

Creating themes for LibreCCM with Freemarker

Starting with version 2.5 the LibreCCM platform support Freemarker as an alternative to XSL. Freemarker is a project of the Apache Foundation and a well known and mature template engine for Java. The support for Freemarker in version 2.5 is a backport from the upcoming version 7 of the LibreCCM platform.

Compared to XSL Freemarker is a lot easier to use, especially if you have worked with other template engines like Twig, Velocity etc before. In version 7 of the LibreCCM platform Freemarker will become the primary template engine. XSL will still be supported, but we recommended that you port your themes to Freemarker. Why Freemarker and not one of the other template engines? Freemarker is able to process XML in a similar way than XSL.

Freemarker also allows it to define user defined directives and functions. To make it easier to create impressive themes we provide functions and macros for Freemarker we provide several functions and macros which hide the complexity of the XML data model created by CCM from the template author. It is recommended not to access the XML data model directly. Instead the provided functions should be used. Otherwise your theme might brake when the XML structure changes.

General structure of a Freemarker theme

Freemarker themes have a different structure than the usual “old style” themes of LibreCCM.

ToDo

Predefined variables and functions

Several variables and functions are predefined and available without importing another file.

Variables

contextPath

The context path in which CCM is running.

contextPrefix

The context prefix.

dispatcherPrefix

Prefix for the CCM dispatcher (usually /ccm)

host

The current host.

model

The XML document created by LibreCCM.

negotiatedLanguage

The language negotiated between the user agent and LibreCCM.

requestScheme

The protocol (http or https).

selectedLanguage

The language selected by the user.

themePrefix

The prefix of the theme. Only available if a development theme is viewed.

Functions**getLocalizedText**

String getLocalizedText(String key)

Returns the localized text from the resource bundle of the theme.

getContentItemTemplate

String getContentItemTemplate(String objectType, view="DETAIL", style="")

This is an internal function!

Returns the path for the template of a content item of a specific type.

`_formatDateTime`

An internal functions date time formatting. This functions should not be used directly.

Functions and Macros

Freemarker functions for ccm-cms

Functions for all content types

Import path `/ccm-cms/content-item.ftl`

This functions and macros work for all content item types. Unless stated otherwise all functions support the detail view as well as the list view generated by the `ObjectList` components of the ccm-navigation module.

Node `getContentItem()`

Retrieves the content item of the current page. This works only on pages generated by the ContentSection application. To get the index item of a navigation page use the functions provided by the ccm-navigation module.

This function is primarily used to as parameter for other functions dealing with content items.

`@contentItem item: Node view: String style: String`

This macro generates the detail view of a content item. Using the type of the provided content items, the provided type and the provided type this macro internally tries to find a view definition in for the provided parameters in the theme manifest. The template defined in this definition is used to generate the HTML representation of the item.

The parameters `view` and `style` are optional. The `view` parameter defaults to `detail`, the default value for the `style` parameter is an empty string.

`String getItemTitle(item: Node)`

This function retrieves the title of the provided content items.

`String getPageTitle(useCategoryMenu: String, useRootIndexItemTitle: boolean)`

Retrieves the title of the current page. The title retrieved depends on the provided parameters. Both parameters are optional. `useCategoryMenu` is used to select the category menu which is used to retrieve the page title. The default

value is `categoryMenu`. The `useRootIndexItemTitle` defaults to `false` and is used to determine if the title of the category or the title of the index item is used as page title for the index page of an category.

String getItemSummary(item: Node)

Retrieves the summary/lead text of the provided content item if the item has a property. The following property names are tried: `lead`, `summary`. If none is found an empty string is returned.

String getPageDescription(item: Node)

Retrieves the value of the `pageDescription` property of the provided content item if the item has such a property. If not an empty string is returned.

String generateContentItemLink(oid: String)

Generates a link to the content item with the provided OID.

String getEditLink(item: Node)

This function generates a link for editing the provided content item if the link is available. If the link is not available, for example if the current user is not permitted to edit the item, an empty string is returned.

Functions of types derivated from `OrganizationalUnit`

Import Path `/ccm-cms/orgaunit.ftl`

Sequence getAvailableTabs(item: Node)

Returns a sequence of the information tabs available for the provided organizational unit item. For processing the return values of this functions the other functions provided by this file should be used.

String getTypeNameKey(item: Node)

Retrieves the type of the provided orga unit item.

String getTabLabel(tab: Node)

Retrieves the label of the provided tab.

boolean isTabSelected(tab: Node)

Checks if the provided tab is the currently selected tab.

String getTabLink(tab: Node)

Retrieves the link for the provided tab.

Node getSelectedTab(item: Node)

Retrieves the data of the selected tab.

String getTypeOfSelectedTab(item: Node)

Returns the type of the selected tab.

Node getPropertyFromTab(tab: Node)

Returns a property from the provided tab.

Mixed getTabContent(tab: Node)

Returns the content of the provided tab.

String getAddendum(data: Node)

Retrieves the value of the addendum property from the provided tab data (as returned by `getTabContent`) if the data contains such a property.

Sequence getMembers(data: Node)

Retrieves the members of the orga unit from the provided tab data (as returned by `getTabContent`) if the data contains such a property.

String getMemberRole(member: Node)

Retrieves the role of the provided member.

String getMemberStatus(member: Node)

Retrieves the status of the provided member.

Sequence<Node> getContactEntries(member: Node)

Retrieves the contact entries of the provided member. For processing the members the functions provided by `person.ftl` can be used.

boolean hasOrgaUnitContact(data: Node)

Returns `true` if the provided data contains a contact item.

Node getOrgaUnitContact(data: Node)

Retrieves the contact item for the orga unit from the provided data.

Node getOrgaUnitContactPerson(data: Node)

Retrieves the person associated with the provided contact.

Node getOrgaUnitContactEntries(data: Node)

Retrieves the contact entries of the provided contact item. For further processing the functions provided by the `ccm-cms-types-contact` module should be used.

Functions for person items

Import path `/ccm-cms/person.ftl`

This functions can be used to process content items derivated from the `GenericPerson` type.

String getSurname(item: Node)

Gets the surname from the provided person item.

String getGivenName(item: Node)

Gets the given name from the provided person item.

String getTitlePre(item: Node)

Gets the value of the `titlePre` property of the provided person item.

String getTitlePost(item: Node)

Gets the value of the `titlePost` property of the provided person item.

```
String getHomepageLink(item: Node, contactType: string="commonContact",
entry: String="homepage")
```

Retrieves the link to the homepage of the provided person item if the item has an contact entry for a homepage. The optional parameters `contactType` and `entry` can be used to select the contact and the entry from which the value is read. The default value for `contactType` is `commonContact`. For `entry` the default value is `homepage`.

```
String getAddress(item: Node, contactType: String="commonContact")
```

Retrieves the address item associated with contact of the provided person. The contact to use can be selected using the optional parameters `contactType`. The default value is `commonContact`. # Functions for File Attachments

Import Path /ccm-cms-assets-fileattachments.ftl

This module provides functions for dealing with file attachments. A possible usage these functions:

```
<#list FileAttachments.getFileAttachments(item)>
  <div class="file-attachments">

    <h2>
      ${getLocalizedText("layout.page.main.fileAttachments")}
    </h2>

    <ul class="file-attachments">
      <#items as file>
        <#if FileAttachments.getFileType(file) == "caption">
          <li class="caption">
            <strong>${FileAttachments.getFileName(file)}</strong>
            <p>
              ${FileAttachments.getFileDescription(file)}
            </p>
          </li>
        <#else>
          <li class="file-attachment">
            <a href="${FileAttachments.getFileUrl(file)}">
              <span class="fa fa-download"></span>
              ${FileAttachments.getFileDescription(file)}
              (${FileAttachments.getMimeTypeFileExtension(file)},
              ${FileAttachments.getFileSize(file, "KiB")} KB)
            </a>
          </li>
        </#if>
      </#items>
    </ul>
  </div>
</list>
```

```

        </#if>
      </#items>
    </ul>
  </div>
</#list>

```

getFileAttachments(item: Node): Sequence<Node>

Retrieves the file attachments of the provided content item.

getFileType(file: Node): String

Returns the type of the file attachments which is either `caption` or `file`.

getMimeType(file: Node): String

Returns the mime type of the file, for example `image/png` or `application/pdf`.

getMimeTypeFileExtension(file: Node): String

Returns the usual file extension for the mime type of the file.

getFileSize(file: Node, unit: String = "byte"): Number

Returns the size of the provided file. The unit in which the size of the file is returned can be changed by using the optional parameter `unit`. The default value for the unit is `byte`.

getFileId(file: Node): String

Returns the ID of the file.

getFileName(file: Node): String

Returns the name of file.

getFileDescription(file: Node): String

Returns the description of the file.

getFileUrl(file: Node): String

Returns the URL of the file.

