Vizrt Community Expansion

Taglib Reference

3.8.0.130433


vizrt

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Last Updated

13.09.2012


vizrt

Table of Contents

1 Introduction

Welcome to the Vizrt Community Expansion Taglib reference guide.

2 articleextra Tag Library

2.1 articleextra:articleType

The tag gets the article type object.

Syntax

```
<articleextra:articleType
   id="..."
   name="..."?
   typeName="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the group information from. This can be a PresentationArticle or Article object.

`typeName`
The name of the article type you want to get.

2.2 articleextra:use

The tag does the same as the escenic article:use. It has an extra options called state. With this tag you can also display draft, submitted, approved and deleted articles

Syntax

```
<articleextra:use
   article="..."?
   articleId="..."?
   name="..."?
   property="..."?
   source="..."?
   sourceId="..."?
   state="...">
   ...
</articleextra:use>
```

Attributes

`article`
The article used to be set as default article.

`articleId`
Article id to identify the article to be set as default article.

name
Name of an bean to specify article to be set as default article.

property
Property used to find the article to be set as default article.

source
Source of the article to set as default article.

sourceId
Source id used to identify the article to be set as default article.

state, mandatory
Diplays only articles with the given state. Options:
draft,submitted,approved and deleted. You can use multiple states.

3 auth Tag Library

3.1 auth:hasPermission

User/permission tag to check the user permissions, the permissions must be set as a comma separated list. Class returns TRUE if the user has the given permission and FALSE otherwise if an 'id' was given as an argument. If the permissions are set for the given user then the body content will be rendered, otherwise the body content will not be rendered.

```
<auth:hasPermission
      permissions="CONTENTADD"
      sectionId="2201">
  User has permission
</auth:hasPermission>
```

Syntax

```
<auth:hasPermission
   id="..."?
   permissions="..."
   section="..."?
   sectionId="..."?
   user="..."?
   userId="..."?>
   ...
</auth:hasPermission>
```

Attributes

`id`, no runtime expressions
The name of the scripting variable.

`permissions`, mandatory
The name of the permissions to check the user has. May be a list of comma separated permissions.

`user`
The name of the scripting-variable where to get the user from. This can be a UserData, PresentationPerson or Person object. Default it uses the logged in user.

`userId`
Id from the escenic user. Default it uses the logged in user.

`section`
The name of the scripting-variable where to get the section form. This can be a PresentationSection or Section object. Default it uses the section your in.

`sectionId`
The id from the section you want to check. Default it uses the section your in.

3.2 auth:hasRole

User/role tag to check the role for a certain user. Class returns TRUE if the user has the given role and FALSE otherwise if an 'id' was given as an argument. If the role is set for the given user then the body content will be rendered, otherwise the body content will not be rendered.

```
<auth:hasRole id="testRole" role="ADMIN"/>
<util:valueof param="testRole"/>
```

Syntax

```
<auth:hasRole
   id="..."?
   role="..."
   section="..."?
   sectionId="..."?
   user="..."?
   userId="..."?>
   ...
</auth:hasRole>
```

Attributes

`id`, no runtime expressions
The name of the scripting-variable.

`role`, mandatory
The name of the role the user should have.

`user`
The name of the scripting-variable where to get the user from. This can be a UserData, PresentationPerson or Person object. Default it uses the logged in user.

`userId`
Id from the escenic user. Default it uses the logged in user.

`section`
The name of the scripting-variable where to get the section form. This can be a PresentationSection or Section object. Default it uses the section your in.

`sectionId`
The id from the section you want to check. Default it uses the section your in.

3.3 auth:roles

Retrieves all the roles of the user with the given name or userId specified.

Syntax

```
<auth:roles
   id="..."
   name="..."?
```

```
userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable. This tag will set a **`java.util.List`** of **`com.ndc.auth.api.domain.UserRole`** objects to this scripting variable

`name`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The ID of the user to find roles

4 community Tag Library

4.1 community:user

Retrieves the User from the community with the given name, userId, sectionId, articleId specified. If none of these are given it gets the logged in user.

```
<community:user id="user" name="section"/>
<util:valueof param="user.article.fieldElement(USERNAME)"/>
```

Syntax

```
<community:user
   articleId="..."?
   id="..."
   name="..."?
   sectionId="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the user information from. This can be a PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article object.

`userId`
The id of the user you want to get.

`sectionId`
The id of the user his homesection.

`articleId`
The id of the user his profileArticle.

4.2 community:isCurrentUser

Checks if the current user equals the given user.

```
<community:isCurrentUser name="user">
  Same user
</community:isCurrentUser>
```

Syntax

```
<community:isCurrentUser
   id="..."?
   name="..."?
   userId="..."?>
   ...
```

```
</community:isCurrentUser>
```

Attributes

`id`, no runtime expressions
Name of the new scripting-variable.

`name`
Name of the scripting-variable where to check the current user with. This can be a PresentationUser, PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article object.

`userId`
The id of the user you want to check.

4.3 community:isNotCurrentUser

Checks if the current user not equals the given user.

Syntax

```
<community:isNotCurrentUser
   id="..."?
   name="..."?
   userId="..."?>
   ...
</community:isNotCurrentUser>
```

Attributes

`id`, no runtime expressions
Name of the new scripting-variable.

`name`
Name of the scripting-variable where to check the current user with. This can be a PresentationUser, PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article object.

`userId`
The id of the user you want to check.

4.4 community:group

etrieves the group from the community with the given name, groupId, sectionId, articleId specified.

Syntax

```
<community:group
   articleId="..."?
   groupId="..."?
   id="..."
   name="..."?
   sectionId="..."?/>
```

Attributes

id, mandatory, no runtime expressions
Name of the scripting-variable.

name
Name of the scripting-variable where to get the group information from. This can be a PresentationSection, Section, PresentationArticle or Article object.

groupId
The id of the group you want to get.

sectionId
The id of the group his homesection.

articleId
The id of the group his profileArticle.

4.5 community:groups

Retrieves all the groups from the user with the given name or userId specified.

