*="*value*" />

Attributes

propertySetName

Required (String) − The name of the property set containing the property.

propertySetType

Required (String) − Type of property set containing the property.

propertyName

Required (String) − The name of the property from which to retrieve the restricted values.

id

Required (String) − The identifier of the variable to hold the list of allowed values for the property, as a String array.

result

Optional (String) − The identifier of an Integer variable that will be created and initialized with the result of the operation.

Possible values:

- Query is successful: PropertySetTagConstants. PROPERTY_SEARCH_OK.
- Problem accessing the property: PropertySetTagConstants. PROPERTY_SEARCH_FAILED.
- Property set named by propertySetName and propertySetType could not be found: PropertySetTagConstants. INVALID_PROPERTY_SET.
- The requested property is not restricted: PropertySetTagConstants.PROPERTY_NOT_RESTRICTED.

Example

This example shows how to retrieve and print a list of languages from a "PreferredLanguage" property.

```
<%@ page import= "com.bea.p13n.property.servlets.jsp.taglib.PropertySetTagConstants"%>
<p>Possible values for PreferredLanguage:
<ps:getRestrictedPropertyValues propertySetName="Demographics"
propertySetType="USER" propertyName="PreferredLanguage"
id="values" result="myResult"/>
<ul>
<% if (myResult.intValue() == PropertySetTagConstants.PROPERTY_SEARCH_OK) { for ( int i=0; i<va
<li><%=values[i]%><% } }%>
</ul>
```

Related Topics

<ps:getPropertyNames> Tag

<ps:getPropertySetNames> Tag

Internationalization and Localization JSP Tags

Following are the Portal Internationalization and Localization JSP Tags. The are contained in i18n_taglib.jar and l10n_taglib.jar.

i18n:getMessage	Retrieves and displays localized static text or messages from a JspMessageBundle.
i18n:localize	Defines the language, country, variant, and base bundle name to be used throughout a page when accessing resource bundles with the i18n:getmessage tag.
l10n:include	Includes a localized version of a page.
l10n:forward	Forwards to a localized version of a page.
l10n:resolve	Resolves to a localized version of a page.

<i18n:getMessage> Tag

This tag is used in conjunction with the <i18n:localize> tag to retrieve localized static text or messages from a JspMessageBundle for display on a page or in a Portlet.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Optional (String) − Holds the value of the label (or message) in the JSP. If set, this attribute is the name of the variable to hold the message as a String. If not set, the message is output to the JSP.

messageName

Required (String) − The key for the message bundle.

messageArgs

Optional (Object arguments) − The arguments to the message bundle. If no arguments are provided, it is assumed that static text (not a message) is to be returned.

For example, {"Wednesday", "78"}; might be used to construct the message Today is Wednesday, and the temperature is 78 degrees Fahrenheit.

bundleName

Optional (String) − If properly initialized in the <i18n:localize> tag, there is no need to pass this tag attribute unless you want to use a different bundle for a particular tag invocation.

Performance Tuning: You can determine how often WebLogic Portal checks for updated content in resource bundles. The default is for resource bundles to never expire. In the portal Web project's <project>/WEB−INF/web.xml file, add a <context−param> entry with i18n.bundle.reload.seconds, as shown in the following example:

```
<context−param>
    <param−name>i18n.bundle.reload.seconds</param−name>
    <param−value>300</param−value>
</context−param>
```

The <param−value> number is in seconds; 300 seconds is 5 minutes. Changing the value is a balance between flexibility and performance. If your resource bundle content changes frequently, lowering the value makes WebLogic Portal update content more frequently, though at a cost to performance. If your resource bundle content stays relatively static, increase performance by choosing a higher value, such as 86400 (24 hours).

The default is for resource bundles to never exprire, which can be explicitly set by a negative value such as −1. If you use the default, you must redeploy the Web application to update resource bundles.

language

Optional (String) − If properly initialized in the <i18n:localize> tag, there is no need to pass this tag attribute, unless you want to use a different language for a particular tag invocation.

country

Optional (String) − If properly initialized in the <i18n:localize> tag, there is no need to pass this tag attribute, unless you want to use a different country for a particular tag invocation.

variant

Optional (String) − If properly initialized in the <i18n:localize> tag, there is no need to pass this tag attribute, unless you want to use a different variant for a particular tag invocation. The variant is used when localization demands a more specific locale than can be denoted by having just language and a country. For example, two variants for English are U.S. English and U.K. English.

locale

Optional (java.util.Locale) − If properly initialized in the <i18n:localize> tag, there is no need to pass this tag attribute, unless you want to use a different locale (language, country, and variant) for a particular tag invocation.

Example

This example uses two messages ("greeting" and "message") in a resource bundle file called i18nExampleResourceBundle to produce this output:

"Welcome To This Page! 14 out of 100 files have been saved."

```
<%
// Definition of a single language preference
String language = "en";
// Creation of message arguments
Object[] args = new Object[]
{
new Integer(14),
new Integer(100)
};
%>
<i18n:localize language="<%=language%>" bundleName="i18nExampleResourceBundle"/>
<html>
<body>
<i18n:getMessage messageName="greeting"/>
<i18n:getMessage messageName="message" messageArgs="<%=args%>"/>
</body>
</html>
```

The following code shows the entries in the property file named i18nExampleResourceBundle. properties:

```
greeting=Welcome To This Page!
message={0} out of {1} files have been saved.
```

Related Topics

<i18n:localize> Tag

<l10n:forward> Tag

<l10n:include> Tag

<l10n:resolve> Tag

<i18n:localize> Tag

This tag allows you to define the language, country, variant, and base bundle name to be used throughout a page when accessing resource bundles via the <i18n:getmessage> tag. This tag also specifies a character encoding and content type to be specified for a JSP. Because of this, the tag should be used as early in the page as possible before anything is written to the output stream so that the bytes are properly encoded.

When an HTML page is included in a larger page, only the larger page can use the <i18n:localize> tag. This is because the <i18n:localize> tag sets the encoding for the page, and the encoding must be set in the parent (including) page before any bytes are written to the response s output stream. The parent page must set an encoding that is sufficient for all the content on that page as well as any included pages.

The preferred approach is to retrieve all strings dynamically from the <i18n:getMessage> tag, and avoid embedding strings statically (that is, avoid hard−coding them) in your JSP.

If your page contains only dynamic strings (strings retrieved using the <i18n:getMessage tag>), then do not use the <i18n:localize> tag in conjunction with the <%@ page contentType="<something>" > page directive defined in the JSP specification. The directive is unnecessary if you are using the <i18n:localize> tag, and can result in inconsistent or wrong contentType declarations.

If you must mix static strings and dynamic strings on the same page, then you will need to use the page directive. Ensure that the character set specified by the <i18n:localize> tag is compatible with the character set specified in the page directive.

Syntax

<tagName *attribute*="*value*" />

Attributes

bundleName

Optional (String) − The base name of the MessageBundle is used to retrieve localized text for a JSP page.

Performance Tuning: You can determine how often WebLogic Portal checks for updated content in resource bundles. The default is for resource bundles to never expire. In the portal Web project's <project>/WEB−INF/web.xml file, add a <context−param> entry with i18n.bundle.reload.seconds, as shown in the following example:

```
<context−param>
    <param−name>i18n.bundle.reload.seconds</param−name>
    <param−value>300</param−value>
</context−param>
```

The <param−value> number is in seconds; 300 seconds is 5 minutes. Changing the value is a balance between flexibility and performance. If your resource bundle content changes frequently, lowering the value makes WebLogic Portal update content more frequently, though at a cost to performance. If your resource bundle content stays relatively static, increase performance by choosing a higher value, such as 86400 (24 hours).

The default is for resource bundles to never exprire, which can be explicitly set by a negative value such as −1. If you use the default, you must redeploy the Web application to update resource bundles.

language

Optional (String or String []) − A String two character ISO Language Code denoting the user's preferred language, or a String[] containing a list of preferred language codes for a user, with stronger preferences indexed lower (earlier) in the array.

country

Optional (String) − The two−character ISO Country Code for a country. For example, this code would be used to look for a MessageBundle containing text localized to English speaking users in the U.S. (en−us) as opposed to English speaking users in the U.K. (en−uk).

variant

Optional (String) − A String representing a locale's variant. The variant is used when localization demands a more specific locale than can be denoted by having just language and a country. For example, two variants for English are U.S. English and U.K. English.

locale

Optional (java.util.Locale) − Instead of specifying language, country, and variant as Strings, a java.util.Locale object can be provided. If provided, the values in the Locale's language, country, and variant fields will negate any of the other language, country, and variant values passed to the tag as Strings.

charset

Optional (String) − The name of the character encoding set to use for this page. Defaults to "UTF−8" if no encoding can be determined for the chosen language; otherwise an encoding appropriate for the chosen language is used.

contentType

Optional (String) − The type of content contained in the page, defaults to "text/html".