Freemarker functions for Image Attachments

Provides functions for dealing with image attachments of a content item.

Import path /ccm-cms-assets-imagestep.ftl"

Example usage:

```
<#import "/ccm-cms-assets-imagestep.ftl" as Images>

<#list Images.getImageAttachments(item)>
  <div class="image-attachments">
    <#items as image>
      <figure>
        <div>
          <a data-fancybox="gallery">
            
          </a>
        </div>
        <figcaption>
          ${Images.getImageCaption(image)}
        </figcaption>
      </figure>
    </#items>
  </div>
</#list>
```

getImageAttachments(item: Node): Sequence<Node>

Get the image attachments of the provided content item.

getImageId(image: Node): String

Gets the ID of the provided image.

getImageName(image: Node): String

Gets the name of the provided image.

getImageCaption(image: Node): String

Gets the caption of the provided image.

getImageSortKey(image: Node): String

Gets the sort key of the provided image.

getImageWidth(image: Node): String

Gets the width of the provided image.

getImageHeight(image: Node): String

Gets the height of the provided image.

getImageUrl(image: Node): String

Gets the URL of the provided image.

Freemarker functions for Sidenote assets

Functions for processing note assets assigned to a content item.

Import path /ccm-cms-assets-notes.ftl

getNotes(item: Node): Sequence<Node>

Returns the notes assigned to a content item.

getContent(item: Node): String

Gets the content of a note. The return value is the HTML content of the node.

Freemarker functions for related links

Import path /ccm-cms-assets-relatedlinks

Functions for processing the related links assigned to a content item.

**getRelatedLinks(item: Node, linkListName: String = "NONE"):
Sequence<Node>**

Retrieves the related links assigned to a content item. Related links can be organized in named list. The optional parameters `linkListName` controls which list is used. If the parameter is omitted the default value `NONE` is used.

getLinkType(link: Node): String

Gets the type of the provided link which can either be `externalLink`, `internalLink` or `caption`.

getLinkTitle(link: Node): String

Gets the title of the provided link.

getLinkDescription(link: Node): String

Gets the description of the provided link.

getLinkOrder(link: Node): String

Gets the order value for the provided link.

getInternalLinkParameters(link: Node): String

Gets the URL parameters of the of the provided link (The part after the question mark).

getTargetUri(link: Node): String

Gets the URI of the target of the provided link.

Freemarker functions for Public Personal Profiles

Import path /ccm-cms-publicpersonalprofile.md

Functions for processing the data of a public personal profile.

getProfileOwner(data: Node): Node

Get the data about the profile owner. The return value is a XML node which can be further processed with other functions.

getProfileOwnerSurname(owner: Node): String

Gets the surname of a profile owner.

getProfileOwnerGivenName(owner: Node): String

Gets the given name of a profile owner.

getProfileOwnerTitlePre(owner: Node): String

Gets the titles a profile owner.

getProfileOwnerTitlePost(owner: Node): String

Gets the titles a profile owner.

getProfilePosition(data: Node): String

Returns the value of the `position` property of a profile.

getProfileInterests(data: Node): String

Returns the value of the `interests` property of a profile.

getProfileMisc(data: Node): String

Returns the value of the `misc` property of a profile.

getProfileOwnerContact(owner: Node): Node

Gets the contact data of the owner. The contact data is in the same format as a content item of the type `ccm-cms-types-contact`. The returned data can be processed further using the functions for content items of the type `ccm-cms-types-contact?`. The functions provided by the `ccm-cms-types-contact` module can be used to process this data.

getProfileImage(data: Node): String

Returns the data of the image attached to the profile, if any. The returned data is a image attachment which can be processed further by the functions provided for processing image assets (see `ccm-cms-assets-imagestep`).

getProfileOwnerName(data: Node): String

Gets the name of the profile owner which is the name of the content item of the type `Person` assigned to the profile.

getPersonalPublications(data: Node): Sequence<Node>

Gets the data about the personal publications of the profile owner, organized in publications groups.

getPersonalPublicationsAvailablePublicationGroups(data: Node): Sequence<Node>

Get the available publications groups. The items of the sequence can be processed further using `getPublicationGroupId` and `getPublicationGroupLink`.

getPublicationGroupId(group: Node): String

Returns the ID of the publication group.

getPublicationGroupLink(group: Node): String

Returns the link for showing the publications of the group.

getPublicationGroups(data: Node): Sequence<Node>

Get all publication groups currently displayed.

getPublicationsOfGroup(data: Node): Sequence<Node>

Gets the publications of a group. The publication can be processed further by the functions provided by `ccm-sci-publications`.

hasPublicationsPaginator(profile: Node): boolean

Determines if there is paginator for the current publication group.

getPublicationsPaginatorBaseUrl(profile: Node): String

Returns the base URL for the publications paginator.

getPublicationsPaginatorPageCount(profile: Node): String

Returns the number of pages from the publications paginattor.

getPublicationsPaginatorPageNumber(profile: Node): String

Returns the current page of the current publication group.

getPublicationsPaginatorPageParam(profile: Node): String

Gets the name of the URL parameter for changing the current page.

getPublicationsPaginatorPageSize(profile: Node): String

Gets the page size.

getPublicationsPaginatorObjectBegin(profile: Node): String

Gets the index of the first displayed item of current publication group.

getPublicationsPaginatorObjectCount(profile: Node): String

Gets the index of the number of items in the current publication group.

getPublicationsPaginatorObjectEnd(profile: Node): String

Gets the index of the last displayed item of current publication group.

getPublicationsPaginatorPrevPageLink(profile: Node): String

Gets the link to the previous page of the current publication group.

getPublicationsPaginatorFirstPageLink(profile: Node): String

Gets the link the first page of the current publication group.

getPublicationsPaginatorNextPageLink(profile: Node): String

Gets the link to the next page of the current publication group.

getPublicationsPaginatorLastPageLink(profile: Node): String

Gets the link to the last page of the current publication group.

getAvailableProjectGroups(data: Node): Sequence<Node>

Returns the available project groups. The sequence can be processed further using `getProjectGroupId` and `getProjectGroupLink`.

getProjectGroupId(group: Node): String

Returns the ID of the Project group.

getProjectGroupLink(group: Node): String

Returns the link for showing the Projects of the group.

getProjectGroups(data: Node): String

Gets all project groups currently displayed.

getProjectsOfGroup(data: Node): Sequence<Node>

Gets the projects of a group. The projects can be processed further by the functions provided by `ccm-sci-types-project`.

hasProjectsPaginator(profile: Node): boolean

Determines if the current project group has a paginator.

getProjectsPaginatorBaseUrl(profile: Node): String

Returns the base URL for the projects paginator.

getProjectsPaginatorPageCount(profile: Node): String

Returns the number of pages from the project paginator.

getProjectsPaginatorPageNumber(profile: Node): String

Returns the current page of the current project group.

getProjectsPaginatorPageParam(profile: Node): String

Gets the name of the URL parameter for changing the current page.

getProjectsPaginatorPageSize(profile: Node): String

Gets the page size.

getProjectsPaginatorObjectBegin(profile: Node): String

Gets the index of the first displayed item of current project group.

getProjectsPaginatorObjectCount(profile: Node): String

Gets the number of items in the current project group.

getProjectsPaginatorObjectEnd(profile: Node): String

Gets the index of the last displayed item of current project group.

getProjectsPaginatorPrevPageLink(profile: Node): String

Gets the link to the previous page of the current project group.

getProjectsPaginatonFirstPageLink(profile: Node): String

Gets the link the first page of the current project group.

getProjectsPaginatorNextPageLink(profile: Node): String

Gets the link to the next page of the current project group.

getProjectsPaginatorLastPageLink(profile: Node): String

Gets the link to the last page of the current project group. # Freemarker functions for ccm-cms-types-address

Import Path /ccm-cms-type-address.ftl

getAddressText(item: Node): String

Returns the value of the **text** property of the address.

getCity(item: Node): String

Returns the value of the **city** property of the address.