```
<community:groups id="groups" name="user"/>
<logic:empty name="groups">
  No groups found
</logic:empty>
<logic:notEmpty name="groups">
  <logic:iterate
        id="group"
        name="groups"
        type="com.ndc.presentation.PresentationGroup">
    <util:valueof param="group.section.url"/>
    <util:valueof param="group.article.fieldElement(NAME)"/>
  </logic:iterate>
</logic:notEmpty>
```

Syntax

```
<community:groups
   id="..."
   name="..."?
   type="..."?
   userId="..."?/>
```

Attributes

id, mandatory, no runtime expressions
Name of the scripting-variable.

name
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

userId
The id of the user you want to get.

type
The type of groups you want to receive. This can be "accepted","invited","pending","admin". Default is "accepted"

4.6 community:friends

Retrieves all the friends from the user with the given name or userId specified. You can set a type of friend.

```
<community:friends id="friends" name="user"/>
<logic:empty name="friends">
  No real friends found
</logic:empty>
<logic:notEmpty name="friends">
  <logic:iterate
      id="friend"
      name="friends"
      type="com.ndc.presentation.PresentationUser">
    <util:valueof param="friend.section.url"/>
    <util:valueof param="friend.article.fieldElement(USERNAME)"/>
  </logic:iterate>
</logic:notEmpty>
```

Syntax

```
<community:friends
   id="..."
   name="..."?
   type="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user you want to get.

`type`
The type of friends you want to receive. This can be "accepted","candidate","pending". Default is "accepted"

4.7 community:groupMembers

Retrieves all the members from the group with the given name or groupId specified. You can set a type of member.

```
<community:groupMembers
      id="members"
      name="group"/>
<logic:empty name="members">
  No real members found
</logic:empty>
<logic:notEmpty name="members">
  <logic:iterate
       id="member"
       name="members"
       type="com.ndc.presentation.PresentationUser">
    <util:valueof param="member.section.url"/>
    <util:valueof param="member.article.fieldElement(USERNAME)"/>
```

```
    </logic:iterate>
  </logic:notEmpty>
```

Syntax

```
<community:groupMembers
   groupId="..."?
   id="..."
   name="..."?
   type="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`name`
Name of the scripting-variable where to get the group information from. This can be a PresentationGroup, PresentationSection, Section, PresentationArticle or Article object.

`groupId`
The id of the group you want to get.

`type`
The type of members you want to receive. This can be "accepted","candidate","invited". Default is "accepted"

4.8 community:isFriend

Checks if a user is a friend or almost is a friend.

```
  <community:isFriend id="isFriend" friend="user"/>
  <logic:equal name="isFriend" value="false">
    <a href="#">Add as friend</a>
  </logic:equal>
```

Syntax

```
<community:isFriend
   friend="..."?
   friendId="..."?
   id="..."?
   name="..."?
   type="..."?
   userId="..."?>
   ...
</community:isFriend>
```

Attributes

`id`, no runtime expressions

`name`

`userId`

`friendId`

friend

type

4.9 community:isGroupMember

Check if the user is member of the group. Default it gets the loggedin user.

```
<community:isGroupMember
      id="isGroupOwner"
      name="group"
      type="admin" />
<logic:equal name="isGroupOwner" value="false">
  Only for admin
</logic:equal>
```

Syntax

```
<community:isGroupMember
   groupId="..."?
   id="..."?
   name="..."?
   type="..."?
   user="..."?
   userId="..."?
   value="..."?>
   ...
</community:isGroupMember>
```

Attributes

id, no runtime expressions
Name of the scripting-variable.

name
Name of the scripting-variable where to get the group information from. This can be a PresentationGroup, PresentationSection, Section, PresentationArticle or Article object.

groupId
The id of the group you want to get.

user
The object of the user you want to get. This can be a PresentationUser, PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article

userId
The id of the user you want to get.

type
The type of members you want to receive. This can be "all","invited","pending","admin". Default is "all"

value
This can be "true" or "false". Default is "true"

4.10 community:friendsNotifications

Retrieves type friends notifications from the user with the given name or userId specified. Default the loggedin use will be used.

```
<community:friendsNotifications
    id="unreadResignedFriends"
    name="user"
    type="resigned" />
<logic:iterate
    id="item"
    name="unreadResignedFriends"
    type="com.ndc.presentation.PresentationUser">
  <util:valueof param="item.article.fieldElement(USERNAME)"/>
</logic:iterate>
```

Syntax

```
<community:friendsNotifications
   id="..."
   max="..."?
   name="..."?
   sort="..."?
   type="..."
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article object.

`userId`
The id of the user you want to get.

`type`, mandatory
Type can be "accepted", "declined" or "resigned".