Example1

This example shows how to use <i18n:localize> to define a single language preference.

```
<%
// Definition of a single language preference
String language = "en";
%>
<i18n:localize language="<%=language%>" bundleName="i18nExampleResourceBundle"/>
<html>
<body>
<i18n:getMessage messageName="greeting"/>
</body>
</html>
```

Example 2

This example shows how to use <i18n:localize> to define two language preferences, English and Spanish.

```
<%
// Array that defines two languages preferences − English and
// Spanish in that order of preference.
String[] languages = new String[] { "en", "es" };
%>
<i18n:localize language="<%=languages%>" bundleName="i18nExampleResourceBundle"/>
<html>
<body>
<i18n:getMessage messageName="greeting"/>
</body>
</html>
```

Related Topics

<i18n:getMessage> Tag

<l10n:forward> Tag

<l10n:include> Tag

<l10n:resolve> Tag

<l10n:forward> Tag

This tag searches for a localized version of the page URL based on the current locale and forwards to that page. If no locale is set, this tag uses the locale from the HTTP request headers. The behavior of this tag is similar to the jsp:forward tag except for the locale.

Syntax

<tagName *attribute*="*value*" />

Attributes

page

Required (String) − Path to the resource, which is relative to the current page or absolute with respect to the Web application context−root.

language

Optional (String) − Language code.

country

Optional (String) − Country code.

variant

Optional (String) − Language variant. For example, "us" (United States) is a variant of English (en).

Example

```
<l10n:forward page="/l10n_content/foo.jsp" language="en" variant="gb" />
```

Related Topics

<l10n:include> Tag

<l10n:resolve> Tag

<i18n:getMessage> Tag

<i18n:localize> Tag

<l10n:include> Tag

This tag searches for a localized version of the page URL based on the current locale and includes that page. If there is no locale set, this tag uses the local from the HTTP request headers. The behavior of this tag is similar to the jsp:include tag, except for the locale.

Syntax

<tagName *attribute*="*value*" />

Attributes

page

Required (String) − Path to the resource, which is relative to the current page or absolute with respect to the Web application context−root.

language

Optional (String) − Language code.

country

Optional (String) − Country code.

variant

Optional (String) − Language variant. For example, "us" (United States) is a variant of English (en).

Example

```
<l10n:include page="/l10n_content/foo.jsp" language="en" variant="gb" />
```

Related Topics

<l10n:forward> Tag

<l10n:resolve> Tag

<i18n:getMessage> Tag

<i18n:localize> Tag

<l10n:resolve> Tag

This tag searches for a localized version of the page URL based on the current locale and resolves the URL to a localized version. If there is no locale set, this tag uses the locale from the HTTP request headers. The localized URL is always absolute.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Optional (String) − Stores the absolute URL generated. If this attribute is not used, the URL is printed to the JSP.

page

Required (String) − Path to the resource, which is relative to the current page or absolute with respect to the Web application context−root.

language

Optional (String) − Language code.

country

Optional (String) − Country code.

variant

Optional (String) − Language variant. For example, "us" (United States) is a variant of English (en).

Example

```
<l10n:resolve page="/l10n_content/foo.jsp" language="en" variant="gb" />
```

Related Topics

<l10n:forward> Tag

<l10n:include> Tag

<i18n:getMessage> Tag

<i18n:localize> Tag

Portal Rendering JSP Tags

Following are the Portal Rendering JSP Tags. The are contained in render_taglib.jar.

render:beginRender	Used in the portal skeletons for rendering a portal resource. This tag defines the opening HTML tag for a resource.
render:endRender	Used in the portal skeletons for rendering a portal resource. This tag defines the closing HTML tag for a resource.
render:renderChild	Used in the portal skeletons for rendering a portal resource. This tag is used to render portlet titlebars, titlebar buttons, menus (navigation such as page tabs), and table cells within a layout.
render:jspContentUrl	lets you create URLs to windows and set content of those windows.
render:pageUrl	Lets you create links to switch to a page or a book based on labels.
render:standalonePortletUrl	Lets you create URLs to floatable portlets. Use this tag to create links to submit requests to portlets hosted by an external portal, such as floating portlets.
render:postbackUrl	Lets you create URLs to submit GET/POST requests to the portal framework. This tag is necessary because all portlet requests first need to go through the portal, before being delegated to the caller.
render:resourceUrl	Represents a URL to a resource. Resource URLs are typically used for static/dynamic resources such as files or images. Resource URLs cannot be used to submit requests to the portal framework.
render:windowUrl	Lets you create links to windows based on labels and switch a window to a mode, state, or both.
render:toggleButtonUrl	Used in the portal skeletons for rendering a portal resource. This tag lets you build URLs for the toggle buttons in portlet titlebars.
render:param	This tag lets you append query parameters to the URL tags.
render:jspUri	Retrieves the location of the JSP in which this tag is used.
render:getJspUri	Retrieves the location of the JSP in which the tag is used. This helps create correct relative URLs to resources such as images within a JSP. The URL omits the JSP filename.
render:createUri	Creates a URI based on the application's configured skin path.
render:writeUri	Writes the URI based on the application's configured skin path.
render:getSkinPath	

	Returns the path of an image used in the current skin. Every Look & Feel can have a different directory structure as well as different client classifications and localization; this tag resolves the location based on just the file name.
render:createId	Creates an ID for a rendered component.
render:writeId	Writes an ID for a rendered component.
render:writeAttribute	Used in the portal skeletons for rendering a portal resource. This tag sets HTML attributes for a tag.
render:encodeName	Lets you encode names such as HTML form names and JavaScript function names to make them unique such that the encoded names are unique within the context of the a portal page.

<render:beginRender> Tag

This tag is used in the portal look and feel skeleton files that render the structure of portal components. Portal components have a hierarchical relationship with each other. For example, a book contains a page and a page contains a placeholder. The <render:beginRender> and <render:endRender> tags serve as opening and closing tags for a component within which child components are rendered, which ultimately result in opening and closing HTML tags within which child components are nested.

The <render:beginRender> tag (and it's closing </render:beginRender> tag) in a component's skeleton contain the code and/or HTML markup that is processed and rendered before any child component rendering.

The WebLogic Portal Extensions provide predefined sets of skeletons that meet most portal rendering needs. Unless you are creating new skeleton JSPs by hand (rather than copying and modifying existing skeleton JSPs), you are not likely to use this tag.

Syntax

<tagName>
</tagName>

Attributes

none

Example

This example shows the page.jsp skeleton for portal page rendering. The result of the <render:beginRender> tag is an opening HTML tag populated with id, class, and style attributes for the portal page. In many skeleton files, code is included above the render tags to construct the class attribute, which is a string for a specific CSS style. The id and style attributes are usually populated for a portal component in the Propert Editor window in WebLogic Workshop Platform Edition.

The <render:endRender> tag results in the closing tag.

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:beginRender>
       <!-- Begin Page -->
       <div
             <render:writeAttribute name="id" value="<%= pageCtx.getPresentationId()%>"/>
             <render:writeAttribute name="class" value="<%= pageCtx.getPresentationClass()%>" d
             <render:writeAttribute name="style" value="<%= pageCtx.getPresentationStyle()%>"/>
       >
</render:beginRender>

<render:endRender>
       </div>
       <!-- End Page -->
```

```
</render:endRender>
```

Related Topics

<render:endRender> Tag

<render:writeAttribute> Tag

<render:renderChild> Tag

<render:endRender> Tag

This tag is used in the portal look and feel skeleton files that render the structure of portal components. Portal components have a hierarchical relationship with each other. For example, a book contains a page and a page contains a placeholder. The <render:beginRender> and <render:endRender> tags serve as opening and closing tags for a component within which child components are rendered, which ultimately result in opening and closing HTML tags within which child components are nested.

The <render:endRender> tag (and its closing </render:endRender> tag) in a component's skeleton contain the code and/or HTML markup that is processed and rendered before any child component rendering and serves as the closing tag of the parent component.

The WebLogic Portal Extensions provide predefined sets of skeletons that meet most portal rendering needs. Unless you are creating new skeleton JSPs by hand (rather than copying and modifying existing skeleton JSPs), you are not likely to use this tag.

Syntax

<tagName>
</tagName>

Attributes

none

Example

This example shows the page.jsp skeleton for portal page rendering. The result of the <render:beginRender> tag is an opening HTML tag populated with id, class, and style attributes (if available) for the portal page. The <render:endRender> tag results in the closing tag.

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:beginRender>
    <!-- Begin Page -->
    <div
        <render:writeAttribute name="id" value="<%= pageCtx.getPresentationId() %>"/>
        <render:writeAttribute name="class" value="<%= pageCtx.getPresentationClass() %>" defau
        <render:writeAttribute name="style" value="<%= pageCtx.getPresentationStyle() %>"/>
    >
</render:beginRender>

<render:endRender>
    </div>
    <!-- End Page -->
</render:endRender>
```

Related Topics

<render:beginRender> Tag

<render:writeAttribute> Tag

<render:renderChild> Tag

<render:renderChild> Tag

This tag is used in the portal look and feel skeleton files that render the structure of portal components. The <render:renderChild> tag is used to render portlet titlebars, titlebar buttons, menus (navigation such as page tabs), and table cells within a layout.

The WebLogic Portal Extensions provide predefined sets of skeletons that meet most portal rendering needs. Unless you are creating new skeleton JSPs by hand (rather than copying and modifying existing skeleton JSPs), you are not likely to use this tag.