getPostalCode(item: Node): String

Gets the postal code of the address.

getState(item: Node): String

Gets the value of the **state** property of the address. (state means the a federal state or the equivalent here, for example California in the USA oder Lower Saxony in Germany)

getCountry(item: Node): String

The country of the address.

getIsoCountryCode(item: Node): String

Gets the ISO country code for the country of the address. # Freemarker functions for Article content items

getLead(item: Node): String

Retrieves the lead text of the provided article.

getMainText(item: Node): String

Retrieves the main text of the provided item. The return value is a HTML string. # Freemarker functions for Contact items

Import path /ccm-cms-types-contact.ftl

getAddress(item: Node): Node

Returns the address associated with the provided contact item. The address can be processed further using the functions provided by the ccm-cms-types-address module.

getPerson(item: Node): Node

Returns the person associated with the provided contact. The returned person item can be processed further using functions provided by the ccm-cms module.

getContactEntries(item: Node): Sequence<Node>

Returns the contact entries of the provided contact.

getContactEntry(item: Node, keyId: String): Node

Returns the contact entry with the provided **keyId** if the provided contact has a matching contact entry. If not **null** is returned.

getContactEntryLabel(entry: Node): String

Returns the label of the provided contact entry.

getContactEntryValue(entry: Node): String

Returns the value of the provided contact entry. # Freemarker functions for Article content items

getLead(item: Node): String

Retrieves the lead text of the provided article.

getMainText(item: Node): String

Retrieves the main text of the provided item. The return value is a HTML string. # Freemarker functions for Bookmark items

getDescription(item: Node): String

Gets the description of the bookmark.

getLink(item: Node): String

Gets the link for the bookmark's target.

Freemarker functions for Contact items

Import path /ccm-cms-types-contact.ftl

getAddress(item: Node): Node

Returns the address associated with the provided contact item. The address can be processed further using the functions provided by the ccm-cms-types-address module.

getPerson(item: Node): Node

Returns the person associated with the provided contact. The returned person item can be processed further using functions provided by the ccm-cms module.

getContactEntries(item: Node): Sequence<Node>

Returns the contact entries of the provided contact.

getContactEntry(item: Node, keyId: String): Node

Returns the contact entry with the provided keyId if the provided contact has a matching contact entry. If not null is returned.

getContactEntryLabel(entry: Node): String

Returns the label of the provided contact entry.

getContactEntryValue(entry: Node): String

Returns the value of the provided contact entry. # Freemarker functions for Event items

Import path /ccm-cms-types-event.ftl

getLead(item: Node): String

Returns the lead text of the event.

getMainText(item: Node): String

Returns the main text of the event.

getEndDate(item: Node): DateTimeNode

Returns the end date of the provided event item. To format the date the `formatDateTime` function provided by the ccm-cms module should be used.

getEndDateYear(item: Node): String

Returns the year part of the end date of the event.

getEndDateMonth(item: Node): String

Returns the month part of the end date of the event.

getEndDateShortMonth(item: Node): String

Returns the the short name of month part of the end date of the event.

getEndTime(item: Node): String

Gets the end time of the event.

getEndTimeHour(item: Node): String

Gets the hour part of the end time of the event.

getEndTimeMinute(item: Node): String

Gets the minute part of the end time of the event.

getEndTimeSecond(item: Node): String

Gets the second part of the end time of the event.

getStartDate(item: Node): DateTimeNode

Returns the start date of the provided event item. To format the date the `formatDateTime` function provided by the ccm-cms module should be used.

getStartDateYear(item: Node): String

Returns the year part of the start date of the event.

getStartDateMonth(item: Node): String

Returns the month part of the start date of the event.

getStartDateShortMonth(item: Node): String

Returns the the short name of month part of the start date of the event.

getStartTime(item: Node): String

Gets the start time of the event.

getStartTimeHour(item: Node): String

Gets the hour part of the start time of the event.

getStartTimeMinute(item: Node): String

Gets the minute part of the start time of the event.

getStartTimeSecond(item: Node): String

Gets the second part of the start time of the event.

getLocation(item: Node): String

Gets the location of the event.

getMainContributor(item: Node): String

Gets the value of the mainContributor property of the event.

getEventType(item: Node): String

Returns the value of the eventType property of the event.

getCost(item: Node): String

Returns the value of the cost property of the event.

getMapLink(item: Node): String

Returns the value of the mapLink property of the event.

getEventDateAddendum(item: Node): String

Returns the value of the addendum property of the event. # Freemarker functions for ExternalLink items

Import path /ccm-cms-types-externallink.ftl

getDescription(item: Node): String

Gets the description of the external link item.

getComment(item: Node): String

Gets the value of the comment property of the link item.

isTargetNewWindow(item: Node): String

Returns true if the link should be opened in a new window/tab.

getUrl(item: Node): String

Returns the URL of the external link. # Freemarker functions for File Storage Items

Import path /ccm-cms-types-filestorageitem.ftl

getDescription(item: Node): String

Gets the description of the file storage item.

getFileId(item: Node): String

Returns the ID for the file represented by the file storage item.

getFileName(item: Node): String

Returns the name of the file represented by the file storage item.

getFileLink(item: Node, mode: String="download", useFileName: boolean = true): String

Returns the link for downloading or viewing the file. The optional parameter mode controls if the link for downloading or for viewing the file is generated. The supported values are download (default value) and stream. Unknown values are interpreted as download.

The optional `useFileName` parameter controls if the name of the file (see `getFileName`) is included in the link. The default value is `true`. # Freemarker functions for Form items

Import Path `/ccm-cms-types-formitem.md`

getDescription(item: Node): String

Gets the description of the form item.

getFormAction(item: Node): String

Gets the URL to which the form is send.

hasHoneypot(item: Node): boolean

Returns `true` if the form contains a honeypot field for catching bots.

getHoneypotName(item: Node): String

Gets the name of the honeypot field.

hasMinTimeCheck(item: Node): boolean

Returns `true` if a check of the time the user needs to fill out the form should be added to the form. Bots are normally extremly fast (faster than any human).

getMinTimeCheckValue(item: Node): String

The minium time for the min time check.

hasVisitedField(item: Node): boolean

Returns `true` if a visited field is part of the form.

getVisitedField(item: Node): String

Gets the value of the visited field.

getVisitedFieldName(item: Node): String

Returns the name of the visited field.

getPageStateFieldName(item: Node): String

Gets the name to use for the (hidden) field holding the page state.

getPageStateFieldValue(item: Node): String

Gets the value of the (hidden) field holding the page state.

getComponents(item: Node): Sequence<Node>

Returns the components of the form.

getComponentName(component: Node): String

Returns the name of the provided component.

getComponentDefaultValue(component: Node): String

Returns the default value of the provided component.

getComponentDescription(component: Node): String

Returns the description of the provided component.

getComponentParameterName(component: Node): String

Returns the parameter name of the provided component.

getComponentType(component: Node): String

Returns the type of the provided component.

getFormSectionTitle(formSection: Node): String

Gets the title of the provided form section.

getFormSectionComponents(formSection: Node): String

Gets the components of the form section.

getLabelComponents(label: Node): Sequence<Node>

Gets the components of a label.

getButtonGroupComponents(buttonGroup: Node): Sequence<Node>

Gets the components of a button group.

isMultipleSelect(select: Node): boolean

Returns **true** if the provided select component allows multiple values.

isDataDrivenSelect(select: Node): boolean

Returns **true** if the provided select component is a data driven select.

getDataOptions(select: Node): Sequence<Node>

Gets the options of a data driven select.

getDataOptionLabel(option: Node): String

Get the label of an option of an data driven select.

getDataOptionId(option: Node): String

Get the id of an option of an data driven select.

getDataOptionComponents(option: Node): Sequence<Node>

Get the components of an option of an data driven select.

hasOtherOption(select: Node): boolean

Returns **true** if the provided select component has an **other** option.

getOtherOptionLabel(select: Node): String

Returns the label for the **other** option.

getOtherOptionValue(select: Node): String

Returns the value for the **other** option.

hasMaxLength(component: Node): boolean

Returns **true** if the provided component has a **maxLength** property.

getMaxLength(component: Node): String

Gets the value of the `maxLength` property of the provided component.

hasSize(component: Node): boolean

Returns `true` if the provided component has a `size` property.

getMaxSize(component: Node): String

Gets the value of the `size` property of the provided component.