`sort`
You may sort on the date of when the friend notification was made. – **`date`** or **`date`**.

Per default, the tag will return results with the newest item first (this is also what you get if you write **`-date`** to the sort field). However, you may also write **`date`** to get the oldest first.

`max`
max items to return

4.11 community:groupsNotifications

Retrieves type group notifications from the user with the given name or userId specified. Default the loggedin use will be used.

```
<community:groupsNotifications
    id="unreadDeclinedGroups"
    name="user"
    type="Declined"/>
<logic:iterate
    id="group"
    name="unreadDeclinedGroups"
    type="com.ndc.presentation.PresentationGroup">
  <util:valueof param="group.article.fieldElement(NAME)"/>
</logic:iterate>
```

Syntax

```
<community:groupsNotifications
   id="..."
   max="..."?
   name="..."?
   sort="..."?
   type="..."
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson, Person, PresentationSection, Section, PresentationArticle or Article object.

`userId`
The id of the user you want to get.

`type`, mandatory
The type of Notifications you want to receive. This can be "accepted","Declined","Resigned"

`sort`
You may sort on the date of when the friend notification was made. – **`date`** or **`date`**.

Per default, the tag will return results with the newest item first (this is also what you get if you write **`-date`** to the sort field). However, you may also write **`date`** to get the oldest first.

`max`
max items to return

4.12 community:groupMembersNotifications

Retrieves type groupMember notifications from the group with the given name or groupId specified.

```
<community:groupMembersNotifications
    id="unreadResigned"
    name="group"
    type="Resigned"/>
```

```
<logic:iterate
    id="user"
    name="unreadResigned"
    type="com.ndc.presentation.PresentationUser">
  <util:valueof param="user.article.fieldElement(USERNAME)"/>
</logic:iterate>
```

Syntax

```
<community:groupMembersNotifications
   groupId="..."?
   id="..."
   name="..."?
   type="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`name`
Name of the scripting-variable where to get the group information from. This can be a PresentationGroup, PresentationSection, Section, PresentationArticle or Article object.

`groupId`
The id of the group you want to get.

`type`, mandatory
The type of Notifications you want to receive. This can be "accepted","declined","resigned"

5 msg Tag Library

5.1 msg:messages

Returns collection of messages with given status and type.

Syntax

```
<msg:messages
   direction="..."?
   id="..."
   public="..."?
   state="..."?
   status="..."?
   type="..."?
   user="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`direction`
FROM or TO(default)

`state`
DRAFT or SENT(default)

`user`
Name of the scripting-variable where to get the user information from. This must be a PresentationUser, PresentationPerson or Person object. Default it uses the logged in user.

`userId`
Id from the escenic user. Default it uses the logged in user.

`status`
DRAFT, SUBMITTED, SENT, RECEIVED, READ, REPLIED. In case of multiple statusses use comma to sepperate. Default it uses all.

`public`
true or false. Default is both

`type`
TEXTMESSAGE (default) or VIDEOMESSAGE

5.2 msg:message

Returns specific message with given messageId or articleId. Only one of messageId and articleId must be given.

Syntax

```
<msg:message
   articleId="..."?
   id="..."
   messageId="..."?
   name="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`messageId`
The id from the messagen you want to see.

`name`
Name of the scripting-variable where to get the article information from. This must be an PresentationArticle or Article object.

`articleId`
The id from the article you want to see.

5.3 msg:nrOfMessages

Returns the number of messages for certain user by given type.

```
messages recieved (<msg:nrOfMessages
                        user="userProfile"
                        direction="TO"/>)
```

Syntax

```
<msg:nrOfMessages
   direction="..."?
   id="..."?
   public="..."?
   state="..."?
   status="..."?
   type="..."?
   user="..."?
   userId="..."?/>
```

Attributes

`id`, no runtime expressions
Name of the scripting-variable.

`direction`, no runtime expressions
FROM or TO(default)

`state`
DRAFT or SENT(default)

`user`
Name of the scripting-variable where to get the user information from. This must be a PresentationUser, PresentationPerson or Person object. Default it uses the logged in user.

userId
Id from the escenic user. Default it uses the logged in user.

status
DRAFT, SUBMITTED, SENT, RECEIVED, READ, REPLIED. In case of multiple statusses use comma to sepperate. Default it uses all.

public
true, false. default both.

type
TEXTMESSAGE (default) or VIDEOMESSAGE

5.4 msg:messagingOption

Returns Value of certain messagingOption for given user in a publication.

Syntax

```
<msg:messagingOption
   id="..."?
   name="..."?
   optionName="..."
   userId="..."?>
   ...
</msg:messagingOption>
```

Attributes

id, no runtime expressions
Name of the scripting-variable.

name
Name of the scripting-variable where to get the user information from. This must be a PresentationUser, PresentationPerson or Person object. Default it uses the logged in user.

userId
Id from the escenic user. Default it uses the logged in user.

optionName, mandatory
Name of the option to retreive: PRIVATE_FRIENDREQUEST, PRIVATE_MESSAGE, PRIVATE_REACTION, PRIVATE_GROUPMESSAGE, VIDEYOU_FRIENDREQUEST, PRIVATE_RECEIVEDVIDEYOU, VIDEYOU_RECEIVEDVIDEYOU, VIDEYOU_MESSAGE, VIDEYOU_REACTION, PRIVATE_VIDEYOUWATCHED, VIDEYOU_VIDEYOUWATCHED, PRIVATE_VIDEYOUNEWS, VIDEYOU_VIDEYOUNEWS, PRIVATE_KRABBEL, DIERZ_KRABBEL

5.5 msg:repliesOnMessage

Returns all replies on the given message.