Syntax

<tagName *attribute*="*value*" />

Attributes

presentationContext

Required (String) − The presentation context for the component being rendered. In the skeleton JSPs provided in the WebLogic Workshop Portal Extensions, the presentationContext value is passed in as a variable from a *PresentationContext declaration earlier in the JSP, as shown in the following example.

Example

This example shows the book.jsp skeleton for portal book rendering. The <render:renderChild> tag uses a "menu" variable declared as the MenuPresentationContext in the first code block to render the navigation style used in the book.

```
<%@ page import="com.bea.netuix.servlets.controls.page.BookPresentationContext,
com.bea.netuix.servlets.controls.page.MenuPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>

<%
    BookPresentationContext book = BookPresentationContext.getBookPresentationContext(request);
    MenuPresentationContext menu = (MenuPresentationContext) book.getFirstChild("page:menu");
    String bookClass = "bea-portal-book";

    if (book.isDesktopBook())
    {
    bookClass += "-primary";
    }
    String bookContentClass = bookClass + "-content";
%>
<render:beginRender>
    <!-- Begin Book -->
    <div
        <render:writeAttribute name="id" value="<%= book.getPresentationId() %>"/>
        <render:writeAttribute name="class" value="<%= book.getPresentationClass()%>" defaultVa
        <render:writeAttribute name="style" value="<%= book.getPresentationStyle() %>"/>
        cellspacing="0"
        >
        <render:renderChild presentationContext="<%= menu %>"/>
    <!-- Begin Book Content -->
```

```
    <div class="<%= bookContentClass %>">
</render:beginRender>
<render:endRender>
            </div>
    <!-- End Book Content -->
    </div>
    <!-- End Book -->
</render:endRender>
```

Related Topics

<render:beginRender> Tag

<render:endRender> Tag

<render:writeAttribute> Tag

<render:jspContentUrl> Tag

This tag lets you create URLs to windows and set content of those windows. You can use the <render:param> tag to append a query to the URL.

If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

windowLabel

Required (String) − Sets the window label to link to.

windowMode

Optional (String) − Sets the window mode for which to display the linked window, such as "HELP" or "EDIT."

windowState

Optional (String) − Sets the window state in which to display the linked window, such as "MAXIMIZED" or "MINIMIZED."

contentUri

Optional (String) − Sets the content of the portlet. This is a relative path to a JSP or HTML file.

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:jspContentUrl var="jspContent" windowLabel="news" contentUri="/myportlets/foo.jsp" />
<a href="<%=pageContext.getAttribute("jspContent")%>">See the latest foo news</a>
```

Related Topics

Creating URLs to Portal Resources

<render:pageUrl> Tag

<render:windowUrl> Tag

<render:param> Tag

<render:pageUrl> Tag

This tag lets you create links to switch to a page or a book based on labels. Such links could be placed anywhere, even on JSPs or HTML pages outside the portal.

You can use the <render:param> tag to append a query to the URL.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

pageLabel

Required (String) − Sets the page label to link to.

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

This example creates a URL to a page, stores it as a variable, and uses the variable in the link.

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:pageUrl var="pageUrl" pageLabel="home" />
<a href="<%=pageContext.getAttribute("pageUrl")%>">Home</a>
```

Related Topics

Creating URLs to Portal Resources

<render:windowUrl> Tag

<render:jspContentUrl> Tag

<render:toggleButtonUrl> Tag

<render:param> Tag

<render:standalonePortletUrl> Tag

This tag lets you create URLs to floatable portlets. Use this tag to create links to submit requests to portlets hosted by an external portal, such as floating portlets.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) −

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

Code to float the current portlet:

```
<a href="<render:standalonePortletUrl/>">Float</a>
```

Related Topics

Creating URLs to Portal Resources

<render:postbackUrl> Tag

This tag lets you create URLs to submit GET/POST requests to the portal framework. This tag is necessary because all portlet requests first need to go through the portal, before being delegated to the caller.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

```
<form method="post" action="<render:postbackUrl/>">
    <input type="reset"> <input type="submit"> <br>
    <textarea name="props" rows=20 cols=80 wrap="false" ><%=debug%></textarea>
</form>
```

Related Topics

Creating URLs to Portal Resources

<render:resourceUrl> Tag

This tag represents a URL to a resource. Resource URLs are typically used for static/dynamic resources such as files or images. Resource URLs cannot be used to submit requests to the portal framework.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

path

Optional (String) − Sets the path to the resource.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

```
<% String pathToLogoutJsp = loginPortletsUri+ "formLogin/logout.jsp"; %>
<render:resourceUrl path="<%=pathToLogoutJsp%>"/>
```

Related Topics

Creating URLs to Portal Resources

<render:windowUrl> Tag

This tag lets you create links to windows based on labels and switch a window to a mode, state, or both.

You can use the <render:param> tag to append a query to the URL.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

windowLabel

Required (String) − Sets the window label to link to.

windowMode

Optional (String) − Sets the window mode for which to display the linked window, such as "HELP" or "EDIT."

windowState

Optional (String) − Sets the window state in which to display the linked window, such as "MAXIMIZED" or "MINIMIZED."

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

This example creates a URL to a help window, stores it as a variable, and uses the variable in the link. The window will be displayed maximized.

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:windowUrl var="maxhelpurl" windowLabel="helpwindow" state="MAXIMIZED" mode="HELP" />
<a href="<%=pageContext.getAttribute("maxhelpurl")%>">Help</a>
```

This tag is used in the Portal Search portlet.

Related Topics

Creating URLs to Portal Resources

<render:pageUrl> Tag

<render:jspContentUrl> Tag

<render:toggleButtonUrl> Tag

<render:param> Tag

<render:toggleButtonUrl> Tag

This tag lets you build URLs for the toggle buttons in portlet titlebars, such as Minimize/Restore or Maximize/Restore.

You can use the <render:param> tag to append a query to the URL.

The WebLogic Portal Extensions provide predefined sets of skeletons that meet most portal rendering needs. Unless you are creating new skeleton JSPs by hand (rather than copying and modifying existing skeleton JSPs), you are not likely to use this tag.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

var

Optional (String) − The variable that will contain the value of the URL.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

scheme

Optional (String) − The protocol for the URL. Should be "http" or "https". If you do not use this attribute, the protocol used to make the current request is used.

domain

Optional (String) − Sets the domain of the server to access, such as www.bea.com. If you do not use this attribute, the domain that was used to make the current request is used.

port

Optional (String) − Sets the server port to use for the domain. If you do not use this attribute, the port that was used to make the current request is used.

pathPrefix

Optional (String) − Sets the prefix to the path.

encodeSession

Optional (boolean) − Enables URL rewriting for enconding a session ID in the URL. If the value is set to "false," the URL is not encoded with the session ID. If you do not use this attribute, the default value is "true."

template

Use the template defined within url−template−config.xml when outputting the url for this tag.

Example

This example shows the togglebutton.jsp skeleton file for toggle button rendering.

```
<%@ page import="com.bea.netuix.servlets.controls.window.ToggleButtonPresentationContext,
com.bea.netuix.servlets.controls.window.TitlebarPresentationContext,
com.bea.netuix.servlets.controls.window.WindowPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %><%
    ToggleButtonPresentationContext button = ToggleButtonPresentationContext.getToggleButtonPre
    String buttonClass = button.getPresentationClass();
    String defaultButtonClass = "bea−portal−button";
%>
<render:genericUrl var="rolloverImgSrc" path="<%= button.getRolloverImageSrc() %>"/>
<%
     String rolloverImgSrc = (String) pageContext.getAttribute("rolloverImgSrc");
%>
<render:beginRender><a <render:writeAttribute name="id" value="<%= button.getPresentationId() %
<render:writeAttribute name="class" value="<%= buttonClass %>" defaultValue="<%= defaultButtonC
<render:writeAttribute name="style" value="<%= button.getPresentationStyle() %>"/>href="<render
<render:writeAttribute name="alt" value="<%= button.getAltText() %>"/><render:writeAttribute na
```

Related Topics

Creating URLs to Portal Resources

<render:pageUrl> Tag

<render:windowUrl> Tag

<render:jspContentUrl> Tag

<render:param> Tag

<render:param> Tag

This tag lets you append query parameters to the URL tags. It allows you to attach request parameters to the url. For example, the example below creates a url with an additional request parameter like lang=foo .

Syntax

<tagName *attribute*="*value*" />

Attributes

name

Required (String) − Name of the query parameter.

value

Optional (String) − Value of the query parameter.

Example

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
<render:jspContentUrl var="jspContent" windowLabel="news"      contentUri="/myportlets/foo.jsp"
<a href="<%=pageContext.getAttribute("jspContent")%><render:param name="lang" value="en" />">Se
```

Related Topics

<render:pageUrl> Tag

<render:windowUrl> Tag

<render:jspContentUrl> Tag

<render:jspUri> Tag

This tag retrieves the location of the JSP in which the tag is used. This helps create correct relative URLs to resources such as images. The URL omits the JSP filename.

