Light modules

Related topics:

- Module structure
- Light development in Magnolia

A Magnolia light module is a file and folder-based module. It can define every Magnolia item which can be configured in YAML – such as apps, templates, dialogs, themes, and others. It can also contain a lightweight YAML-based module descriptor and web resources such as css and javascript files.

Light modules can be built with a simple text editor, with no special tools (such as Maven) required. A light module cannot incorporate java classes.

- Light module structure
- Light modules folder: magnolia.resources.dir
- What items can be defined in a light module?
- Creating a light module with Magnolia CLI

Light module structure

Example of light module structure:

```makefile
magnolia.resources.dir/
<module-name>/
    apps/
    dialogs/
        myDialog.yaml
    module.yaml
    webresources/
    templates/
        components/
            myComponent.ftl
            myComponent.yaml
        pages/
            myPage.ftl
            myPage.js
            myPage.yaml
```

Light modules can contain more or fewer folders compared to this example.

Light modules folder: magnolia.resources.dir

Magnolia light modules must reside within the Magnolia resources directory; this directory is also often referred to as the Magnolia light modules folder. The name and the location of this directory are arbitrary, but you must provide the path to the folder via configuration.

`magnolia.resources.dir` is a property defining the directory from which resources are loaded in a Magnolia instance. This directory is used for file-based resources such as light modules and for overriding classpath resources. The property is configured in WEB-INF/config/default/magnolia.properties and has the default value `$,magnolia.home/modules`. To see the current value of the property, go to the Config Info tab in the About Magnolia app.

You can use symbolic links (a.k.a symlinks or soft links) in the resources directory to include light modules located elsewhere on your system.

⚠️ Magnolia 5.6.1+: A change in the light modules folder may also flush the server cache. You can configure this behavior under `/modules/cache/config/contentCaching/defaultPageCache/flushPolicy/policies/lightModule/pathToCacheMappings/defaultPageCache` with the `pattern` property, which specifies the pattern for the operation. The default value is `.*(templates|webresources|i18n)/.*`.

What items can be defined in a light module?

The following items can be defined within a light module using YAML. Items defined by YAML can be decorated, or can be reused with YAML inherit and include.

- App definitions
- Dialog definitions
- Field type definitions
Media editor definitions
Message view definitions
Renderer definitions
REST endpoint definitions
Template definitions
Theme definitions
Virtual URI mappings
Workflow definitions
Work item handler definitions

Other non YAML-based items which can be in a light module:
- i18n message bundles (see Internationalization and related pages)
- JavaScript Models

Creating a light module with Magnolia CLI

Using the Magnolia CLI you can create the folder structure for a light module with the command create-light-module. Open a shell, cd to your light modules directory, and execute the following command:

```
mgnl create-light-module one-pager-module
```

When creating a name for a module, do not use spaces, accented characters such as é, à, ç, ö, ü or special characters (e.g. slashes /, \ and so on). The name must match the regular expression `[a-zA-Z0-9-_]`.

Note that this only works if you have installed Magnolia CLI, see Magnolia CLI Installation.