Notes from my first light development

Here I collected links/snippets/remarks and issues that I deemed important enough to note down while building a front-end with magnolia as recommended in the version 5.7: with Magnolia CLI and light module front-end development.

Installation

1. install magnolia cli: `npm install @magnolia/cli -g`
    - verify with `mgnl help`
2. get the default magnolia bundle that is used in the Docs, that is magnolia-community-webapp
    - so jumpstart with option nr. 2: `mgnl jumpstart`
    - start it up by`mgnl start`
    - visit: http://localhost:8080/magnoliaAuthor
        - Username: superuser
        - Password: superuser

Tip: to have local prototypes run: `mgnl customize-local-config`

Creating a light module, standard components/pages development

The linked tutorial describes well enough how they are to be created/used, a bit more compact overview:

- open the light-modules directory and run the following: `mgnl create-light-module bucket-list-module`
    - creates all the files needed for a page template definition called bucket-list based on the my-page-prototype variant that could be created /downloaded in the previous step.
    - Out-of-the-box prototypes are awesome, having many examples/possible scenarios with comments.
    - To create page/component, first `cd bucket-list-module`
        - `mgnl create-page bucket-list -P my-page-prototype`
            - open magnoliaAuthor / Add page / name: item 1; template: bucket-list
            - Enter the title Dive with sharks in the page properties dialog.
            - in order to add/remove content to a page we must have components available to that page
        - `mgnl create-component activity -P my-component-prototype`
            - make it available: `mgnl add-availability activity bucket-list@main`
            - for this to work, we have to have the FreeMarker expression [@cms.area name="main"/] inside the HTML body tag in our pages template script.
            - if it has only this one component, then when adding we get instantly the add "activity" component's dialog, otherwise the component's position after/before and type has to be set first
            - a component can also be added from Magnolia Templating Kit module
                - `mgnl add-availability mtk:components/image pages/bucket-list@main`
            - components from other light modules can be used via magnolia cli
                - note down the desired module to be installed from the list: `mgnl search map`
                - make sure to cd to light-modules dir and type: `mgnl install google-maps-magnolia`
                - cd back to bucket-list-module

                  ```
                  mgnl add-availability google-maps-magnolia:components/google-maps pages/bucket-list@main
                  ```

Cover some basic features of a website

Theme

Resource files can be set manually or dynamically, this later method is referred as a theme functionality in magnolia.

- A theme is a module or part of a module composed of CSS, JS, and images.
- A site definition of which a theme is part of contains site-wide configuration such as navigation, common page areas, and JavaScript that you want to load on every page. A site definition also contains a reference to your theme and defines locales for entering content in multiple languages.
- Themes can be configured:
  - In any module.
  - In YAML on the file system or in the JCR.
- The theme configuration must reside in the themes folder of the module in /<module-name>/themes/<theme-name> . This can be done in YAML or JCR. Ex.: travel-demo/themes/travel-demo-theme.yaml

example yaml:

```
cssFiles:
bootstrap:
addFingerPrint: true
link: /.resources/travel-demo-theme/libs/twitterbootstrap/css/bootstrap.min.css
media: all
roboto-condensed:
addFingerPrint: true
link: https://fonts.googleapis.com/css?family=Roboto+Condensed:400,300,700
media: all
jsFiles:
respimage:
addFingerPrint: true
link: /.resources/travel-demo-theme/libs/lazysizes/respimage.min.js
addtoany:
link: https://static.addtoany.com/menu/page.js
imaging:
# Variations are an effective alternative to resizing images with CSS:
class: info.magnolia.templating.imaging.VariationAwareImagingSupport
variations:
"1600":
class: info.magnolia.templating.imaging.variation.SimpleResizeVariation
width: 1600
"1366":
class: info.magnolia.templating.imaging.variation.SimpleResizeVariation
width: 1366
```

Example usage:

```
[#assign site =
sitefn.site()!]

[#assign theme =
sitefn.theme(site)!]

[#list theme.cssFiles as cssFile]
<link rel="stylesheet" href="${cssFile.link}" media="${cssFile.media}" />
[/#list]

[#list theme.jsFiles as jsFile]
<script src="${jsFile.link}"></script>
[/#list]
```

Prototype page

Use same content/page structure at multiple places

- create a site definition
- has to have a prototype.yaml in templates/pages
- can have component areas with editables turned off by default and inherited from parent page
- a theme specific for this page structure can also can be defined here

Navigation

The out-of-the-box, most straightforward way is to take advantage of the page hierarchy and use a macro to generate the navigation in the prototype page.