Syntax

```
<msg:repliesOnMessage
   articleId="..."
   id="..."
   name="..."?
   public="..."?
   type="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`, no runtime expressions
Name of the scripting-variable where to get the article information from. This must be an PresentationArticle or Article object.

`articleId`, mandatory
The id from the article you want to see.

`type`
TEXTMESSAGE(default) or VIDEOMESSAGE

`public`
true or false. Default is both.

5.6 msg:changeRecipientStatus

WARNING: This a tag that does an update. It updates the message status of the given recipient.

Syntax

```
<msg:changeRecipientStatus
   name="..."
   status="..."
   userId="..."?/>
```

Attributes

`name`, mandatory, no runtime expressions
Name of the scripting-variable. It must be a Message object.

`userId`
The id of the user. Default it gets the logged in user.

`status`, mandatory
The status where you want to change it to. DRAFT, SUBMITTED, SENT, RECEIVED, READ or REPLIED.

6 pager Tag Library

6.1 pager:iterator

The tag iterates over the active page items. The size is set in the PagerSource

Syntax

```
<pager:iterator
   id="..."
   name="..."
   type="..."?>
   ...
</pager:iterator>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting variable.

`name`, mandatory
Name of the scripting-variable where to get the PagerSource object.

`type`
The object type that is in the PagerSource object.

6.2 pager:actions

It is a iterator that gets all the Pager actions. The page numbers/actions

Syntax

```
<pager:actions
   id="..."
   name="...">
   ...
</pager:actions>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`name`, mandatory
Name of the scripting-variable where to get the PagerSource object.

6.3 pager:setPage

Set the active page for the pager. You can set it your self with the page attribute or set the parameter attribute where he can get it from.

Syntax

```
<pager:setPage
   default="..."?
   name="..."
   page="..."?
   parameter="..."?>
   ...
</pager:setPage>
```

Attributes

`name`, mandatory, no runtime expressions
Name of the scripting-variable where to get the PagerSource object.

`page`
The active page number/identifier.

`parameter`
The url parameter where to get active page number/identifier

`default`
The page number/identifier when no parameter or page is available

6.4 pager:pagerSource

Converts a Collection to a PagerSource.

Syntax

```
<pager:pagerSource
   id="..."
   maxActionItems="..."
   name="..."
   pageSize="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`, mandatory
Name of the scripting-variable where to get the collection object.

`pageSize`, mandatory

`maxActionItems`, mandatory
The maximum number of pager items it creates.

7 qualification Tag Library

7.1 qualification:count

Gets the given qualification type count from article or User. Default it gets it form the currentArticle.

```
<qual:count id="nrOfFavorites" type="favorite" />
<util:valueof param="nrOfFavorites"/>
```

Syntax

```
<qualification:count
   articleId="..."?
   id="..."?
   name="..."?
   type="..."
   userId="..."?/>
```

Attributes

`id`, no runtime expressions

`name`
Name of the scripting-variable where to get favorites form. This can be a PresentationUser, PresentationPerson, Person, PresentationArticle or Article object.

`userId`
Name of the scripting-variable. If it is empty it prints.

`articleId`
The id of the article.

`type`, mandatory
The type of qualification you want to get. StarRating, Flagging, Favorite

7.2 qualification:ranking

The prefered way of using this tag is as follows:

```
<%
Calendar calendar = Calendar.getInstance();
calendar.set(2009, 0, 01);
Date fromDate = calendar.getTime();
calendar.set(2009, 05, 20);
Date toDate = calendar.getTime();
%>
<qual:ranking
  id="myRanking"
  type="StarRating"
  includeArticleTypes="blog, essay"
  from="<%= fromDate %>"
  to="<%= toDate %>"
```

```
    max="5"
  />
```

You may also use the **`userIds`** parameter to pass the IDs (seperated with comma) of the users you want ranking information from.

The old way of using the tag with composing a comma seperated list of article IDs enclosed in brackets and passing it to the **`articleId`** parameter is deprecated.

Syntax

```
<qualification:ranking
   articleId="..."?
   articleIds="..."?
   from="..."?
   id="..."
   includeArticleTypes="..."?
   max="..."?
   sort="..."?
   to="..."?
   type="..."
   userIds="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`articleId`
Deprecated, see tag description.

`articleIds`
Attribute accepts ',' separated String of article Ids.

`includeArticleTypes`
Comma separated list of article type IDs.

`userIds`
Comma separated list of user IDs.

`type`, mandatory
StarRating

`from`
Expects a java.util.Date object from when the first ranking of the returned list was made.

`to`
Expects a java.util.Date object from when the last ranking of the returned list was made.

`sort`
Property on which sorting should be done. Valid values are: {averageRank,lastRanked,totalVotes}. Any of the values can be prefixed with a '+' or '-' to indicate ascending or descending sort. If no prefix is present, the default sort order is descending.

max
max items to return

7.3 qualification:list

Will create both an scripting variable and a page scoped bean of the Variable-class. Returns a Collection of qualifications matching the provided criteria.

Syntax