The tag is identical to <render:getjspUri> and is provided here for the sake of backward compatibility.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URI to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName attribute="value">
</tagName>

Attributes

id

Optional (String) − The name of the variable to store the JSPs URL.

webAppQualified

Optional (Boolean) − If set to "true," the name of the portal Web project is appended to the URL. This is important for resources such as images that require a Web application−qualified URL. For example, /sampleportal/images/logo.gif. If set to "false," the URL is created relative to the portal Web application.

Example

```
<render:getJspUri id="loginPortletsUri"/>
<% String pathToLogoutJsp = loginPortletsUri + "formLogin/logout.jsp"; %>
<render:resourceUrl path="<%=pathToLogoutJsp%>"/>
```

Related Topics

<render:resourceUrl>

<render:getJspUri> Tag

This tag retrieves the location of the JSP in which the tag is used. This helps create correct relative URLs to resources such as images within a JSP. The URL omits the JSP filename.

> ***Note:*** If you use this tag in a struts or Java Page Flow application, in order for the URL to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.
>
> ```
> <!−− Enable or disable Pageflow support −−>
> <pageflow>
>     <enable>true</enable>
> </pageflow>
> ```

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

id

Optional (String) − The name of the variable to store the JSPs URL.

webAppQualified

Optional (Boolean) − If set to "true," the name of the portal Web project is appended to the URL. This is important for resources such as images that require a Web application−qualified URL. For example, /sampleportal/images/logo.gif. If set to "false," the URL is created relative to the portal Web application.

Example

```
<render:getJspUri id="loginPortletsUri"/>
<% String pathToLogoutJsp = loginPortletsUri + "formLogin/logout.jsp"; %>
<render:resourceUrl path="<%=pathToLogoutJsp%>"/>
```

Related Topics

<render:resourceUrl>

<render:createUri> Tag

This tag creates a URI based on the application's configured skin path. The URI is assigned to the id attribute.

If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URI to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The URI of the skin root directory.

uriFragment

Required (String) − The relative path to the skin's root directory.

Example

```
<render:createUri id="<%= id %>" uriFragment="<%= uriFragment %>" />
```

Related Topics

<render:writeUri> Tag

<render:writeUri> Tag

The URI created by writeUri combines the URI fragment with the base skin path for the portal plus the webapp directory name (ie: /<webapp>/<basepath>/<fragement>). Neither accounts for client classification, localization, or other dynamic resource resolution features.

Note: If you use this tag in a JSP that is used in a struts or Java Page Flow application, in order for the URI to resolve correctly, Java Page Flow support must be enabled in the portal Web project's WEB−INF/netuix−config.xml file, as shown in the following example. Notice the <enable> element is set to true.

```
<!−− Enable or disable Pageflow support −−>
<pageflow>
    <enable>true</enable>
</pageflow>
```

Syntax

<tagName *attribute*="*value*" />

Attributes

uriFragment

Required (String) − The relative path to the skin's root directory.

Example

```
<render:writeUri uriFragment="<%= uriFragment %>" />
```

Related Topics

<render:createUri> Tag

<render:getSkinPath> Tag

This tag returns the path of an image used in the current skin. Every Look & Feel can have a different directory structure as well as different client classifications and localization; this tag resolves the location based on just the file name.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

imageName

Required (String) − This image used in the skin for which you want to return the path. The image name is appended to the path.

id

Optional (String) − The variable in which to store the image path.

Example

```
<render:getSkinPath id="skinPath" imageName="window−icon.gif" />
```

<render:createId> Tag

This tag creates an ID for a rendered component.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The variable that will contain the return values from the render toolkit based on the following attribute values.

baseId

Required (String) − The rendered component's base ID.

idT

Required (String) − The rendered component's idT.

defaultId

Optional (String) − The default ID of the rendered component.

Example

```
<render:createId id="<%= id %>" baseId="<%= baseId %>" idT="<%= idT %>" defaultId="<%= defaultI
```

Related Topics

<render:writeId> Tag

<render:writeId> Tag

This tag writes an ID for a rendered component.

Syntax

<tagName *attribute*="*value*" />

Attributes

baseId

Required (String) − The rendered component's base ID.

idT

Required (String) − The rendered component's idT.

defaultId

Optional (String) − The default ID of the rendered component.

Example

```
<render:writeId baseId="<%= baseId %>" idT="<%= idT %>" defaultId="<%= defaultId %>" />
```

Related Topics

<render:createId> Tag

<render:writeAttribute> Tag

This tag is used in the portal look and feel skeleton files that render the structure of portal components. Portal components have a hierarchical relationship with each other. For example, a book contains a page and a page contains a placeholder. The <render:beginRender> and <render:endRender> tags serve as opening and closing tags for a component within which child components are rendered, which ultimately result in opening and closing HTML tags within which child components are nested. The <render:writeAttribute> tag is to populate the <render:beginRender> HTML tag with HTML attributes.

The WebLogic Portal Extensions provide predefined sets of skeletons that meet most portal rendering needs. Unless you are creating new skeleton JSPs by hand (rather than copying and modifying existing skeleton JSPs), you are not likely to use this tag.

Syntax

<tagName *attribute*="*value*" />

Attributes

name

Required (String) − Name portion of the name/value pair. Enter the name of a valid HTML attribute, such as id, class, or style, for the type of HTML tag you are using.

value

Optional (String) − Value portion of the name/value pair. Enter the value of the HTML attribute you specified. The WebLogic Workshop Portal Extensions use presentation methods to return a value, as shown in the following example. The methods also enable the population of id, class, and style values that are entered in the WebLogic Workshop Enterprise Edition Property Editor window for a selected portal resource.

defaultValue

Optional (String) − The default attribute value. You can use a variable for the default value as shown in the following example.

Example

This example shows the page.jsp skeleton for portal page rendering. The <render:writeAttribute> tags are nested within the HTML tag to provide the tag's id, class, and style attributes. In many skeleton files, code is included above the render tags to construct the class attribute, which is a string for a specific CSS style. The id and style attributes are usually populated for a portal component in the Propert Editor window in WebLogic Workshop Platform Edition.

```
<%@ page import="com.bea.netuix.servlets.controls.page.PagePresentationContext,
com.bea.netuix.servlets.controls.page.BookPresentationContext"%>
<%@ taglib uri="render.tld" prefix="render" %>
.
.
.
```

```
<render:beginRender>
    <!-- Begin Page -->
    <div
        <render:writeAttribute name="id" value="<%= pageCtx.getPresentationId() %>"/>
        <render:writeAttribute name="class" value="<%= pageCtx.getPresentationClass() %>" defau
        <render:writeAttribute name="style" value="<%= pageCtx.getPresentationStyle() %>"/>
    >
</render:beginRender>

<render:endRender>
    </div>
    <!-- End Page -->
</render:endRender>
```

Related Topics

<render:beginRender> Tag

<render:endRender> Tag

<render:renderChild> Tag

<render:encodeName> Tag

This tag lets you encode names such as HTML form names and JavaScript function names to make them unique such that the encoded names are unique within the context of the a portal page.

This tag frees you from dealing with potential name collisions that occur when two or more portlets (or multiple instances of the same portlet) use the same name to in the generated markup.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

name

Required (String) − The name to encode. The name can be read in as a variable. If the name is a valid Java identifier, the encoded name will also be a valid Java identifier.

var

Optional (String) − The variable that will contain the value of the URLencoded names.

scope

Optional (String) − This attribute has three primary variations:

PAGE SCOPE − (This is the default) The named reference remains available in this PageContext until the return from the current Servlet.service() invocation.

REQUEST SCOPE − The named reference remains available from the ServletRequest associated with the Servlet that until the current request is completed.

SESSION SCOPE − (Valid only if this page participates in a session). The named reference remains available from the HttpSession (if any) associated with the Servlet until the HttpSession is invalidated.

Example

This example shows a JSP with an HTML form and a JavaScript function that populates value of a text field.

```
<form action="&" method="POST">

   <input name="Name" id="nameField" value=""/>

</form>

<script type="text/javascript" language="JavaScript">
function replaceIt()
{
```

```
var elem = document.getElementById("nameField");
elem.value = "Some Value";
}
</script>
```

This JSP fragment may not function as expected when the value of the id attribute on the input field is not unique. This may happen, for example, when there is another portlet on the same page using the same value for the id attribute or the name of the JavaScript function. In order to avoid this, you must encode both the JavaScript function name and the value of the attribute as shown below:

```
<form action="&" method="POST">

 <input name="Name" id="<render:encodeName name= nameField /> " value=""/>

</form>

<script type="text/javascript" language="JavaScript">
function <render:encodeName name= replaceIt />()
{
var elem = document.getElementById("nameField");
elem.value = "Some Value";
}
</script>
The use of this tag will now ensure that the id attribute and the function name are rewritten t
```

Portlet Preferences JSP Tags

Following are the Portlet Preferences JSP Tags. The are contained in prefs_taglib.jar.

pref:getPreference	Gets the default value of a portlet preference or sets a default value.
pref:getPreferences	Gets all the values of a given portlet preference or supplies default value.
pref:forEachPreference	Iterates over all the preferences available for a portlet and stores the results in a variable.
pref:ifModifiable	Includes the body of the tag if the given portlet preference is modifiable.
pref:else	Includes the body of the tag if the given portlet preference is not modifiable.