getRequired(component: Node): String

Determines if the provided component represents a mandatory field of the form.

getDateFieldDayParamName(component: Node): String

Returns the name of the day param of a date input component.

getDateFieldMonthParamName(component: Node): String

Returns the name of the month param of a date input component.

getDateFieldYearParamName(component: Node): String

Returns the name of the year param of a date input component.

getDateFieldDefaultValueDay(component: Node): String

Returns the default value for the day of the provided date input component.

getDateFieldDefaultValueMonth(component: Node): String

Returns the default value for the month of the provided date input component.

getDateFieldDefaultValueYear(component: Node): String

Returns the default value for the year of the provided date input component.

getDateFieldMonthList(component: Node): Sequence<Node>

Gets the list of permitted months.

getDateFieldYearList(component: Node): Sequence<Node>

Gets the list of permitted years.

getMonthLabel(month: Node): String

Gets the label for the provided month.

getYearLabel(year: Node): String

Gets the label for the provided year.

getTextAreaRows(textArea: Node): String

Gets the number of rows for the provided text area component.

getTextAreaCols(textArea: Node): String

Gets the number of cols for the provided text area component. # Freemarker functions for Image items

Import Path /ccm-cms-types-image.ftl

getArtist(item: Node): String

Returns the name of the artist how created the image.

getCopyright(item: Node): String

Returns the value of the `copyright` property of the image.

getDescription(item: Node): String

Returns the description of the image.

getLicense(item: Node): String

Returns the license text for the image.

getMaterial(item: Node): String

Gets the value of the `material` property of the image.

getPublishDate(item: Node): String

Gets the publish date of the image.

getWidth(item: Node): String

Gets the value of the width property of the image.

getHeight(item: Node): String

Gets the value of the height property of the image.

getMainText(item: Node): HTMLString

Gets the main text describing the image.

getOrigin(item: Node): String

Gets the value of the origin property of the image.

getOriginalSize(item: Node): String

Gets the original size of the image.

getUrl(item: Node): String

Returns the URL of the image resource.

getCaption(item: Node): String

Returns the caption of the image.

getImageId(item: Node): String

Gets the ID of the image resource.

getThumbnailId(item: Node): String

Gets the ID of the thumbnail image.

getThumbnailWidth(item: Node): String

Gets the width of the thumbnail.

getThumbnailHeight(item: Node): String

Gets the height of the thumbnail.

getSite(item: Node): String

Gets the value of the `site` property of the image.

getSource(item: Node): String

Gets the value of the `source` property of the image.

getTechnique(item: Node): String

Gets the value of the `technique` property of the image. # Freemarker functions for MultiPartArticles

Import Path /ccm-cms-types-multiparticle.ftl

getSummary(item: Node): String

Returns the summary of the provided MultiPartArticle item.

getSections(item: Node): Sequence<Node>

Returns the currently displayed sections of the multipart article. The sections can be processed further by some of the other funtions provided by this library.

getSectionTitle(section: Node): String

Returns title title of the provided section.

getSectionTitle(section: Node): HtmlString

Gets the content of the provided section.

getSectionRank(section: Node): String

Gets the value of the `rank` property of the provided section.

getPageNumber(item: Node): String

Gets the number of the page of the multipart article displayed.

getNumberOfPages(item: Node): String

Returns the number of pages of the provided multipart article.

hasPreviousPage(item: Node): boolean

Returns **true** if the provided multipart article has previous pages.

hasPreviousPage(item: Node): boolean

Returns **true** if the provided multipart article has more pages.

hasMultiplePages(item: Node): boolean

Returns **true** if the provided multipart article has more than one page.

getLinkToPreviousPage(item: Node): String

Returns the link to the previous page if any.

getLinkToNextPage(item: Node): String

Returns the link to the next page if any.

getAllSectionsLink(item: Node): String

Returns the link for showing the complete multipart article on a single page. #
Freemarker functions for generating the table of contents of a MultiPartArticle

Import Path /ccm-cms-types-multiparticle-toc.ftl

These functions can be used to generate the table of contents (toc) for a multi
part article. An example:

```
... // Other imports
<#import "/ccm-cms-types-multiparticle-toc.ftl" as Toc>

... // Other things

<#list Toc.getSections(item)>
  <div class="mpa-toc">
    <h3>${getLocalizedText("mpa.toc")}</h3>
    <ul class="mpa-toc">
      <#items as section>
        <li>
          <a href="${Toc.getSectionLink(section)}"
            class="${Toc.isActiveSection(item, section)?then('active', '')}">
```

```

                ${Toc.getSectionTitle(section)}
            </a>
        </li>
    </#items>
</ul>
</div>
</#list>

```

getSections(item: Node): Sequence<Node>

Returns the sections of the provided MultiPartArticle item. The sections can be further be processed using the other functions provided by this file.

getSectionTitle(section: Node): String

Gets the title of the provided section.

getSectionLink(section: Node): String

Gets the link for displaying the provided section.

isActiveSection(section: Node): String

Returns **true** if the provided section is the active section. # Freemarker functions for News items

Import Path /ccm-cms-types-newsitem.ftl

getLead(item: Node): String

Returns the lead text of the provided news item.

getMainText(item: Node): HtmlString

Returns the main text of the news item.

getNewsDate(item: Node): Node

Returns the date of the news. For formatting the date the `formatDateTime` function from the `utils.ftl` library can be used.

getNewsDateYear(item: Node): String

Gets the value of the `year` property of the news date.

getNewsDateMonth(item: Node): String

Gets the value of the `month` property of the news date.

getNewsDateDay(item: Node): String

Gets the value of the `Day` property of the news date.

getNewsDateDayNameShort(item: Node): String

Gets the value of the `year` property of the news date as short day name.

newsDateHour(item: Node): String

Gets the value of the `hour` property of the news date.

newsDateMinute(item: Node): String

Gets the value of the `minute` property of the news date.

newsDateSecond(item: Node): String

Gets the value of the `second` property of the news date.

newsDateIso(item: Node): String

Gets the date of the news as ISO date. # Freemarker functions for SiteProxy items

Import Path /ccm-cms-types-siteproxy.ftl

getContent(item: Node): ?

Returns the content of the site proxy. The type of the return value depends on the external content the site proxy is showing. # Freemarker functions for retrieving data from the user banner component.

Import Path /ccm-core/user-banner.ftl

getGreeting(): String

Retrieves to the greeting value provided by the *UserBanner* component.

isLoggedIn(): boolean

Return `true` if the current user is logged and `false` otherwise.

isNotLoggedIn(): boolean

Return **true** if the current user is *not* logged and **false** otherwise.

getChangePasswordUrl(): String

Returns the URL where a authenticated user can change his or her password.

getLoginLink(): String

Returns the URL of the login page.

getLogoutLink(): String

Returns the URL for logging out.

getScreenName(): String

Returns the username of the current user. If the user is not authenticated the will return an empty string.

getUserGivenName(): String

Returns the given of the current user, if availabe. If the user is not authenticated the will return an empty string.

getUserFamilyName(): String

Returns the given of the current user, if availabe. If the user is not authenticated the will return an empty string.

Freemarker function for ObjectLists

Import Path /ccm-navigation/object-list.ftl

Many functions provided by this library have a parameter **listId**. In most cases the value is **itemList**.

getItems(listId: String): Sequence<Node>

Returns the items in the object list with the provided ID.

getObjectCount(listId: String): number

Returns then number of objects in the object list with the provided ID.

getPagniatorBaseUrl(listId: String): String

Gets the base URL of the list paginator.

getPaginatorBegin(listId: String): Number

Returns the index of the first item displayed.

getPaginatorEnd(listId: String): Number

Returns the index of the last item displayed.

getPageCount(listId: String): Number

Gets the number of pages of the object list with the provided ID.

getPageNumber(listId: String): Number

Gets the number of page displayed.

getPageParam(listId: String): String

Gets the name of the page param for the object list with the provided ID.

getPageSize(listId: String): Number

Gets the number of objects per page for the object list with the provided ID.

getPrevPageLink(listId: String): String

Gets the link to the previous page of the object list with the provided ID.

getNextPageLink(listId: String): String

Gets the link to the next page of the object list with the provided ID.

getFirstPageLink(listId: String): String

Gets the link to the first page of the object list with the provided ID.

getLastPageLink(listId: String): String

Gets the link to the last page of the object list with the provided ID.

getItemTitle(item: Node): String

Gets the title of a list item.

