MOKIO FRONTEND HELPERS

* Helpers marked with [S] are available only with gem mokio_skins

1. <head> includes

- include_skin_css_all [S] – includes all css that are available in the skin (css folder)

- include_skin_css (name) [S] – includes css file with given name

- include_skin_js_all [S] – includes all javascripts files available in the skin (js folder)

- include_skin_js (name) [S] - includes javascript file with given name

- include_jquery_ui [S] – includes a file with jquery.ui.core library

- include_meta [S] – include meta tags. Values should be set in Mokio backend for given article or menu. By default, for list of contents meta is taken from menu, for home page from first content and for single content from this content.

2. Menu

- build_menu (initial_id, position, limit = 1, with_nav = true, nav_class="navmenu", hierarchical = false) – builds menu tree for specified arguments, returns html.

Parameters:

initial_id - root's id
position - menu position, root child name or id
limit - how deep should builder look for children, count starts after position
hierarchical - specifies if you want to use hierarchical links or not

By default Mokio has two languages enabled: polish and english
Currently selected language is kept in frontend_default_lang variable

Each language version of the site has its own menu – that can be edited from the Mokio backend Menu is represented as tree structure. Root node is the language and its children are the positions on which menu can be displayed, i.e. PL → TOP. Only second and lower levels children are real menu elements that are displayed on the website.

If hierarchical is set to false (default behavior), generated links have the following structure: my.domain.com/:menu_id and to use it in frontend you need to specify following route in your routes.rb:
get “/menu:id” => “contents#show”

If hierarchical is set to true, generated links represent all the structure to the root node. For example:
For following menu:

- level1
    - level2
        - level3
- level1a

Generated links will look as follows:

/level1
/level1/level2
/level1/level2/level3
/level1a

To handle it properly in frontend you need to add following lines to your routes.rb:

```
get "/*menu_path/:menu_id" => "content#show"
get "/:menu_id" => "content#show"
```

Menu cannot exist without a language. When you add new localization (language) in the backend - main menu structure is automatically created.

Adding new language

To add new language, navigate to:
Mokio → Language settings → Add new language

Adding new menu element

To add new menu element navigate to
Mokio → Menu → Add menu

Choose requested language.

Active – specifies is menu (link to the assigned contents) should be active – so is clicking on it in frontend actually shows the contents
Visible – specifies if menu (link to the assigned contents) should be generated when generating menu structure. If set to false (red cross), clicking on the link in frontend actually shows the contents, however link is not visible in menu. This can be usefull when you would like to have links to your contents in other contents – menu link can be pasted to HTML editor of given content's content field.

Choose if menu is an external URL or a link to contents from your site.

Parent element – parent of given menu – for yout main nodes it should be menu position (TOP from the example above).

Target – specifies whether the contents of this menu should be opened in current window (as a part of the website – default behavior), in the same window but as a whole page or in new window/tab.

Example:

Let's create new menu element and name it menu1. Then let's add to its children: menu1-a and menu1-b.

= build_menu("top",100)

or only menu1-a and menu1-b:

= build_menu("menu1",1)

- build_menu_by_name (initial_name, limit) - builds menu tree for specified arguments, returns html

Parameters:
initial_name - parent menu position name
limit - how deep should builder look for children, count starts after position

- build_items (item, limit, index) - recursive building menu items

Parameters:
item - Mokio::Menu object

limit - how deep should builder look for children
index - how deep is function already

- create_menu – alias for **build_menu**

- isMenu?(obj) - raises IsNotAMokioMenuErrorr if obj isn't a Mokio::Menu object

- menu_content_all (menu) - returns all contents added to given menu – regardless if they are active or not
- menu_content (menu, limit = nil) - returns active contents added to menu

Parameters:

menu - Mokio::Menu object
limit - Limit contents count

- menu_content_titles - returns active contents titles added to menu

Parameters:

menu - Mokio::Menu object
limit - Limit contents count

3. HTML blocks

- menu_static_modules(menu, limit) – returns static_modules added to menu

Parameters:

menu - Mokio::Menu object
limit - Limit contents count

- menu_static_modules_titles - returns static_modules titles added to menu

Parameters:

menu - Mokio::Menu object
limit - Limit static modules count

- menu_slug (menu) – returns menu slug

- menu_locale_root_id - returns menu root id for active locale (language)

- menu_root_id - returns menu root id for given name

- build_static_modules - displays HTML blocks for given position. HTML blocks are a pieces of HTML that can be associated with given pages and given positions on the page.
For one position we can have many HTML blocks. One HTML block can be associated with many positions.
Position to which HTML block is assigned can have its own template, that will be used to generate final HTML. If a position doesn't have any template, HTML block content will be displayed using default template (nothing will be added to defined HTML code).
Both position and HTML block can be activated or deactivated. To see HTML block, both position and HTML block must be active.
To display HTML block, it has to be assigned at least to one position.

Parameters:

position name – name of the position, for which HTML blocks will be displayed.

Example:

=build_static_modules ("footer") - displays all blocks assigned to “footer” position

- build_from_content(content, with_intro = true) - builds html for a single static_module, without specific template, with or without intro

Parameters:

content - single static module from result

- build_from_view_file(content, tpl) - builds html for a single static_module with tpl template and render

- build_content(obj, position, with_intro = true) - checks visibility and generate static modules tree from Mokio::StaticModule object

Parameters:

obj - static modules object

position – module position object

4. Content

- isContent?(obj) - raises IsNotAMokioContentError if obj isn't a Mokio::Content object

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)

- content_title (obj) - returns title field for given object

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)

- content_intro (obj) - returns intro field for given object as html

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)

- content_content (obj) - returns content field for given object as html

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)

- content_main_pic(obj, version = nil) - returns main picture for given object as html

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)
version – version of the picture (version available by default are as follows: “edit” - 100px x 100px and “main_pic” - 150px x 150px)

- main_pic_url (obj, version = nil) - returns main picture url for given object as string

Parameters:

obj - Mokio::Content object (including inherited Mokio::Article etc..)

version – version of the picture (version available by default are as follows: “edit” - 100px x 100px and “main_pic” - 150px x 150px)

5. External scripts

- build_external_script (name) – includes given external_script (by its name) as javascript tag. Can be useful for instance for Google Analytics code.

Adding new external script

In Mokio backend navigate to:

Mokio → External Scripts → Add new external script

- build_all_external_scripts – includes all external scripts from mokio_external_scripts

- build_common(obj) – build a single external script from Mokio::ExternalScript object

6. Various

- render_template [S] – renders given template from templates folder