<pref:getPreference> Tag

When you create a portlet with the WebLogic Workshop Portal Extensions Portlet Designer, you can add different preferences to it that help control the portlet's content and behavior (choose ***Insert−−>New Preference***). The portlet preferences JSP tags let you expose those preferences to end users, letting them view and set preferences.

The <pref:getPreference> tag returns the value of a given preference. Use this tag to get the value of a preference, given the name of the preference. The tag may also supply a default value for the value of the tag. If the named preference does not exist, this tag returns the supplied default value. If the named preference has more than one value, the tag returns the first value (no guarantee of order).

Syntax

<tagName *attribute*="*value*" />

Attributes

name

Required (String) − The name of the portlet preference.

defaultValue

Optional (String) − The default value of the preference. If the given preference has no value, the value you enter is used as the default value. If this attribute is not set, an empty string is used as the default value.

var

Optional (String) − The variable to assign the value to. This tag sets a java.lang.String value in the PageContext with the given var name. If this attribute is not set, the value of the preference is written to the underlying response.

Example

```
<pref:getPreference name="ContentURL" var="ContentURL" />
```

The Portal Samples contain examples of portlet preference JSP tags. In the Sample Portal, the RSS RSS News Feed Portlet uses the JSP tags, and the Tutorial Portal contains a Portlet Preferences page.

Related Topics

<pref:getPreferences> Tag

<pref:forEachPreference> Tag

<pref:ifModifiable> Tag

<pref:else> Tag

<pref:getPreferences> Tag

When you create a portlet with the WebLogic Workshop Portal Extensions Portlet Designer, you can add different preferences to it that help control the portlet's content and behavior (choose ***Insert−−>New Preference***). The portlet preferences JSP tags let you expose those preferences to end users, letting them view and set preferences.

The <pref:getPreferences> tag returns all the values of a given preference. Use this tag to get all the values of a preference, given the name of the preference. The tag may also supply a default value. If the named preference value does not exist, this tag returns the supplied default value.

Syntax

<tagName *attribute*="*value*" />

Attributes

name

Required (String) − The name of the portlet preference.

defaultValue

Optional (String) − The default value of the preference. If the given preference has no value, the value you enter is used as the default value. If this attribute is not set, an empty string is used as the default value. You can specify multiple default values separated by the given separator (used the separator attribute). For example, if a preference has two values, "blue" and "black", this attribute may be specified as "blue,black" with the separator as "," or "blue</td><td>black" with the separator as "</td><td>".

var

Optional (String) − The variable to assign the value to. This tag sets a java.lang.String value in the PageContext with the given var name. If this attribute is not set, the value of the preference is written to the underlying response with the values separated by the separator.

separator

Optional (String) − If a defaultValue is specified, specify the text that will separate multuple values. If the var is not specified, the separator is used to separate multiple values.

Example

```
<td>downloadFolder</td>
<pref:getPreferences name="downloadFolder" var="downloadFolder" separator="} {"/>
<td><%for(int i = 0; i < downloadFolder.length; i++) {
    %>{<%=downloadFolder[i]%>} <%}%></td>
```

The Portal Samples contain examples of portlet preference JSP tags. In the Sample Portal, the RSS RSS News Feed Portlet uses the JSP tags, and the Tutorial Portal contains a Portlet Preferences page.

Related Topics

<pref:getPreference> Tag

<pref:forEachPreference> Tag

<pref:ifModifiable> Tag

<pref:else> Tag

<pref:forEachPreference> Tag

When you create a portlet with the WebLogic Workshop Portal Extensions Portlet Designer, you can add different preferences to it that help control the portlet's content and behavior (choose ***Insert−−>New Preference***). The portlet preferences JSP tags let you expose those preferences to end users, letting them view and set preferences.

The <pref:forEachPreference> tag loops over all the preferences available for a portlet and stores the results in an array of String objects.

Syntax

<tagName *attribute*="*value*" />

Attributes

nameVar

Required (String) − Variable to assign the name of the preference to.

valueVar

Required (String) − Variable to assign the value of the preference to.

Example

```
<pref:forEachPreference nameVar="name" valueVar="values">
    <tr>
        <td><%=name%></td>
        <td>
            <!−− Use the es:forEachInArray tag to iterate over the values −−>
            <es:forEachInArray id="prefVal" array="<%=values%>" type="String">
            <%= prefVal %>
            </es:forEachInArray>
        </td>
    </tr>
</pref:forEachPreference>
```

The Portal Samples contain examples of portlet preference JSP tags. In the Sample Portal, the RSS RSS News Feed Portlet uses the JSP tags, and the Tutorial Portal contains a Portlet Preferences page.

Related Topics

<pref:ifModifiable> Tag

<pref:else> Tag

<pref:getPreference> Tag

<pref:getPreferences> Tag

<es:forEachInArray> Tag

<pref:ifModifiable> Tag

When you create a portlet with the WebLogic Workshop Portal Extensions Portlet Designer, you can add different preferences to it that help control the portlet's content and behavior (choose ***Insert−−>New Preference***). The portlet preferences JSP tags let you expose those preferences to end users, letting them view and set preferences.

The <pref:ifModifiable> tag includes the body of the tag if the given portlet preference is modifiable. If the preference is not modifiable, the body of this tag is skipped. Use the <pref:else> tag to display body content if the <pref:ifModifiable> tag content is skipped.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

name

Required (String) − The name of the portlet preference.

Example

```
<pref:ifModifiable name="showChannelDescription">
    <input type="checkbox" name="showChannelDescription" value="true" <%=showChannelDescription
    Show Channel Description
    <%somethingToEdit = true;%>
</pref:ifModifiable>

<pref:else>
    Channel description is unavailable.
</pref:else>
```

The Portal Samples contain examples of portlet preference JSP tags. In the Sample Portal, the RSS RSS News Feed Portlet uses the JSP tags, and the Tutorial Portal contains a Portlet Preferences page.

Related Topics

<pref:else> Tag

<pref:getPreference> Tag

<pref:getPreferences> Tag

<pref:forEachPreference> Tag

<pref:else> Tag

When you create a portlet with the WebLogic Workshop Portal Extensions Portlet Designer, you can add different preferences to it that help control the portlet's content and behavior (choose ***Insert−−>New Preference***). The portlet preferences JSP tags let you expose those preferences to end users, letting them view and set preferences.

The <pref:else> tag includes the body of the tag if the given portlet preference is not modifiable. If the preference is modifiable, the body of this tag is skipped. Use to display body content if the content in the <pref:ifModifiable> tag is skipped.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

None.

Example

```
<pref:ifModifiable name="showChannelDescription">
    <input type="checkbox" name="showChannelDescription" value="true" <%=showChannelDescription
    Show Channel Description
    <%somethingToEdit = true;%>
</pref:ifModifiable>

<pref:else>
    Channel description is unavailable.
</pref:else>
```

The Portal Samples contain examples of portlet preference JSP tags. In the Sample Portal, the RSS RSS News Feed Portlet uses the JSP tags, and the Tutorial Portal contains a Portlet Preferences page.

Related Topics

<pref:ifModifiable> Tag

<pref:getPreference> Tag

<pref:getPreferences> Tag

<pref:forEachPreference> Tag

Multichannel JSP Tags

Following are the Portal Multichannel JSP Tags. The are contained in client_taglib.jar.

cscm:default	Renders its contents only if the client's classification has been mapped to "default" or if the client is not recognized.
cscm:not−default	Renders its content only if the client classification has been mapped to anything other than "default."
cscm:recognized	Renders its content if the client has been mapped to any classification, even "default."
cscm:not−recognized	Renders its content if the client has not been mapped to any classification.
cscm:when	Renders its content for any client classification listed in this tag's "client" attribute that matches the client classification name in the HTTP request.
cscm:when−not	Renders its content for any client classification not listed in this tag's "client" attribute. The client name comes from the HTTP request.

<cscm:default> Tag

The <cscm:default> tag is part of the multichannel framework that lets you build portals for mobile devices.

This tag renders its contents only if the client's classification has been mapped to "default" or if the client is not recognized.

A classification of "default" can mean that the request was not resolvable to an existing classification label (defined in <project>\WEB−INF\client−classifications.xml) or that it resolves to an existing classification label called "default." Default is treated by the system as a typical PC−based browser.

Syntax

<tagName>
</tagName>

Attributes

none

Example

```
<cscm:default>
        <img src="images/big_logo.gif">
</cscm:default>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:not−default> Tag

<cscm:recognized> Tag

<cscm:not−recognized> Tag

<cscm:when> Tag

<cscm:when−not> Tag

<cscm:not−default> Tag

The <cscm:not−default> tag is part of the multichannel framework that lets you build portals for mobile devices.

This multichannel tag renders its content only if the client classification has been mapped to anything other than "default."

A classification of "default" can mean that the request was not resolvable to an existing classification label (defined in <project>\WEB−INF\client−classifications.xml) or that it resolves to an existing classification label called "default." Default is treated by the system as a typical PC−based browser.

Syntax

<tagName>
</tagName>

Attributes

none

Example

```
<cscm:not−default>
        <img src="images/small_logo.gif">
</cscm:not−default>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:default> Tag

<cscm:recognized> Tag

<cscm:not−recognized> Tag

<cscm:when> Tag

<cscm:when−not> Tag

<cscm:recognized> Tag

The <cscm:recognized> tag is part of the multichannel framework that lets you build portals for mobile devices.

