MOKIO SKINS

1. Creating a skin for your website with Mokio CMS

a. Open Mokio backend: localhost:3000/backend

b. Login with default admin account:

login: admin@admin.com

password: admin

c. Navigate to **Skins → Add new skin**

d. Prepare skin packed with zip with the following structure:

```
[ZIP] skin_name
[FOLDER] skin_name
-[FOLDER] templates – main skin templates (currently available
formats: erb, haml, slim)
--default.html.erb
--layout.html.erb
--home.html.erb
--article.html.erb
--pic_gallery.html.erb
--mov_gallery.html.erb
--contact.html.erb
--list.html.erb
-[FOLDER] js (javascripts for your skin)
--file.js
--file2.js
-[FOLDER] css (css stylesheets for yous skin)
--style.css
--style2.css
-[FOLDER] images (pictures, photos, graphics for your skins)
(jpg,gif,png)
--img1.jpg
--img2.png
--img3.gif
```

Requirements for the skin:

- ZIP filename needs to be the same as skin name
- Skin name has to be unique per system
- Supported styles formats: css
- Supported scripts formats: js
- Supported pictures formats: jpg, gif, png

Skin templates are HTML files with pieces of ruby code with various helpers provided by Mokio Skins (see point 6). More information about erb, haml and slim format can be found here:

ERB: http://www.stuartellis.eu/articles/erb/

HAML: http://haml.info/

SLIM: http://slim-lang.com/

2. Activate your skin

To activate your skin, navigate to: **Skins → Edit skin**

a. Change Active field to On (green)

b. Save the skin

3. Edit skin

To edit your skin, navigate to: **Skins → Edit skin**

On the left there is a list of skin files available for edit.

Useful shortcuts:

- **[Ctrl R]** - display hints in editor
- **[Ctrl S]** or Save File – save your changes in given file
- **[Ctrl Z]** - undo changes

4. Helpers available in editor [Ctrl R]

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

- include_skin_css (name) – includes css file with given name

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

- include_skin_js (name) - includes javascript file with given name

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

- include_meta – include meta tags. Values should be set in 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.

- render_template – renders given template from templates folder

- build_menu (initial_id, position, limit = 1, with_nav = true,

nav_class="navmenu") – 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

- 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

- menu_static_modules – 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

- 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)

- build_external_script – includes given external_script (by its name)

- build_all_external_scripts – includes all external scripts from mokio_external_scripts

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