```
<qualification:list
   articleId="..."?
   from="..."?
   id="..."
   max="..."?
   sort="..."?
   to="..."?
   type="..."
   userId="..."?/>
```

Attributes

id, mandatory, no runtime expressions
Name of the scripting-variable.

userId
Only list qualifications matching these [UserIds].

articleId
Only list qualifications assignedto these articleIds.

type, mandatory
Indicate which type of qualification: StarRating, Flagging, Favorite

sort
Sort qualification records. Possible options: +totalAverage, -totalAverage, +qualifiedOn, -qualifiedOn

from
List qualification records assigned after this date

to
List qualification records assigned before this date

max
Maximum number of items to return.

7.4 qualification:qualification

Retrieves the Qualification for an Article. This can be a 'starrating' or 'flagging'.

Syntax

```
<qualification:qualification
   articleId="..."?
   id="..."
```

```
name="..."?
type="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the article information from. This must be an PresentationArticle or Article object.

`articleId`
The ID of the Article

`type`, mandatory
StarRating, Flagging

7.5 qualification:hasQualified

Returns true if the given article is a favorite of User. If userId is empty, the loggedin user is used.

```
<qual:hasQualified type="favorite" id="isFavorite"/>
<logic:equal name="isFavorite" value="false">
  Add to favorites
</logic:equal>
```

Syntax

```
<qualification:hasQualified
   articleId="..."?
   id="..."?
   name="..."?
   type="..."
   userId="..."?>
   ...
</qualification:hasQualified>
```

Attributes

`id`, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the article or user information from. This must be an PresentationUser, PresentationPerson, Person, PresentationArticle or Article object.

`userId`
The ID of the user

`articleId`
The id of the article to check if it is a favorite of the user. Default it get the article from the scope.

`type`, mandatory
The type of qualification you want to get. StarRating, Flagging, Favorite This is from Glenn at TV2

7.6 qualification:favoritePagerSource

Will create both an scripting variable and a page scoped bean of the Variable-class. Returns a pagerSource that you can use in the pager tag.

Syntax

```
<qualification:favoritePagerSource
   id="..."
   max="..."
   maxActionItems="..."
   name="..."?
   pageSize="..."
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the article information from. This must be an PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user you want to get.

`pageSize`, mandatory

`max`, mandatory
The maximum number of favoriteItems it returns.

`maxActionItems`, mandatory
The maximum number of pager items it creates.

7.7 qualification:mostFavoritePagerSource

Gets a pagerSource of the articles that are added most as favorite. Will create both an scripting variable and a page scoped bean of the Variable-class. Returns a pagerSource that you can use in the pager tag.

```
<qual:mostFavoritePagerSource
      id="favList"
      max="10"
      pageSize="10"
      maxActionItems="10"/>
<c:forEach items="${favList.pageItems}" var="item">
  <article:use articleId="${item.articleId}">
    ${article.title} (${item.count})
  </article:use>
</c:forEach>
```

Syntax

```
<qualification:mostFavoritePagerSource
   id="..."
   max="..."
   maxActionItems="..."
   pageSize="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`pageSize`, mandatory

`max`, mandatory
The maximum number of articles it returns.

`maxActionItems`, mandatory
The maximum number of pager items it creates.

7.8 qualification:userQualification

Will create both an scripting variable and a page scoped bean of the Variable-class. Returns the user qualification of the given User.

Syntax

```
<qualification:userQualification
   id="..."
   name="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting variable.

`name`
Name of the scripting-variable where to get the user information from. This must be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user you want to get.

7.9 qualification:userQualifications

Returns the user qualifications in the group if the groupId, groupName or name (with valid UserQualificationGroup)-parameter is used. Returns all user qualifications this user has received if userId or name (with a valid PresentationUser, PresentationPerson or Person object) is used. You can use the name, userId, groupId or groupName to get the information

Syntax

```
<qualification:userQualifications
   groupId="..."?
   groupName="..."?
```

```
id="..."
name="..."?
userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the group from. This must be a UserQualificationGroup, PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user you want to get.

`groupId`
The id of the group.

`groupName`
The name of the group.

7.10 qualification:group

Returns the user qualification group.

Syntax

```
<qualification:group
   groupId="..."?
   groupName="..."?
   id="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`groupName`
The name of the group.

`groupId`
The id of the group.

7.11 qualification:groups

Returns all the user qualification groups of the publication.

Syntax

```
<qualification:groups
   id="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

7.12 qualification:groupRecommendations

Returns all the user qualification that are recommended to the given group. You can use the name, groupId or groupName to get the group.

Syntax

```
<qualification:groupRecommendations
   groupId="..."?
   groupName="..."?
   id="..."
   name="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`name`
Name of the scripting-variable where to get the group from. This must be a UserQualificationGroup object.

`groupId`
The id of the group.

`groupName`
The name of the group.

8 stats Tag Library

8.1 stats:actionCount

Calculates the number of times a action type has occured on the given arguments. All the arguments are optional.

Syntax

```
<stats:actionCount
   article="..."?
   articleId="..."?
   author="..."?
   authorId="..."?
   from="..."?
   id="..."?
   includeArticleTypes="..."?
   section="..."?
   sectionId="..."?
   to="..."?
   type="..."
   user="..."?
   userId="..."?>
   ...
</stats:actionCount>
```

Attributes

`id`, no runtime expressions
Name of the scripting-variable.

`type`, mandatory
Action type can be: View, create, update, tag, rate, comment, search, login, message, favorite

`user`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user to get the count from.

`article`
Name of the scripting-variable where to get the user information from. This must be an PresentationArticle or Article object.

`articleId`
The id of the article to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

sectionId
The section id.

author
Name of the scripting-variable where to get the author information from. This can be a PresentationUser, PresentationPerson or Person object.

authorId
The id of the author to get the count from.

from
Date object or hours back from now

to
Date object or hours back from now

8.2 stats:actionList

Returns a list of actions based on the given arguments.