This tag renders its content if the client has been mapped to any classification, even "default." Classification labels are defined in <project>\WEB−INF\client−classifications.xml. It does not render its content to a client that has not been mapped to a classification in client−classifications.xml.

Syntax

<tagName>
</tagName>

Attributes

none

Example

```
<cscm:recognized>
        <img src="images/small_logo.gif">
</cscm:recognized>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:default> Tag

<cscm:not−default> Tag

<cscm:not−recognized> Tag

<cscm:when> Tag

<cscm:when−not> Tag

<cscm:not−recognized> Tag

The <cscm:not−recognized> tag is part of the multichannel framework that lets you build portals for mobile devices.

This tag renders its content if the client has not been mapped to any classification (defined in <project>\WEB−INF\client−classifications.xml).

Syntax

<tagName>
</tagName>

Attributes

none

Example

```
<cscm:not−recognized>
        <img src="images/big_logo.gif">
</cscm:not−recognized>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:default> Tag

<cscm:not−default> Tag

<cscm:recognized> Tag

<cscm:when> Tag

<cscm:when−not> Tag

<cscm:when> Tag

The <cscm:when> tag is part of the multichannel framework that lets you build portals for mobile devices.

This tag renders its content for any client classification listed in this tag's "client" attribute that matches the client classification name. The client name comes from the HTTP request and is mapped to a client classification in <project>\WEB−INF\client−classifications.xml.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

client

Required (String) − The name(s) of the client classification(s) that will be shown the enclosed content. For more than one client name, use a comma−separated list.

Example

```
<cscm:when client="pda,phone">
        <img src="images/small_logo.gif">
</cscm:when>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:default> Tag

<cscm:not−default> Tag

<cscm:recognized> Tag

<cscm:not−recognized> Tag

<cscm:when−not> Tag

<cscm:when−not> Tag

The <cscm:when−not> tag is part of the multichannel framework that lets you build portals for mobile devices.

This tag renders its content for any client classification not listed in this tag's "client" attribute. The client name comes from the HTTP request and is mapped to a client classification in <project>\WEB−INF\client−classifications.xml.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

client

Required (String) − The name(s) of the client(s) that will not be shown the enclosed content. For more than one client name, use a comma−separated list.

Example

```
<cscm:when−not client="phone">
        <img src="images/pda_only_logo.gif">
</cscm:when−not>
```

Related Topics

Creating Portals for Mobile Devices

<cscm:default> Tag

<cscm:not−default> Tag

<cscm:recognized> Tag

<cscm:not−recognized> Tag

<cscm:when> Tag

Entitlement JSP Tags

Following are the Portal Entitlement JSP Tags. The are contained in auth_taglib.jar.

auth:isAccessAllowed	Provides fine−grained entitlement−setting on application resources for which entitlements are not available by default.
auth:isUserInRole	

<auth:isAcccessAllowed> Tag

This tag provides fine−grained entitlement−setting on application resources for which entitlements are not available by default.

If the result of the entitlement check is not "grant", the body of this tag will be skipped. For convenience, an empty body form of the tag may be used and the return value id will be set true for "grant" decisions.

Using this tag involves the following process:

1. Identify the taxonomy of the resource to be entitled. For example, if you are entitling a link on a JSP, the taxonomy would be: desktop > book > page > portlet > JSP > link.
2. In the WebLogic Administration Portal, create and define a visitor role that will be able to access the resource you are entitling.
3. Add the <auth:isAccessAllowed> tag to your JSP, wrapped around the resource you want to entitle, and set the appropriate tag attributes.

Syntax

<tagName *attribute*="*value*" />

Attributes

resourceId

Required (String) − Represents the application−defined taxonomy (hierarchy of resources) including the resource being requested.

capability

Optional (String) − The requested capability for the resource.

subject

Optional (Subject object) − The Subject for which the request will be evaluated.

roleScope

Optional (int) − The level in the taxonomy at which role policies will be looked for to grant or deny access to the resource. If you do not use this attribute, a role will be looked for up to the enterprise application level. Possible values are:

- <%=EntitlementContraints.GLOBAL_ROLE_INHERITANCE%> − Looks for role policies that are defined in the WebLogic Server Console at a global scope.
- <%=EntitlementContraints.APPLICATION_INHERITANCE%> − Looks for role policies that are defined in the WebLogic Server Console at an application scope.
- <%=EntitlementContraints.ENT_APP_ROLE_INHERITANCE%> − Looks for role policies at an enterprise application and global scope.

- <%=EntitlementContraints.WEBAPP_ROLE_INHERITANCE%> − Looks for role policies at a Web application, enterprise application, and global scope.
- <%=EntitlementContraints.LEAF_NODE_ROLE_INHERITANCE%> − Looks for role policies at a resource leaf node and global scope only.
- <%=EntitlementContraints.HIERARCHICAL_ROLE_INHERITANCE%> − Looks for role policies at each level up the taxonomy: leaf node, application taxonomy, Web application, enterprise application, and global scope.

needContextHandler

Optional (boolean) − Determines whether or not a context handler should be generated.

inheritSecurityPolicy

Optional (boolean) − Determines whether or not to grant or deny access to the resource based on existing security policies. If you do not use this attribute, the default value is "false."

id

Required (String) − The name of the variable that will hold the result of the tag evaluation (grant or deny).

Example

This example sets entitlements on a link on a JSP. The resourceId value is read in from a variable declared earlier in the code. Because of the roleScope value, it looks for existing role policies starting at the leaf node in the resource taxonomy. If the user does not belong to the role policy granting access to this resource, the user will not see the link.

```
<%@ taglib uri="auth.tld" prefix="auth" %>
.
.
.
<auth:isAccessAllowed resourceId="<%=resourceId%>" roleScope="<%=EntitlementContraints.HIERARCH
        <p><a href="HRpersonnel.html">Click here for secure personnel information.</a>
</auth:isAccessAllowed>
```

<auth:isUserInRole> Tag

The <auth:isUserInRole> tag allows you to test the current user s role so you can selectively display content wrapped by the tag. This allows an application to restrict display of application content by requiring authorization of the user accessing the JSP. If used within an entitled portlet, this effectively allows multiple levels of (finer grained) authorization. This tag uses the WLS security SPI within its implementation.

The set of roles evaluated by the <auth:isUserInRole> tag are the Visitor Roles defined by using the WebLogic Administration Portal and global roles defined using the WebLogic Administration Console. Also, any Role Mapping Provider that has roles mapped to the portal resource hierarchy will also be evaluated.

A single call to <auth:isUserInRole> will cause all Visitor Roles for the current web application to be evaluated, so care should be taken as to how large the role set is. The map of computed roles is evaluated at most once per request, but is not cached across requests.

Syntax

<auth:isUserInRole roleName= roleNameToTest roleMap= roleMapVarName />

Attributes

roleName
Required (String) − The name of the role required of the current user.

roleMap
Required (String) The name of the page variable to assign the return java.util.Map value to. The caller may use the roleMap to determine what other evaluated roles the current user is in, if needed.

id
Required (String) − The name of the page variable to assign the return java.lang.Boolean to. If the caller is in the roleName, true is returned.

Example

In this example, a visitor role named VisitorRole has been previously defined in the WebLogic Administration Portal and is being tested for within a JSP page.

```
<%@ taglib uri="auth.tld" prefix="auth" %>
       .
       .
       .
       String aRoleName =  VisitorRole ;

<auth:isUserInRole roleName="<%=aRoleName%>" roleMap="myRoleMap"        id="access"/>

User is in role <%=aRoleName%>: <%=access.booleanValue()%>
       Full set of user roles: <%=myRoleMap.toString()%>

       <auth:isUserInRole roleName="VisitorRole2  roleMap="myRoleMap"        id="access">
       <p>Authorized application content for VisitorRole2</p>
       </auth:isUserInRole>
```

Commerce JSP Tags

Following are the Portal Commerce JSP Tags. The are contained in cat_taglib.jar, productTracking_taglib.jar, and eb_taglib.jar.

cat:catalogQuery	Retrieves a set of catalog items based on keywords or a query expression.
cat:catalogSelector	Retrieves a personalized set of catalog items based on a predefined content selector rule.
cat:getProperty	Retrieves a property for display from either a ProductItem or Category.
cat:iterateThroughView	Iterates through a view of one Catalog item at a time until the end of the View is reached.
cat:iterateViewIterator	Iterates through a collection of Categories or ProductItems.
productTracking:clickProductEvent	Generates a behavior event when a user has clicked (through) a product impression.
productTracking:displayProductEvent	Generates a behavior event when a user has received (viewed) a product impression, typically an image.
eb:smnav	Scrollable model navigation. For use with legacy applications that implement Webflow and run in a compatibility domain. Creates "Previous | 10−19 | Next" style navigation for pages with multiple results.

<cat:catalogQuery> Tag

The <cat:catalogQuery> tag retrieves a set of catalog items based on keywords or a query expression.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

catalogManagerName

Required (String) − The CatalogManagerName that will be used for the query.

id

Required (String) − The variable to hold the query results.

keywords

Optional (String) − Keywords to use in the query.

expression

Optional (String) − The regular expression to use for the query.

maxResults

Optional (int) − The maximum number of results to return.

viewSize

Optional (int) − The view size in the ViewIterator collection of catalog items.

Example