getItemLead(item: Node): String

Gets the lead text of a list item.

getItemProperty(item: Node, property: String): String

A generic function the get the value of the property with the name provided the property parameter.

hasImage(item: Node): boolean

Determines if the provided list item has an image attachment.

getImageId(item: Node): String

Gets the ID of the image attachment of the provided list item.

getImageUrl(item: Node): String

Gets the URL of the image attachment of the provided list item.

getImageCaption(item: Node): String

Gets the caption of the image attachment of the provided list item.

getFilters(listId: String): Sequence<Node>

Returns the filters for the current list.

getFilterLabel(filter: Node): String

Gets the label of the provided filter.

getFilterType(filter: Node): String

Gets the type of the provided filter.

getSelectFilterOptions(filter: Node): Sequence<Node>

Returns the options of the select filter. If the provided filter is not a filter of the type *select* an empty sequence is returned.

getSelectFilterOptionLabel(option: Node): String

Returns the label of the provided filter.

getCategoryFilterSearchString(filter: Node): String

Returns the search string for the provided category filter.

getCategoryFilterSeparator(filter: Node): String

Gets the separation character for the value of the provided category filter.

getCategoryFilterMultiple(filter: Node): boolean

Determines if the provided category allows multiple selections.

getCategoryFilterCategories(filter: Node): Sequence<Node>

Returns the categories for the provided category filter.

getCategoryFilterCategoryGroups(filter: Node): Sequence<Node>

Returns the category groups of the provided category filter.

getCategoryGroupLabel(group: Node): String

Returns the label of the provided category group.

**getCategoryFilterCategoryGroupsCategories(groups: Sequence<Node>):
Sequence<Node>**

Gets the categories of all category groups.

getCategoryFilterCategoryId(category: Node): String

Gets the ID of the provided category of a category filter.

getCategoryFilterCategoryLabel(category: Node): String

Gets the label of the provided category of a category filter.

Freemarker functions for Portal Workspaces

Import Path /ccm-portalworkspace.ftl

getPortals(): Sequence<Node>

Returns all available portals.

isSelected(portal: Node): boolean

Determines if the provided portal is selected.

getPortalLink(portal: Node): String

Gets the link for selecting the provided portal.

getPortalTitle(portal: Node): String

Returns the title of the provided portal.

getPortalEditForm(): Node

Returns the edit form for the selected portal.

getPortalLayoutForm(): Node

Returns the form for editing the layout of the selected portal.

getAddPageLink(): String

Returns the link for adding another portal.

getBasicPropertiesLink(): String

Gets the link for the basic properties of the selected portal.

getPortletsFromColumn(colNumber: String): Sequence<Node>

Returns the portals in the column colNumber.

getWorkspacePrimaryUrl(): String

Returns the primary URL of the portal workspace.

getMovePortletLeftLink(portlet: Node): String

Returns the link for moving the provided portlet left.

getMovePortletRightLink(portlet: Node): String

Returns the link for moving the provided portlet right.

getMovePortletUpLink(portlet: Node): String

Returns the link for moving the provided portlet up.

getMovePortletDownLink(portlet: Node): String

Returns the link for moving the provided portlet down.

getCustomizePortletLink(portlet: Node): String

Returns the link for customizing the portlet.

getDeletePortletLink(portlet: Node): String

Returns the link for deleting the portlet. # Contributing to MathJax

So you're interested in giving us a hand? That's awesome! We've put together some brief guidelines that should help you get started quickly and easily.

There are lots and lots of ways to get involved, this document covers:

- raising issues
 - bug reports
 - feature requests
 - change requests
- working on MathJax core
 - submitting pull requests
- testing and quality assurance
- writing documentation
- translation
- Conduct

Reporting An Issue

If you're about to raise an issue because you think you've found a problem with MathJax, or you'd like to make a request for a new feature in the codebase, or any other reason... please read this first.

The GitHub issue tracker is the preferred channel for bug reports, feature requests, change requests and submitting pull requests, but please respect the following restrictions:

- Please **search for existing issues**. Help us keep duplicate issues to a minimum by checking to see if someone has already reported your problem or requested your idea.
- Please **do not** use the issue tracker for personal support requests (use the MathJax User Group).
- Please **be civil**. Keep the discussion on topic and respect the opinions of others. See also our Conduct Guidelines

Bug Reports

A bug is a *demonstrable problem* that is caused by the code in the repository. Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. **Use the GitHub issue search** — check if the issue has already been reported.
2. **Check if the issue has been fixed** — try to reproduce it using the latest `develop` or look for closed issues in the current milestone.
3. **Isolate the problem** — ideally create a reduced test case and a live example.
4. **Include a screencast if relevant** - Is your issue about a design or front end feature or bug? The most helpful thing in the world is if we can *see* what you're talking about. Use LICEcap to quickly and easily record a short screencast (24fps) and save it as an animated gif! Embed it directly into your GitHub issue. Kapow.
5. Use the Bug Report template below or click this link to start creating a bug report with the template automatically.

A good bug report shouldn't leave others needing to chase you up for more information. Be sure to include the details of your environment.

Here is a real example

Template Example (click to use):

Short and descriptive example bug report title

Issue Summary

A summary of the issue and the browser/OS environment in which it occurs. If suitable, include the steps required to reproduce the bug.

Steps to Reproduce

1. This is the first step
2. This is the second step
3. Further steps, etc.

Any other information you want to share that is relevant to the issue being reported. Especially, why do you consider this to be a bug? What do you expect to happen instead?

Technical details:

* MathJax Version: 2.3 (latest commit: f3aaf3a2a3e964df2770dc4aaaa9c87ce5f47e2c)
* Client OS: Mac OS X 10.8.4
* Browser: Chrome 29.0.1547.57

Feature Requests

Feature requests are welcome. Before you submit one be sure to have:

1. Read the Roadmaps, **use the GitHub search** and check the feature hasn't already been requested.
2. Take a moment to think about whether your idea fits with the scope and aims of the project, or if it might better fit being a custom extension.
3. Remember, it's up to *you* to make a strong case to convince the project's leaders of the merits of this feature. Please provide as much detail and context as possible, this means explaining the use case and why it is likely to be common.
4. Clearly indicate whether this is a feature request for MathJax core, input & output jax, or extensions.

Change Requests

Change requests cover both architectural and functional changes to how MathJax works. If you have an idea for a new or different dependency, a refactor, or an improvement to a feature, etc - please be sure to:

1. **Use the GitHub search** and check someone else didn't get there first
2. Take a moment to think about the best way to make a case for, and explain what you're thinking. Are you sure this shouldn't really be a bug report or a feature request? Is it really one idea or is it many? What's the context? What problem are you solving? Why is what you are suggesting better than what's already there? Does it fit with the Roadmap?

Submitting Pull Requests

Pull requests are awesome. If you're looking to raise a PR for something which doesn't have an open issue, please think carefully about raising an issue which

your PR can close, especially if you're fixing a bug. This makes it more likely that there will be enough information available for your PR to be properly tested and merged.

Need Help?

If you're not completely clear on how to submit / update / *do* Pull Requests, please check out our source control policies. For more insights, check the excellent in depth Git Workflow guide from Ghost, in particular

- Ghost Workflow guide: commit messages

Testing and Quality Assurance

Never underestimate just how useful quality assurance is. If you're looking to get involved with the code base and don't know where to start, checking out and testing a pull request is one of the most useful things you could do.

If you want to get involved with testing MathJax, there is a set of QA Documentation in our testing framework.

Essentially though, check out the latest develop branch, take it for a spin, and if you find anything odd, please follow the bug report guidelines and let us know!

Checking out a Pull Request

These are some excellent instructions on configuring your GitHub repository to allow you to checkout pull requests in the same way as branches: <https://gist.github.com/piscisaureus/3342247>.

Documentation

MathJax's main documentation can be found at docs.mathjax.org.

The documentation is generated using Sphinx-doc and hosted on Read the docs. The source of the docs is hosted in the MathJax-Docs GitHub repository.

You can clone the repo and submit pull requests following the pull-request guidelines.

Translation

If you wish to add or update translations of MathJax, please do it on TranslateWiki.net (and while you're there you can help other open source projects, too!).

For bug reports and other questions that don't fit on TranslateWiki.net, head over to the [mathjax/mathjax-i18n](https://github.com/mathjax/mathjax-i18n) repository.