```
<stats:actionList
    id="actions"
    type="create,login,comment"
    user="userProfile"/>
<logic:iterate
    id="action"
    name="actions"
    type="com.ndc.statistics.api.domain.ActionHistory">
  <logic:equal
      name="action"
      property="actionType.name"
      value="Login">
    Logged in at <util:valueof param="action.dateTime"/>
  </logic:equal>
  <logic:equal
      name="action"
      property="actionType.name"
      value="Create">
    <logic:equal
        name="action"
        property="articleType"
        value="blog">
      Created a new article:
      <article:use articleId="<%= action.getArticleId().intValue() %>" >
        <a href="<util:valueof param="article.url"/>">go to article</a>
      </article:use>
    </logic:equal>
  </logic:equal>
  <logic:equal
      name="action"
      property="actionType.name"
      value="Comment">
    Commented in at <util:valueof param="action.dateTime"/>
  </logic:equal>
</logic:iterate>
```

Syntax

```
<stats:actionList
   article="..."?
   articleId="..."?
   author="..."?
   authorId="..."?
   from="..."?
   id="..."
   includeArticleTypes="..."?
```

```
max="..."?
section="..."?
sectionId="..."?
to="..."?
type="..."
user="..."?
userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`type`, mandatory
Action type can be:

`user`
Name of the scripting-variable where to get the user l information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id(s) of the user(s) to get the count from.

`article`
Name of the scripting-variable where to get the user information from. This must be an PresentationArticle or Article object.

`articleId`
The id of the article to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

`sectionId`
The section id.

`author`
Name of the scripting-variable where to get the author information from. This can be a **`PresentationUser`**, **`PresentationPerson`** or **`Person`** object.

`authorId`
The id of the author to get the count from.

`from`
Date object or hours back from now

`to`
Date object or hours back from now

`max`
Maximum items to be returned.

8.3 stats:actionPagerSource

Gets a list of all the actions on the given arguments. All the arguments are optional.

Syntax

```
<stats:actionPagerSource
   article="..."?
   articleId="..."?
   author="..."?
   authorId="..."?
   from="..."?
   id="..."
   includeArticleTypes="..."?
   max="..."?
   maxActionItems="..."
   pageSize="..."
   section="..."?
   sectionId="..."?
   to="..."?
   type="..."
   user="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`type`, mandatory
Action type can be: view, create, update, tag, rate, comment, search, login, message, favorite

`user`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user to get the count from.

`article`
Name of the scripting-variable where to get the user information from. This must be an PresentationArticle or Article object

`articleId`
The id of the article to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

`sectionId`
The section id.

author
Name of the scripting-variable where to get the author information from. This can be a PresentationUser, PresentationPerson or Person object.

authorId
The id of the author to get the count from.

from
Date object or hours back from now

to
Date object or hours back from now

max
Maximum items to be returned.

pageSize, mandatory

maxActionItems, mandatory

8.4 stats:userActivityList

Retrieves a list with the level of activity for every user. This depends on the type of actions a user performed because each action has a weightfactor that will be used in the calculation for the total activity. The list is order by activity so the most active users will appear first in the list.

The objects returned in the collection are UserActivity objects with a 'userId' and an 'activity' property.

```
<stats:userActivityList
    id="userActivity"
    type="create,view"
    users="members"
    max="5"/>
<logic:iterate
    id="activity"
    name="userActivity"
    type="com.ndc.statistics.api.domain.UserActivity">
  <util:valueof param="activity.userId"/>
  <util:valueof param="activity.activity"/>
  <util:valueof param="activity.weightedActivity"/>
</logic:iterate>
```

Syntax

```
<stats:userActivityList
   article="..."?
   articleId="..."?
   author="..."?
   authorId="..."?
   from="..."?
   id="..."
   includeArticleTypes="..."?
   max="..."?
   section="..."?
   sectionId="..."?
   to="..."?
   type="..."
   userIds="..."?
   users="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting-variable.

`type`, mandatory
Action type can be: view, create, update, tag, rate, comment, search, login, message, favorite

`users`
Name of the scripting-variable where to get the users collection from. It will calculates from this users. This can be a PresentationUser, PresentationPerson or Person object.

`userIds`
The id of the user to get the count from.

`article`
Name of the scripting-variable where to get the user information from. This must be an PresentationArticle or Article object.

`articleId`
The id of the article to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

`sectionId`
The section id.

`author`
Name of the scripting-variable where to get the author information from. This can be a PresentationUser, PresentationPerson or Person object.

`authorId`
The id of the author to get the count from.

`from`
Date object or hours back from now

`to`
Date object or hours back from now

`max`
Maximum items to be returned.