```
<cat:catalogQuery catalogManagerName="MyCatalogManager" id="results"
keywords="<%= keywordString %>" expression="<%= expressionString %>"
maxResults="10">
    <cat:iterateViewIterator iterator="<%= myIterator %>" id="view"
    returnType="com.beasys.commerce.ebusiness.catalog.ViewIterator" iterateByView="true">
    <%= view.toString() %>
    </cat:iterateViewIterator>
</cat:catalogQuery>
```

Related Topics

Building a Commerce Application

<cat:catalogSelector> Tag

<cat:catalogSelector> Tag

The <cat:catalogSelector> tag retrieves a personalized set of catalog items based on a predefined content selector rule.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

rule

Optional (String) − The name of the content selector rule used to retrieve the personalized list of catalog items.

id

Required (String) − The variable to store the array of catalog items found.

Example

```
<cat:catalogSelector id="items" selector="outdoor" />
<cat:iterateThroughView iterator="<%= items %>" id="item"
    returnType="com.beasys.commerce.ebusiness.catalog.ProductItem"viewIndex="new Integer(0)">
    <%= item.getKey().toString() %>
</cat:iterateThroughView>
```

Related Topics

Building a Commerce Application

Creating Content Selectors

<cat:catalogQuery> Tag

<cat:getProperty> Tag

Use the <cat:getProperty> tag to retrieve a property for display from either a ProductItem or Category. The property can either be an explicit property (a property that can be retrieved using a get method on the Catalog item) or an implicit property (a property available through the ConfigurableEntity getProperty methods on the Catalog item). The tag first checks to see if the specified property can be retrieved as an explicit property. If it cannot, the specified property is retrieved as an implicit property.

Syntax

<tagName *attribute*="*value*" / >

Attributes

getterArgument

Optional (String) − Denotes a reference to an object supplied as an argument to an explicit property getter method. May also be used to obtain implicit or custom properties that are defined using the property set framework, in which case the getterArgument would be the scope name for the property set.

The object must be presented in the form <%= getterArgumentReference %> and must be a run−time expression.

id

Optional (String) − id="newInstance". If the id attribute is supplied, the value of the retrieved property will be available in the variable name to which id is assigned. Otherwise, the value of the property is inlined.

object

Required (Catalog item) − Denotes a reference to a ProductItem or Category object that must be presented in the form <%= objectReference %>.

propertyName

Required (String) − propertyName="propertyName". Name of the property to retrieve. If the property is explicit, it may be one of the values shown in the following table.

returnType

Optional (String) − returnType="returnType". If the id attribute is supplied, declares the type of the variable specified by the id attribute.

propertyName Values

Property Name	***Catalog Item Type***
"contributor | coverage | creationDate | creator | description |	Catalog Item

image | key | language | modifiedDate | name | publisher | relation | rights | source"	(common properties)
"jsp"	Category
"availability | currentPrice | format | jsp | msrp | shippingCode | taxCode | type | visible"	ProductItem

Example1

This example retrieves the detail JSP information from an existing ProductItem:

```
<cat:getPropertyobject="<%= item %>"propertyName="Jsp" getterArgument= "<%= new Integer(Product
id="detailJspInfo"returnType="com.beasys.commerce.ebusiness.catalog.JspInfo"/>
```

Example 2

This example shows how to use the getterArgument attribute to obtain an implicit or custom property for a property set/schema with the following characteristics:

- Name: MyCatalog
- PropertyName: color

Note: Because the getterArgument must be a run−time expression, we assign MyCatalog to a String variable and use the variable as the value to the getterArgument.

```
<% String myPropertySetName = "MyCatalog"; ProductItem myProductItem= .....; // reference to a
<cat:getProperty
    object="<%=myProductItem%>
    propertyName="color"
    getterArgument="<%=myPropertySetName%>"
/>
```

Related Topics

Building a Commerce Application

<cat:iterateThroughView> Tag

The <cat:iterateThroughView> tag iterates through a view of one Catalog item at a time until the end of the View is reached. If you do not specify a specific View (by index) through which to iterate, the current View of the ViewIterator is used. This tag does not reset the state of the ViewIterator upon completion.

Syntax

<tagName *attribute*="*value*">
</tagName>

Attributes

id

Required (String) − id="newInstance". The value of the current iterated object will be available in the variable name to which the id is assigned.

iterator

Required (ViewIterator) − Denotes a reference to a ViewIterator object that must be presented in the form <%= iteratorReference %>.

returnType

Optional (String) − returnType="returnType". Declares the type of the variable specified by the id attribute. Defaults to java.lang.Object.

viewIndex

Optional (Integer) − Specifies the index of the View (relative to the start of the ViewIterator) through which to iterate. The referenced object must be presented in the form <%= viewIndexIntegerReference %>.

Example

This example displays the keys of all the ProductItems contained in the current View of a specified ViewIterator:

```
<cat:iterateThroughView
iterator="<%= myIterator %>"
id="item"
returnType="com.beasys.commerce.ebusiness.catalog.ProductItem">
<%= item.getKey().toString() %>
</cat:iterateThroughView>
```

Example 2

This example displays the keys of all the ProductItems contained in the first View of a specified ViewIterator:

```
<cat:iterateThroughView
```

```
iterator="<%= myIterator %>"
id="item"
returnType="com.beasys.commerce.ebusiness.catalog.ProductItem"viewIndex="new Integer(0)">
<%= item.getKey().toString() %>
</cat:iterateThroughView>
```

Related Topics

Building a Commerce Application

<cat:iterateViewIterator> Tag

<cat:iterateViewIterator> Tag

Use the <cat:iterateViewIterator> tag to iterate through a ViewIterator. A ViewIterator is an iterator over a potentially large collection of remote data that is broken up into a series of fixed sized Views. ViewIterators are returned from all Catalog service API methods that may potentially return a large set of ProductItems or Categories. This tag allows you to iterate the ViewIterator one item (ProductItem or Category) at a time (the default behavior) or by an entire View (fixed size set of ProductItems or Categories) at a time. This tag does not reset the state of the ViewIterator upon completion.

Syntax

<tagName *attribute*="*value*">
</tagName>

id

Required (String) − id="newInstance". The value of the current iterated object will be available in the variable name to which the id is assigned.

iterator

Required (ViewIterator) − Denotes a reference to a ViewIterator object. Must be presented in the form <%= iteratorReference %>.

iterateByView

Optional (String) − iterateByView="{true|false}". Specifies whether to iterate the ViewIterator by View or by Catalog item. If not specified, the ViewIterator will be iterated by Catalog item.

returnType

Optional (String) − returnType="returnType". Declares the type of the variable specified by the id attribute. Defaults to java.lang.Object. If iterateByView is true, the type is assumed to be com.beasys.commerce.ebusiness.catalog.View.

Example 1

This example displays the keys of all Categories in a ViewIterator:

```
<cat:iterateViewIterator iterator="<%= myIterator %>"
id="category"
returnType="com.beasys.commerce.ebusiness.catalog.Category">
<%= category.getKey().toString() %>
</cat:iterateViewIterator>
```

Example 2

This example displays all the Views contained within a ViewIterator:

```
<cat:iterateViewIterator iterator="<%= myIterator %>"
```

```
id="view"
returnType="com.beasys.commerce.ebusiness.catalog.ViewIterator"iterateByView="true">
<%= view.toString() %>
</cat:iterateViewIterator>
```

Related Topics

Building a Commerce Application

<cat:iterateThroughView> Tag

<productTracking:clickProductEvent> Tag

The <productTracking:clickProductEvent> tag is used to generate a behavior event when a user has clicked (through) a product impression (Web content related to a product). This tag returns a URL query string containing event parameters, which is then used when forming the complete URL that hyperlinks the content. At least one of sku, categoryId, or documentId is required.

Syntax

<tagName *attribute*="*value*" />

Attributes

applicationName

Optional (String) − The webApp or application name, if applicable. Can be used to separate data when multiple storefronts are hosted on the same server (or persisted to the same database).

categoryId

Optional (String or Category object) − Category of the product associated with the content displayed, if applicable.

documentId

Required (String) − Name of the item that is displayed, if applicable (that is, an image URL or banner ad ID).

documentType

Optional (String) − Type or category of the item that is displayed, if applicable.

sku

Optional (String or ProductItem) − Object ID of the product associated with the content item that is displayed, if applicable. The sku is not normally required unless neither categoryId nor documentId is specified.

userId

Optional (String) − Name of the user for which the content was retrieved. If the optional value is not provided, it will be set to the value of the request.getRemoteUser().

Example

This example demonstrates a clickthrough example going to the Webflow servlet. This link will cause a clickthrough content event to be generated and also display the indicated content. This example shows how to generate a ClickProductEvent having a document ID using the product name (productItem.getName()) and SKU of the product s identifier.

<%