Working on MathJax Core {core}

Key Branches & Tags

- **develop** is the development branch. All work on the next release is here. Do **NOT** use this branch for a production site.
- **master** contains the latest release of MathJax. This branch may be used in production.

Conduct

We are committed to providing a friendly, safe and welcoming environment for all, regardless of gender, sexual orientation, disability, ethnicity, religion, or similar personal characteristic.

Please be kind and courteous. There's no need to be mean or rude. Respect that people have differences of opinion and that every design or implementation choice carries a trade-off and numerous costs. There is seldom a right answer, merely an optimal answer given a set of values and circumstances.

Please keep unstructured critique to a minimum. If you have solid ideas you want to experiment with, make a fork and see how it works.

We will exclude you from interaction if you insult, demean or harass anyone. That is not welcome behaviour. We interpret the term "harassment" as including the definition in the Citizen Code of Conduct; if you have any lack of clarity about what might be included in that concept, please read their definition. In particular, we don't tolerate behavior that excludes people in socially marginalized groups.

Private harassment is also unacceptable. No matter who you are, if you feel you have been or are being harassed or made uncomfortable by a community member, please contact one of the channel ops or any of the MathJax core team immediately. Whether you're a regular contributor or a newcomer, we care about making this community a safe place for you and we've got your back.

Likewise any spamming, trolling, flaming, baiting or other attention-stealing behaviour is not welcome.

We also suggest to read discourse's rules

References

- We heavily borrowed from – thanks to Mozilla and Ghost!
- <https://github.com/TryGhost/Ghost/blob/master/CONTRIBUTING.md>
- <https://github.com/mozilla/rust/wiki/Note-development-policy>
- <https://github.com/jden/CONTRIBUTING.md/blob/master/CONTRIBUTING.md>
- <http://blog.discourse.org/2013/03/the-universal-rules-of-civilized-discourse/>

MathJax

Beautiful math in all browsers

MathJax is an open-source JavaScript display engine for LaTeX, MathML, and AsciiMath notation that works in all modern browsers. It was designed with the goal of consolidating the recent advances in web technologies into a single, definitive, math-on-the-web platform supporting the major browsers and operating systems. It requires no setup on the part of the user (no plugins to download or software to install), so the page author can write web documents that include mathematics and be confident that users will be able to view it naturally and easily. Simply include MathJax and some mathematics in a web page, and MathJax does the rest.

Some of the main features of MathJax include:

- High-quality display of LaTeX, MathML, and AsciiMath notation in HTML pages
- Supported in most browsers with no plug-ins, extra fonts, or special setup for the reader
- Easy for authors, flexible for publishers, extensible for developers
- Supports math accessibility, cut-and-paste interoperability, and other advanced functionality
- Powerful API for integration with other web applications

See <http://www.mathjax.org/> for additional details.

Installation and Usage

The MathJax installation and usage documentation is available in the `docs/html` directory of the MathJax distribution (see `docs/html/index.html` for the starting point). The documents are also available on the MathJax web site on line at <http://www.mathjax.org/resources/docs/>.

Community

The main MathJax website is <http://www.mathjax.org>, and it includes announcements and other important information. MathJax is maintained and distributed on GitHub at <http://github.com/mathjax/MathJax>. A user forum for asking questions and getting assistance is hosted at Google, and the bug tracker is hosted at GitHub:

Bug tracker: <https://github.com/mathjax/MathJax/issues>

MathJax-Users Group: <http://groups.google.com/group/mathjax-users>

Before reporting a bug, please check that it has not already been reported. Also, please use the bug tracker for reporting bugs rather than the help forum.

Freemaker functions for SQL member lists

Import Path /ccm-sci-member-navigation.ftl

getSciMemberList(listId: String = "itemList"): Node

Gets the member list. Default is to use the list with the name `itemList`. The name can be overridden using the optional `listId` parameter.

getMembers(list: Node): Sequence<Node>

Returns the members in the provided list.

getSurnameFilterValue(list: Node): String

Gets the value of the surname filter of the provided list.

getCount(list: Node): String

Gets the number of items in the list.

getCurrentPage(list: Node): String

Returns the number of the current page of the list.

getLimit(list: Node): String

Gets the maximum number of items per page.

getMaxPages(list: Node): String

Returns the number of pages.

getNextPageLink(list: Node): String

Returns the link to the next page of the list.

getPreviousPageLink(list: Node): String

Returns the link to the previous page of the list.

getOffset(list: Node): String

Returns the index of the items shown.

getMemberItemId(item: Node): String

Returns the ID of the provided member item.

getMemberItemName(item: Node): String

Returns the name of the provided member item.

getMemberItemTitle(item: Node): String

Returns the value of the title property of the provided member item.

getMemberItemSurname(item: Node): String

Returns the value of the surname property of the provided member item.

getMemberItemGivenName(item: Node): String

Returns the value of the given name property of the provided member item.

getMemberItemTitlePre(item: Node): String

Returns the value of the titlePre property of the provided member item.

getMemberItemTitlePost(item: Node): String

Returns the value of the titlePost property of the provided member item.

getMemberItemContactEntries(item: Node): Sequence<Node>

Gets the contact entries of the provided member item.

getMemberItemContactEntry(item: Node, key: String): Node

Gets the contact entry with the provided key of the provided member item.

Freemarker functions for SQL project lists

Import path /ccm-sci-project-navigation.ftl

getSciProjectList(listId: String = "itemList"): Node

Returns an project list. The list can be selected by the optional `listId` parameter. The default value for the parameter is `itemList`.

getProjects(list: Node): Sequence<Node>

Returns the projects in a project list.

getTitleFilterValue(list: Node): String

Gets the value of the title filter of the provided list.

getResearchFieldFilterValue(list: Node): String

Returns the value of the research field filter of the provided list.

getCount(list: Node): String

Returns the number of projects in the provided list.

getCurrentPage(list: Node): String

Returns the number of the current page of the list.

getLimit(list: Node): String

Gets the maximum number of items per page.

getMaxPages(list: Node): String

Gets the number of page of the provided list.

getNextPageLink(list: Node): String

Gets the link to the next page of the provided list.

getPrevPageLink(list: Node): String

Gets the link to the previous page of the provided list.

getOffset(list: Node): String

Gets the index of the first item on the current page.

getProjectItemId(item: Node): String

Returns the ID of the provided project item.

getProjectItemName(item: Node): String

Returns the name of the provided project item.

getProjectItemTitle(item: Node): String

Returns the value of the `title` property of the provided project item.

getProjectItemObjectType(item: Node): String

Returns the value of the object type of the provided project item.

getProjectItemBegin(item: Node): String

Returns the value of the `begin` property of the provided project item.

getProjectItemBeginDay(item: Node): String

Returns the value of the `day` property of begin date of the provided project item.

getProjectItemBeginMonth(item: Node): String

Returns the value of the `month` property of begin date of the provided project item.

getProjectItemBeginYear(item: Node): String

Returns the value of the `year` property of begin date of the provided project item.

getProjectItemEnd(item: Node): String

Returns the value of the `end` property of the provided project item.

getProjectItemEndDay(item: Node): String

Returns the value of the `day` property of end date of the provided project item.

getProjectItemEndMonth(item: Node): String

Returns the value of the `month` property of end date of the provided project item.

getProjectItemEndYear(item: Node): String

Returns the value of the `year` property of end date of the provided project item.

getProjectItemShortDesc(item: Node): String

Returns the value of the `short-desc` property of the provided project item.

getProjectItemMembers(item: Node): Sequence<Node>

Returns the members of the project.

getProjectMemberSurname(member: Node): String

Returns the surname of the provided member.

getProjectMemberGivenname(member: Node): String

Returns the given name of the provided member.