8.5 stats:articlePopularityList

Retrieves a list with the popularity per article. The objects returned in the collection are ItemPopularity objects with a 'itemId' and a 'popularity' property.

```
<stats:articlePopularityList
    id="articlePopularityList"
    type="view"
    includeArticleTypes="blog"
    max="10" />
<logic:iterate
    id="articlePopularity"
    name="articlePopularityList"
    type="com.ndc.statistics.api.domain.ItemPopularity">
  ArticleId <util:valueof param="articlePopularity.itemId"/>
  Popularity <util:valueof param="articlePopularity.popularity"/>
</logic:iterate>
```

Syntax

```
<stats:articlePopularityList
   author="..."?
   authorId="..."?
   from="..."?
   id="..."
   includeArticleTypes="..."?
   max="..."?
   section="..."?
   sectionId="..."?
   to="..."?
   type="..."
   user="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable

`type`, mandatory
Action type can be: view, create, update, tag, rate, comment, search, login, message, favorite

`user`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

`sectionId`
Section id.

author
Name of the scripting-variable where to get the author information from. This can be a PresentationUser, PresentationPerson or Person object.

authorId
The id of the author to get the count from.

from
Date object or hours back from now

to
Date object or hours back from now

max
Maximum items to be returned.

8.6 stats:articlePopularityPagerSource

Retrieves a pager source with the popularity per article. The objects returned in the pager are ItemPopularity objects with a 'itemId' and a 'popularity' property.

```
<stats:articlePopularityPagerSource
    id="pagerSource"
    maxActionItems="10"
    pageSize="5"
    type="view"
    includeArticleTypes="blog" />
<pager:setPage
    name="pagerSource"
    parameter="pageId"
    default="1" />

<pager:iterator
    name="pagerSource"
    id="articlePopularity"
    type="com.ndc.statistics.api.domain.ItemPopularity">
  ${articlePopularity.itemId}
  ${articlePopularity.popularity}
</pager:iterator>

<pager:actions id="action" name="pagerSource">
  <logic:equal
      name="action"
      property="active"
      value="false">
    <span>${action.label}</span>
  </logic:equal>
  <logic:equal
      name="action"
      property="active"
      value="true">
    <a href="${section.value}?pageId=${action.value}">
      ${action.label}
    </a>
  </logic:equal>
</pager:actions>
```

Syntax

```
<stats:articlePopularityPagerSource
   author="..."?
   authorId="..."?
   from="..."?
   id="..."
```

```
includeArticleTypes="..."?
max="..."?
maxActionItems="..."
pageSize="..."
section="..."?
sectionId="..."?
to="..."?
type="..."
user="..."?
userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`type`, mandatory
Action type can be: view, create, update, tag, rate, comment, search, login, message, favorite

`user`
Name of the scripting-variable where to get the user information from. This can be a PresentationUser, PresentationPerson or Person object.

`userId`
The id of the user to get the count from.

`includeArticleTypes`
The type of the article to get the count from.

`section`
Name of the scripting-variable where to get the user information from. This must be an PresentationSection or Section object.

`sectionId`

`author`
Name of the scripting-variable where to get the author information from. This can be a PresentationUser, PresentationPerson or Person object.

`authorId`
The id of the author to get the count from.

`from`
Date object or hours back from now

`to`
Date object or hours back from now

`max`
The maximum number of items to return.

`pageSize`, mandatory

`maxActionItems`, mandatory

8.7 stats:tagSuggest

Retrieves a list of suggested tag for the given tag id. It will get his suggestions from the create tag and search tag recorders.

```
<stats:tagSuggest id="tagSuggestions" tagId="<%=myTag%>"/>
<logic:iterate id="tagSuggestion" name="tagSuggestions">
  <bean:write name="tagSuggestion" property="tagId"/>
</logic:iterate>
```

Syntax

```
<stats:tagSuggest
   id="..."
   max="..."?
   tagId="..."/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`tagId`, mandatory

`max`
The maximum number of items to return.

8.8 stats:tagPopularityList

Retrieves a list with the most popular tags. The objects returned in the collection are ItemPopularity objects with a 'itemId' and a 'popularity' property.

```
<stats:tagPopularityList id="tagPopularities" max="20"/>
<logic:iterate id="tagPopularity" name="tagPopularities">
  <bean:write name="tagPopularity" property="itemId"/>
  <bean:write name="tagPopularity" property="popularity"/>
</logic:iterate>
```

Syntax

```
<stats:tagPopularityList
   from="..."?
   id="..."
   max="..."?
   to="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting-variable.

`from`
Date object or hours back from now

`to`
Date object or hours back from now

max
The maximum number of items to return.

8.9 stats:isEnabled

Checks if the statistics are enabled.

Syntax

```
<stats:isEnabled
   id="..."?>
   ...
</stats:isEnabled>
```

Attributes

id, no runtime expressions
Name of the scripting-variable.

9 tag Tag Library

9.1 tag:tags

Syntax

```
<tag:tags
   articleId="..."?
   id="..."
   name="..."?
   returnIds="..."?
   type="..."?
   userId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions

`name`
PresentationArticle, Article, PresentationUser

`articleId`

`userId`

`type`
theme, custom, tree

`returnIds`

9.2 tag:tag

Gets the tag.

Syntax

```
<tag:tag
   id="..."
   tagId="..."?
   tagName="..."?
   toScope="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
Name of the scripting variable.

`tagId`

`tagName`

toScope
The scope where the variable will be saved in. Default "page". Options: "page", "request", "session" and "application".