```
detailsUrl = WebflowJSPHelper.createWebflowURL(pageContext, "itemsummary.jsp", "link(" + detail
"&" + HttpRequestConstants.CATALOG_ITEM_SKU + "=" + productItem.getKey().getIdentifier() + "&"
HttpRequestConstants.CATALOG_CATEGORY_ID + "=" + category.getKey().getIdentifier() + "&" +
HttpRequestConstants.DOCUMENT_TYPE + "=" + detailsLink, true);
%>
<productTracking:clickProductEvent id="url" documentId="<%= productItem.getName() %>"
sku="<%= productItem.getKey().getIdentifier() %>" />
<%
detailsUrl = detailsUrl + "&" + url;
%>
<a href="<%= detailsUrl %>">
```

Related Topics

<productTracking:displayProductEvent> Tag

<BehaviorTracking:clickContentEvent> Tag

<BehaviorTracking:displayContentEvent> Tag

<productTracking:displayProductEvent> Tag

The <productTracking:displayProductEvent> tag is used to generate a behavior event when a user has received (viewed) a product impression, (typically an image). At least one of sku, categoryId, or documentId is required.

Syntax

<tagName *attribute*="*value*" />

Attributes

applicationName

Optional (String) − The webApp or application name, if applicable. Can be used to separate data when multiple storefronts are hosted on the same server (or persisted to the same database).

categoryId

Optional (String or Category object) − Category of the product associated with the content displayed, if applicable.

documentId

Optional (String) − Name of the item that is displayed, if applicable (that is, an image URL or banner ad ID).

documentType

Optional (String) − Type or category of the item that is displayed, if applicable.

Suggestions:

- DisplayProductEvent.CATEGORY_BROWSE
- DisplayProductEvent.ITEM_BROWSE
- DisplayProductEvent.CATEGORY_VIEW
- DisplayProductEvent.BANNER_AD_PROMOTION

sku

Optional (String or ProductItem) − Object ID of the product associated with the content item that is displayed, if applicable. The sku is not normally required unless neither categoryId nor documentId is specified.

Example

This example shows code that would follow the retrieval of a catalog item. The <BehaviorTracking:displayProductEvent> tag generates an event and passes the document's ID, type and SKU number of the product item.

```
<productTracking:displayProductEvent documentId="<%= item.getName() %>"
```

```
documentType="<%= DisplayProductEvent.ITEM_BROWSE %>"
sku="<%= item.getKey().getIdentifier() %>" />
```

Related Topics

<productTracking:clickProductEvent> Tag

<BehaviorTracking:clickContentEvent> Tag

<BehaviorTracking:displayContentEvent> Tag

<eb:smnav> Tag

For use with legacy applications that implement Webflow and run in a compatibility domain. The <eb:smnav> tag, which is added to your portal Web project when you install commerce in your application, enables scrollable model navigation by creating "Previous | 10−19 | Next" style navigation for pages with multiple results.

This tag relies on a Pipeline Session containing a ScrolableModel object on the PipelineSessionConstants.SCROLLABLE_MODEL key.

Syntax

<tagName *attribute*="*value*" />

Attributes

origin

Optional (String) − The current JSP.

event

Optional (String) − The name of the link configurable in the Webflow as the visitor clicks the "Next" or "Previous" links.

pageindex

Optional (String) − The index of the current page to display.

prevstring

Optional (String) − The localized name for the "Next" string, which could be as simple as ">".

nextstring

Optional (String) − The localized name for the "Previous" string, which could be as simple as "<".

Example

```
<eb:smnav origin="orderhistory.jsp"
          event="link.ViewOrderHistory"
          prevstring="Previous"
          nextstring="Next"
          pageindex="<%= pageIndexString %>" />
```

Related Topics

Building a Commerce Application

Utilities JSP Tags

Following are the Portal Utilities JSP Tags. The are contained in es_taglib.jar.

es:convertSpecialChars	Lets you display special characters as literal strings for display in a browser.
es:counter	Performs Java `for` loop logic.
es:date	Gets a date− and time−formatted String based on the user's time zone preference.
es:forEachInArray	Iterates over an array of returned objects.
es:isNull	Checks if a value is null. For String values, checks to see if the value is empty (zero length).
es:notNull	Checks if a value is not null. For String values, checks to see if the value is greater than zero length.
es:transposeArray	Transposes a standard [row][column] array to a [column][row] array.
es:uriContent	Pulls content from a specified URL.

<es:convertSpecialChars> Tag

The <es:convertSpecialChars> tag Lets you display special characers as literal strings for display in a browser. For example, the following sentence must be converted because it uses the "<" and ">"characters, which signify tag opening and closing to the browser: "Enter <filename> here: "

Syntax

<tagName *attribute*="*value*" />

Attributes

string

Required (String) − The string to be converted.

Example

This example allows a string containing greater−than and less−than brackets to be rendered in HTML.

```
<es:convertSpecialChars string="<filename>"/>
```

<es:counter> Tag

The <es:counter> tag is used to create a `for` loop.

Syntax

<tagName *attribute*="*value*" />

Attributes

type

Optional (String) − The type of the counter. Possible values are Integer or Long. Default is Integer.

id

Required (String) − A unique name for the variable.

minCount

Required (int) − The start position for the loop.

maxCount

Required (int) − The end position for the loop.

Example

This example runs a `for` loop beginning at zero and ending at 10, printing the variable value each time.

```
<es:counter id="iterator" minCount="0" maxCount="10">
<% System.out.println(iterator);%>
</es:counter>
```

<es:date> Tag

Use the <es:date> tag to get a date− and time−formatted String based on the user's time zone preference.

Syntax

<tagName *attribute*="*value*" />

Attributes

timeZoneId

Optional (String) − Defaults to the time zone on the server.

formatStr

Optional (String) − A date and time format string that adheres to the `java.text.SimpleDateFormat`. The default value is "MM dd yyyy hh:mm a".

Example

```
<es:date formatStr="MMMM dd yyyy" timeZoneId="MST" />
```

<es:forEachInArray> Tag

The <es:forEachInArray> tag is used to iterate over an array. It is often used with other WebLogic Portal JSP tags to iterate over arrays of returned objects.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The variable for each value in the array as an object of type Type.

type

Required (String) − The type of each value in the array, which must match the elements of the array.

array

Required (Object array) − The array to iterate over.

counterId

Optional (String) − The position in the array as an Integer.

Example 1

This example iterates over an array and prints the results.

```
<es:forEachInArray id="item" array="<%=items%>" type="String" counterId="i">
<% System.out.println("items[" + i + "]: " + item);%>
</es:forEachInArray>
```

Example 2

This example, used in conjunction with the <pz:contentSelector> tag to retrieve personalized Web content for a user, iterates over an array of content objects retrieved from a content management system for display to the user.

```
<profile:getProfile profileKey="bob" profileId="myProfile" scope="session"/>
<pz:contentSelector rule="PremierCustomerSpotlight" id="docs" />
<ul>
<es:forEachInArray array="<%=docs%>" id="aDoc" type="com.bea.p13n.content.Node">
<li>The document title is: <cm:printproperty id="aDoc" name="Title" encode="html" />
</es:forEachInArray>
</ul>
```

Related Topics

<es:isNull> Tag

The <es:isNull> tag is used to check if a value is null. In the case of a String, the <es:isNull> tag is used to check if the String is null or is empty (zero length). An empty string is null.

Syntax

<tagName *attribute*="*value*" />

Attributes

item

Required (Object) − The variable to evaluate.

Example

The following example prints a message if the value is null.

```
<es:isNull item="<%=value%>">
Error: the value is null.
</es:isNull>
```

Related Topics

<es:notNull> Tag

<es:notNull> Tag

The <es:notNull> tag is used to check if a value is not null. In the case of a String, the <es:notNull> tag is used to check if the String is not null or has a value (more than zero length).

Syntax

<tagName *attribute*="*value*" />

Attributes

item

Required (Object) − The variable to evaluate.

Example

This example prints a message if an object is not null.

```
<es:notNull item="<%=value%>">
The value is not null.
</es:notNull>
```

Related Topics

<es:isNull> Tag

<es:transposeArray> Tag

The <es:transposeArray> tag is used to transpose a standard [row][column] array to a [column][row] array.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The variable that holds the [c][r] array.

type

Required (String) − The type of variable in the [r][c] array, such as String.

array

Required (Object array) − The variable that holds the [r][c] array.

Example

```
<es:transposeArray id="byColumnRow" array="<%=byRowColumn%>" type="String">
</es:transposeArray>
```

<es:uriContent> Tag

The <es:uriContent> tag is used to pull content from a URL. It is best used for grabbing text−heavy pages.

If you combine HTML pages with relative URL s, you must fully qualify them to the correct host in each URL, or else images (on other resources) may not be retrieved properly by the browser.

Syntax

<tagName *attribute*="*value*" />

Attributes

id

Required (String) − The variable that holds the downloaded content of the URI.

uri

Required (String) − The fully qualified URI from which to get the content.

Example

This example retrieves and prints content from a URL.

```
<es:uriContent id="uriContent" uri="http://www.bea.com/index.html">
<%
out.print(uriContent);
%>
</es:uriContent>
```