Freemarker functions for Article In items

Import Path /ccm-sci-publications/article.ftl

This functions are for processing items of types `ArticleInCollectedVolume`, `ArticleInJournal` and `InProceedings`.

getHref(article: Node): String

Generates the link to the detail view of the provided article item. # Freemarker functions for processing the authors of a publication item

Import Path /ccm-sci-publications/authors.ftl

getLink(author: Node, keyId: String): String

Gets the link to the homepage of the author from the contact entries of the author. The key of the contact entry to use is selected using the `keyId` parameter.

getId(author: Node): String

Returns the ID of the author as a string usable as value of the `id` attribute of a HTML element. The returned string consists of the ID of the master version of the author item, and the name of the author, separated by an underscore.

getPosition(author: Node): String

Returns the position of provided author item in the sequence of authors.

isLast(author: Node): boolean

Determines if the provided author is the last author in the sequence of authors.

getSurname(author: Node): String

Gets the surname of the author.

getGivenName(author: Node): String

Gets the given name of the author.

isEditor(author: Node): boolean

Determines if the provided author is an editor. # Freemarker functions for processing Collected Volume items

Import Path /ccm-sci-publications/collected-volume.ftl

getHref(collectedVolume: Node): String

Returns the link to the detail view of the provided collected volume item. # Freemarker functions for generating the exportLink links of an publication item

Import Path /ccm-sci-publications/export-links.ftl

getHref(exportLink: Node): String

Returns the URL for for the provided export link.

getFormatKey(exportLink: Node): String

Gets the key of the format provided by the export link provided by the exportLink parameter.

getFormatName(exportLink: Node): String

Gets the name of the format provided by the export link provided by the `exportLink` parameter. # Freemarker functions for Journal items

Import Path /ccm-sci-publications/journal.ftl

getFirstYear(journal: Node): String

Gets the value of the `firstYear` property of the provided journal.

getHref(journal: Node): String

Returns the link to the detail view of the journal.

getIssn(journal: Node): String

Gets the ISSN of the journal.

getIssn(journal: Node): String

Gets the value of the `lastYear` property of the journal.

getTitle(journal: Node): String

Returns the title of the journal. # Freemarker functions for processing library signatures

Import Path /ccm-sci-publications/library-signatures.ftl

A library signature object contains the library signature (usually an alpha numeric code) and additional information.

getLibrary(signature: Node): String

Returns the the library of the signature.

getSignature(signature: Node): String

Returns the signature itself.

getLibraryLink(signature: Node): String

Returns the link to the homepage of the library.

getMisc(signature: Node): String

Returns the value of the misc property of the signature.

Freemarker functions for processing Orderer objects

Import Path /ccm-sci-publications/orderer.ftl

getName(orderer: Node): String

Gets the name of the orderer. # Freemarker functions of processing proceedings

Import Path /ccm-sci-publications/proceedings.ftl

getHref(proceedings: Node): String

Gets the link to the detail view of the provided proceedings item.

getPaperHref(paper: Node): String

Returns the link to the detail view of the provided paper. # Freemarker functions for native SQL based publication lists

Import Path /ccm-sci-publications-navigation.ftl

getSciPublicationsList(listId: String = "itemList"): Node

Retrieves a publications list. The list to use can be selected using the optional listId parameter.

getPublications(list: Node): Sequence<Node>

Returns the publications of the provided publication list.

getTitleFilterValue(list: Node): String

Returns the value of the title filter of the provided publication list.

getYearOfPublicationFilterValue(list: Node): String

Returns the value of the year of publication filter of the provided publication list.

getAuthorsFilterValue(list: Node): String

Returns the value of the authors filter of the provided publication list.

getSort(list: Node): String

Returns the property which is used to sort the list.

getCount(list: Node): String

Returns the number of publications in the list.

getCurrentPage(list: Node): String

Gets the number of the current page of the list.

getLimit(list: Node): String

Returns the maximum number of publications per page.

getMaxPages(list: Node): String

Returns the number of pages of the list.

getNextPageLink(list: Node): String

Returns the link to the next page of the list.

getNextPageLink(list: Node): String

Returns the link to the previous page of the list.

getOffset(list: Node): String

Gets the index of the first publication of the current page.

getPublicationId(item: Node): String

Returns the ID of the provided publication item.

getPublicationObjectType(item: Node): String

Returns the type of the provided publication item.

getPublicationTitle(item: Node): String

Returns the title of the provided publication item.

getPublicationYear(item: Node): String

Gets the year of publication of the publication.

getPublicationAuthors(item: Node): Sequence<Node>

Gets the authors of the publication.

hasAuthorSurname(author: Node): boolean

Determines if the provided author has a surname.

getAuthorSurname(author: Node): String

Gets the surname of the author.

hasAuthorGivenName(author: Node): boolean

Determines if the provided author has a given name.

getAuthorGivenName(author: Node): String

Gets the given name of the author.

getPublicationPlace(item: Node): String

Gets the value of the place property of the publication.

getPublicationOrganization(item: Node): Node

Getsh the organization assigned to a publication.

getPublicationOrganizationName(item: Node): String

Gets the name of the organization.

getPublicationUnpublishedPlace(item: Node): String

Gets the place of the publication of the type UnPublished.

getPublicationPublisher(item: Node): String

Gets the publisher of the publication.

getPublisherPlace(item: Node): String

Gets the place of the publisher.

getPublisherName(item: Node): String

Gets the name of the publisher.

getPublicationJournal(item: Node): Node

Gets the journal to which the publication is assigned.

getJournalName(journal: Node): String

Gets the name of the journal.

getPublicationIssue(item: Node): String

Gets the issue in which the publication was published.

hasPublicationVolumeOfJournal(item: Node): String

Determines if the publication has a value for the volume property.

getPublicationVolumeOfJournal(item: Node): String

Returns the value of the volume property.

getPublicationPagesFrom(item: Node): String

Gets the value of the pageFrom property.

getPublicationPagesTo(item: Node): String

Gets the value of the pageTo property.

getPublicationCollectedVolume(item: Node): Node

Gets the collected volume to which the publication is assigned.

getCollectedVolumeAuthors(collectedVolume: Node): Sequence<Node>

Returns the authors/editors of the collected volume.

getCollectedVolumeTitle(collectedVolume: Node): String

Returns the title of the collected volume.

getCollectedVolumePublisher(collectedVolume: Node): Node

Returns the publisher of the collected volume.

getCollectedVolume(collectedVolume: Node): String

Gets the place of the collected volume.

hasProceedings(item: Node): boolean

Determines if the publication has proceedings.

getProceedings(item: Node): Sequence<Node>

Returns the proceedings the publication.

Freemarker functions for publication items.

Import Path /ccm-sci-publications/publications.ftl

getAssignedTermsDomains(item: Node, domain: String): Sequence<Node>

Returns the categories from the category system with the name provided by the domain parameters which are assigned to the publication.

getAuthors(item: Node): Sequence<Node>

Returns the authors of the publication.

getPublisher(item: Node): Sequence<Node>

Returns the publisher of the publication.

getYearOfPublication(item: Node): Sequence<Node>

Returns the year of publication.

getNumberOfPages(item: Node): String

Gets the number of pages of the publication.

getNumberOfVolumes(item: Node): String

Gets the number of volumes of the publication.

getVolume(item: Node): String

Gets the value of the volume property of the publication.

getEdition(item: Node): String

Get the edition of the publication.

getIsbn(item: Node): String

Gets the ISBN of the publication.

getLanguageOfPublication(item: Node): String

Gets the language of the publication.

getSeries(item: Node): Node

Gets the series to which the publication is assigned.

isReviewed(item: Node): boolean

Determines if the publication is reviewed.

getAbstract(item: Node): String

Returns the abstract of the publication.

getMisc(item: Node): String

Returns the value of the misc property of the publication.

getExportLinks(item: Node): Sequence<Node>

Returns the export links for the publication.

getPlace(item: Node): String

Returns the value of the `place` property of the publication.

getPagesFrom(item: Node): String

Returns the value of the `pagesFrom` property of the publication.

getPagesTo(item: Node): String

Returns the value of the `pagesTo` property of the publication.

getNumber(item: Node): String

Returns the value of the `number` property of the publication.

getYearFirstPublished(item: Node): String

Returns the value of the `yearFirstPublished` property of the publication.

getLibrarySignatures(item: Node): Sequence<Node>

Returns the library signatures assigned to a publication.

getOrganization(item: Node): Node

Gets the organization assigned to a publication.

getOrderer(item: Node): Node

Gets the orderer assigned to a publication.

getIssn(item: Node): Node

Gets the ISSN of a publication.

getLastAccessed(item: Node): String

Gets the value of the `lastAccessed` property of a publication.

getUrl(item: Node): String

Gets the value of the url property of a publication.

getUrn(item: Node): String

Gets the value of the urn property of a publication.

getDoi(item: Node): String

Gets the value of the doi property of a publication.

getIssue(item: Node): String

Gets the issue of a publication.

getJournal(item: Node): Node

Gets the journal to which a publication is assigned.

getCollectedVolume(item: Node): Node

Gets the collected volume to which a publication is assigned.

getChapter(item: Node): String

Gets the value of the chapter property of a publication.

getNameOfConference(item: Node): String

Gets the name of the conference.

getPlaceOfConference(item: Node): String

Gets the place of the conference.

getDateFromConference(item: Node): DateNode

Gets the start date of the conference.

getDateToConference(item: Node): DateNode

Gets the end date of the conference.

getProceedings(item: Node): Node

Gets the proceedings to which an publication is assigned.

getProceedingsPapers(item: Node): Sequence<Node>

Gets the papers of a proceedings publication item.

getSeriesVolume(item: Node): String

Gets the volume of the series for a publication.

getEvent(item: Node): String

Gets the value of the `event` property of a publication of the type *talk*.

getDateOfTask(item: Node): DateNode

Gets the value of the `date` property of a publication of the type *talk*.