Some approaches to create navigation:

1. LinkList as basic navigation: `mgnl add-availability mtk:components/linkList pages/bucket-list@navigation`
2. Customize LinkList: grab the mtk template and modify it
   - Have space for limited menu items for ex., by adding 5 link list elements to the navigation I can create a manually editable menu columns with dropdowns.
   - Render them differently as needed in navigation section when the link list has single or more elements
   - Drawback: it is not straightforward to get if a menu element, a page is active and so the element should get an active class.

```
<nav>
<ul class="menu">
[@cms.area name="navigation"/]
</ul>
</nav>

areas:
navigation:
editable: false
availableComponents:
navList:
id: presentation:components/navList
inheritance:
components: all
enabled: true

areas:
navigation:
editable: true
```

3. Based on mtk navigation inside a template area

- Grab the hardcoded navigation.ftl macro and customize it

```
[#-- Basic navigation macro which generates simple navigation. You can adjust it to fulfil your needs. --]
[#macro navigation navParentItem depth=1 expandAll=false navClass="nav"]

[#if navParentItem?has_content &amp;&amp; depth > 0]
<ul class="menu">
[#assign navItem = navParentItem]
[#assign activeNavItem = navfn.isActive(content, navItem)] [#-- Active navigation item is the one which is same as current page--]
[#assign openNavItem = navfn.isOpen(content, navItem)] [#-- Open navigation item is the one which is ancestor of current page--]
[#if activeNavItem]
<li class="active">
[#elseif openNavItem]
<li class="open">
[#else]
<li>
[/#if]
<a href="${navfn.link(navItem)!"#"}">${navItem.navigationTitle!navItem.title!navItem.@name}</a>
[#if expandAll || activeNavItem || openNavItem]
[@navigation navItem depth-1 /]
[/#if]
</li>
[#assign navItems = navfn.navItems(navParentItem)]
[#list navItems as navItem]
[#assign activeNavItem = navfn.isActive(content, navItem)] [#-- Active navigation item is the one which is same as current page--]
[#assign openNavItem = navfn.isOpen(content, navItem)] [#-- Open navigation item is the one which is ancestor of current page--]
[#if activeNavItem]
<li class="active">
[#elseif openNavItem]
<li class="open">
[#else]
<li>
[/#if]
<a href="${navfn.link(navItem)!"#"}">${navItem.navigationTitle!navItem.title!navItem.@name}</a>
[#if expandAll || activeNavItem || openNavItem]
[@navigation navItem depth-1 /]
[/#if]
</li>
[/#list]
</ul>
[/#if]
[/#macro]
```

In a more advanced manner the dynamic (2. point, based on pages) and managed (3. manually editable from page edit) is available in public lightmodule components:

- https://www.npmjs.com/package/navigation-magnolia

Multivalue field

Useful when we should get a list of something when a component is added, like a list of images for a slider/carousel

Multilanguage

- /config/server/i18n/authoring/enabled: true
- Set i18n=true in components' dialog fields
- Define the languages that editors should be able to choose as locales
- <site>/i18n/locales, ex.:
    - en
        - country
        - enabled: true
        - language: en

Issue

I have a community edition, started with mgnl jumpstart, tried to follow these descriptions it like:
1. I did set/modify the my user profile to have a different than fallback language (hu - Magyar), but it didn't change the English language in the admincentral as presented on screenshots here.
2. I have added to the /server/i18n/system/languages two more languages (hu, ro).
3. /config/server/i18n/authoring is true, but its class in the figure is
info.magnolia.cms.i18n.DefaultI18nContentSupport
but bellow, the recommended class entry for one site is:
info.magnolia.ui.framework.i18n.DefaultI18NAuthoringSupport
So, which one should it be?
4. In component/page dialogs have set i18n=true in fields
5. Define locales in site, set it, also changed the type of enabled field from the default String to be Boolean.
6. Checking the top bar in the page editor, no dropdown has appeared yet.

Forms

- from the frontend perspective it has a static design you have to go with.
- first configure email sending by SMTP config
- Use it by:

  ```
  availableComponents:
        form:
          id: form:components/form
  ```

- To customize the look try copy/manipulate that form component, it seems to be a cumbersome thing to do.
- To have more control use Spring's Form tag library (advanced topic).

Site search

Issue

The configuration seems to be tied to xml files available in the maven bundle of Magnolia, plain mgnl-cli frontend-lightmodule way of configuration is not supported.
Have to go either the Jackrabbit or Solr way, the Jackrabbit is the default, the Solr handles better a higher volume of assets indexing.

Related articles

- Notes from my first light development