9.3 tag:cloud

Returns a collection of tags with metainformation to create a tag cloud.

```
<tag:cloud
    id="tags"
    max="10"
    sizeMax="20"
    sizeMin="10"/>
<logic:iterate
    id="tag"
    name="tags"
    type="com.ndc.tag.plugin.util.TagMetaInformation">
  <span style="font-size:<util:valueof param="tag.tagSize"/>px">
    <util:valueof param="tag.name"/>
  </span>
</logic:iterate>
```

Syntax

```
<tag:cloud
   id="..."
   includeArticleTypes="..."?
   max="..."
   sizeMax="..."
   sizeMin="..."
   tagId="..."?/>
```

Attributes

id, mandatory, no runtime expressions
The name of the scripting variable.

max, mandatory
The maximum number of tags to be included in the cloud.

sizeMax, mandatory
The max size of the tags. You can use it for font size.

sizeMin, mandatory
The minimum size of the tags. You can use it for font size.

includeArticleTypes
The article type you want to include. May be a list of comma separated types.

tagId
The tag IDs you want to create a cloud from. May be a list of comma separated ids.

9.4 tag:children

Returns the children of the given tag. Or the root tag from a tree.

Syntax

```
<tag:children
   id="..."
   name="..."?
   tagId="..."?
   tagName="..."?
   treeId="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting variable.

`name`
The name of the tag to get the children from.

`tagId`
The id of the tag to get the children from.

`tagName`
The name of the tag to get the children from.

`treeId`
The treeId of the tree where to get the root tags from.

9.5 tag:articles

Returns a collection of articles related to the given tag. Please note that this tag is performance wise very expensive and should not be used without some kind of caching around it. Either by using **`<util:cache/>`** or setting the HTTP header **`Cache-Control: s-maxage`**

```
  <tag:articles
      id="taggedArticles"
      includeArticleTypes="blog"
      tagName="<%= tagName %>"/>

  <logic:iterate
      id="taggedArticle"
      name="taggedArticles"
      type="neo.xredsys.presentation.PresentationArticle">
    <util:valueof param="taggedArticle.url"/>
  </logic:iterate>
```

Syntax

```
<tag:articles
   authorId="..."?
   id="..."
   includeArticleTypes="..."?
   max="..."?
   offset="..."?
   operator="..."?
   sectionId="..."?
   sort="..."?
   tagId="..."?
   tagName="..."?/>
```

Attributes

`id`, mandatory, no runtime expressions
The name of the scripting variable.

`tagName`
The name of the tag to get the articles from. May be a list of comma separated names.

`tagId`
The ID of the tag to get the articles from. May be a list of comma separated IDs.

`authorId`
The ID of the author to get the articles from.

`sectionId`
The section id of the articles. May be a list of comma separated IDs.

`includeArticleTypes`
ArticleTypes. Collection of Strings or comma separated.

`max`
How many articles to list. Default is 50.

`offset`
The start point for a range of results for the query.

`sort`
The list of articles returned can be sorted ascending or descending on the date the tag was set on the article. Valid values are: date (default) or -date.

`operator`
This attribute specified which logic to apply while searching with tag IDs. Valid values are: **`or`** (default), **`and`**.

9.6 tag:trees

Give a collection of tags. If no id is set it returns HTML code with the tag trees.

If id is set it will create both an scripting variable and a page scoped bean of the Variable-class. If no id is set it returns HTML code with the tag trees.

Syntax

```
<tag:trees
   id="..."?
   treeId="..."?
   treeName="..."?
   type="..."?/>
```

Attributes

`id`, no runtime expressions
Name of the scripting variable.

treeId
The id of the tree to retrieve.

treeName

type
The type of the tagTree to retrieve. Valid values are: theme and all (default).

9.7 tag:themeTag

Will create both an scripting variable and a page scoped bean of the Variable-class. You must use one of the following attributes: name or articleID OR tagId/ tagName.

If article is given it returns the last or first child tag in the tree related to article. for example. Type is last and an article is tagged with: "dog > big dogs > Greyhound". It returns the "Greyhound" tag. Type is first and an article is tagged with: "dog > big dogs > Greyhound". It returns the "dog" tag.

If tag is given it returns the FIRST parent tag in the tree. if we take the example above. and we give the "Greyhound" tagName it returns the "Dog" tag.

Syntax

```
<tag:themeTag
   articleId="..."?
   id="..."
   name="..."?
   tagId="..."?
   tagName="..."?
   toScope="..."?
   type="..."?/>
```

Attributes

id, mandatory, no runtime expressions
Name of the scripting variable.

name
The name of the bean to specify the article to get the theme tag from. This can be a Tag, PresentationArticle or Article object.

tagId
The id of the tag to get the theme tag from.

tagName
The name of the tag to get the theme tag from.

articleId
The id of the article to get the theme tag from.

type
First or last theme tag. Options: "first"(default), "last"

toScope
The scope where the variable will be saved in. Default "page". Options: "page", "request", "session" and "application".

9.8 tag:articleCount

Gives number of articles tagged with the specified tag. One of the following attributes must be used : tagId/tagName.

Syntax

```
<tag:articleCount
   id="..."
   includeArticleTypes="..."?
   tagId="..."?
   tagName="..."?/>
```

Attributes

id, mandatory, no runtime expressions
Name of the scripting variable.

tagId
The ID of the tag.

tagName
The name of the tag.

includeArticleTypes
The article type(s) you want to include. The value can be a list of comma separated types.