Freemarker functions for processing Publishers

Import Path /ccm-sci-publications/publisher.ftl

getName(publisher: Node): String

Gets the name of a publisher.

getPlace(publisher: Node): String

Gets the place of a publisher. # Freemarker functions for Series

Import Path /ccm-sci-publications/series.ftl

getFilters(series: Node): Sequence<Node>

Returns the filters for list of publications of the series. The filters can be procesed by the functions provided for object list filters.

getLink(series: Node): String

Returns the link to the detail view of the series.

getName(series: Node): String

Returns the name of a series.

getVolume(series: Node): String

Gets the value of the volume property.

getVolumeHref(volume: Node): String

Gets a link to a volume of a series.

Freemarker functions for SciDepartment items

Import Path /ccm-sci-types-department.ftl

getDescription(data: Node): HtmlString

Returns the description of the department.

getShortDescription(data: Node): String

Returns the short description of the department.

getDepartmentHeads(data: Node): Sequence<Node>

Gets the heads of the department.

getDepartmentHeadId(head: Node): String

Gets the ID of a head of a department.

getDepartmentHeadLink(head: Node): String

Gets the link to the detail view of head of a department.

getDepartmentViceHeads(data: Node): Sequence<Node>

Gets the vice heads of the department.

getDepartmentViceHeadId(head: Node): String

Gets the ID of a vice head of a department.

getDepartmentViceHeadLink(head: Node): String

Gets the link to the detail view of vicehead of a department.

getDepartmentSecretariats(data: Node): Sequence<Node>

Gets the secretariats of the department.

getDepartmentSecretariatId(sec: Node): String

Gets the ID of a secretariats of a department.

getDepartmentSecretariatLink(sec: Node): String

Gets the link to the detail view of secretariats of a department.

getProjects(data: Node): Sequence<Node>

Returns the list of projects assigned to the department.

getProjectId(project: Node): String

Gets the id of a project.

getProjectLink(project: Node): String

Returns the link to the detail view of a project.

Freemarker functions for SciDepartment items

Import Path /ccm-sci-types-institute.ftl

getDescription(data: Node): HtmlString

Returns the description of the institute.

getShortDescription(data: Node): String

Returns the short description of the institute.

getDepartments(data: Node): Sequence<Node>

Returns the departments assigned to a institute.

getDepartmentOid(department: Node): String

Gets the OID of a department.

getDepartmentTitle(department: Node): String

Gets the title of a department.

getDepartmentLink(department: Node): String

Returns the link to the detail view of a department. # Freemarker functions for SciProject items

Import Path /ccm-sci-types-project.ftl

getBegin(item: Node): DateNode

Returns the begin date of the project. To format the date the `formatDateTime` function provided by `ccm-themedirector` should be used.

getEnd(item: Node): DateNode

Returns the end date of the project. To format the date the `formatDateTime` function provided by `ccm-themedirector` should be used.

getDescription(item: Node): HtmlString

Gets the description of the project.

getShortDescription(item: Node): String

Gets the short description of the project.

getSponsors(item: Node): Sequence<Node>

Returns the sponsors of the project.

getSponsorName(sponsor: Node): String

Returns the name of the sponsor.

hasSponsorFundingCode(sponsor: Node): boolean

Determines if the sponsor has assigned a funding code to the project.

getSponsorFundingCode(sponsor: Node): String

Returns the funding code of the project assigned by the sponsor.

getSponsorLink(sponsor: Node): String

Gets the link to the homepage of the sponsor.

hasFunding(item: Node): String

Determines if the project has a text describing the funding of the project.

getFunding(item: Node): HtmlString

Gets the text describing the funding of the project.

hasFundingVolume(item: Node): boolean

Determines if the project has a value for the `fundingVolume` property.

getFundingVolume(item: Node): String

Returns the value of the `fundingVolume` property.

getMembers(item: Node): Sequence<Node>

Returns the members of the project.

getMemberRole(member: Node): String

Gets the role of the member.

getMemberStatus(member: Node): String

Gets the status of the member.

getMemberId(member: Node): String

Gets the ID of the member.

getMemberLink(member: Node): String

Gets a link to the detail view of the member.

getMemberSurname(member: Node): String

Gets the surname of the member.

getMemberGivenName(member: Node): String

Gets the given name of the member.

getMemberTitlePre(member: Node): String

Gets the value of the `titlePre` property of the member.

getMemberTitlePost(member: Node): String

Gets the value of the `titlePost` property of the member.

getInvolvedOrganizations(item: Node): Sequence<Node>

Gets the organizations involved in the project.

getInvolvedOrganizationName(orga: Node): String

Gets the name of the organization.

getInvolvedOrganizationLink(orga: Node): String

Gets the link to the homepage of the organization. # Freemarker functions for subsites

Import Path /ccm-subsite.ftl

getSubsiteName(): String

Returns the name of the current subsite. # Freemarker functions for language related tasks.

Import Path :/ccm-themedirector/language.ftl

getAvailableLanguages(): Sequence<String>

Returns the available languages for the current document as sequence. This sequence can be used for creating links for selecting the language:

```

<ul class="language-selector">
  <#list Lang.getAvailableLanguages()?sort as lang>
    <li class="${(lang==negotiatedLanguage)?then('selected', '')}">${lang}</li>
  </#list>
</ul>

```

This example uses the `list` directive from Freemarker to iterate over the available languages returned by `getAvailableLanguages`. The Freemarker build-in `?then` is used together with the `negotiatedLanguage` variable to check if the current language is the selected language. If this is the case a CSS class is added to the HTML.

Utility functions

Import Path : `/ccm-themedirectory/utils.ftl`

`getPageApplication(): String`

Return the application of the current page.

`getPageTitle(): String`

Return the title of the current page as provided by the *Category Menu* Component of the *ccm-navigation* module.

`getSiteHostName(): String`

Returns the host name of the CCM installation as provided by the *SiteBanner* component of the *ccm-core* module.

`getSiteName(): String`

Returns the host name of the CCM installation as provided by the *SiteBanner* component of the *ccm-core* module.

`getBooleanAttrValue(fromNode: Node, attrName: String): boolean`

A helper function which tries to convert the value of the attribute `attrName` of the node `fromNode` to a boolean. The following values are interpreted as `true`: `true`, `yes`. All other values are interpreted as `false`.

formatDateTime(style: String, date: DateValueNode): String

Formats the value of date/time value node according to the provided **style**. The is defined in the theme manifest in the **date-time-formats** section. It is possible to define different styles for different languages. The style definition in the theme manifest must be in the format expected by the Java `DateTimeFormatter` class.

Example

In the theme manifest in the following format is defined:

```
"date-time-formats": [  
  ...  
  {  
    "style": "news",  
    "lang": "de",  
    "format": "dd.MM.YYYY"  
  },  
  {  
    "style": "news",  
    "lang": "en",  
    "format": "MM/dd/YY"  
  },  
  ...  
]
```

Each style must have a name. It is possible to have different patterns for a style for different languages. The pattern itself is provided by the **format** property. For a documentation of the pattern format please refer to the documentation of the Java `DateTimeFormatter`.

The second parameter of these function is a date value, at the moment this is an XML node if several attributes providing the year, month etc. of the date. This value is usually provided by special function for the specific content type. A typical usage of the **formatDateTime** function looks like this:

```
<span>${Utils.formatDateTime("standard", News.getNewsDate(item))}</span>
```

In this example the **getNewsDate** function was used to retrieve the date of a news.

News.getNewsDate gets the date of a news item. If the date of the news is 2019-04-01 the return value of the function for german is

01.04.2019

and for english:

4/1/19