Right-To-Left support in Admincentral

[1 How far should RTL support span?] [2 Proposals] [3 Technical study] [3.1 1. RTL support across the web stack] [3.2 2. RTL support in CKEditor] [3.3 3. "Bidifying" UI components] [4 Resources on the topic from the web]

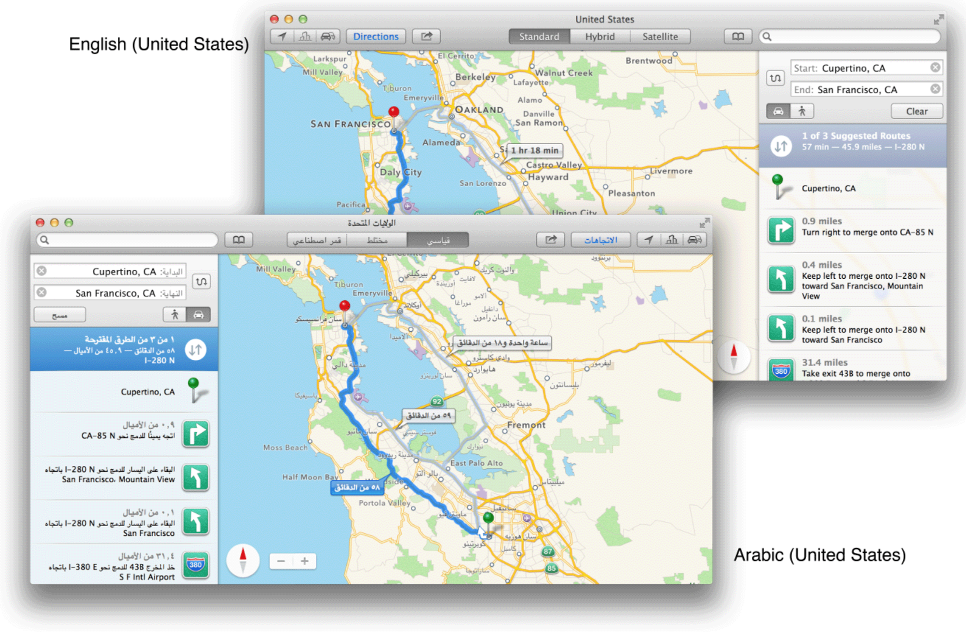
How far should RTL support span?

In a **good citizen application**, RTL support typically involves the following:

1. Direction of static texts, paragraphs, with right alignment as default
    - latin chunks of text are left in LTR direction, this is called *bidirectional text*
2. Direction of input elements (similarly)
3. "Reversal" of the UI horizontal flow
    - general layout, e.g. content/article on the right, meta on the left
    - column ordering in tables
    - other UI components which have a left-aligned horizontal flow, such as tabs

Here's a nice example from Apple developer's guide (actually the only nice example I found)

English (United States)



Arabic (United States)

For the **Magnolia 5 Admincentral**, RTL support translates as follows (non-exhaustive list):

- Direction of local static texts
    - Actions in action bar
    - Headers of dialogs, alerts
- Forms / dialogs
    - Direction of text fields in forms, dialogs, detail subapps
    - CKEditor
    - Layout (fields labels vs. actual field)
- Column ordering in workbench trees, lists
- Potential layout reversals
    - Content app (reverse workbench vs action bar)
    - App launcher
- ...

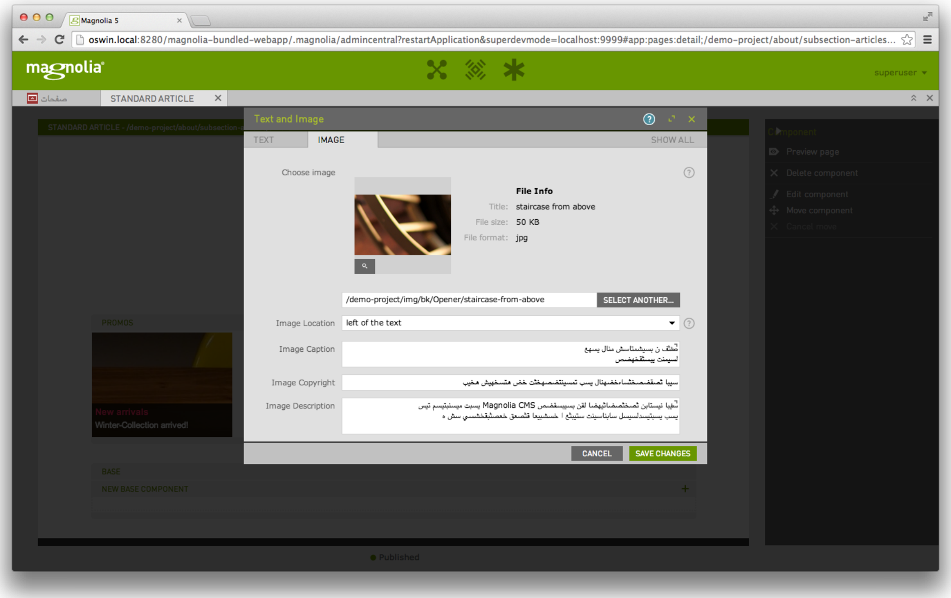
Proposals

Block A: RTL Authoring

- [illegible] property though

Estimate: *2w*

Below is the outcome of "automatic" text direction applied to basic text fields.
Note that latin texts stick to the left, and that this does not offer form i18nization, i.e. captions are not translated and layout orientation stays the same.



Choose image

File Info

Title:	staircase from above
File size:	50 KB
File format:	jpg

/demo-project/img/bk/Opener/staircase-from-above

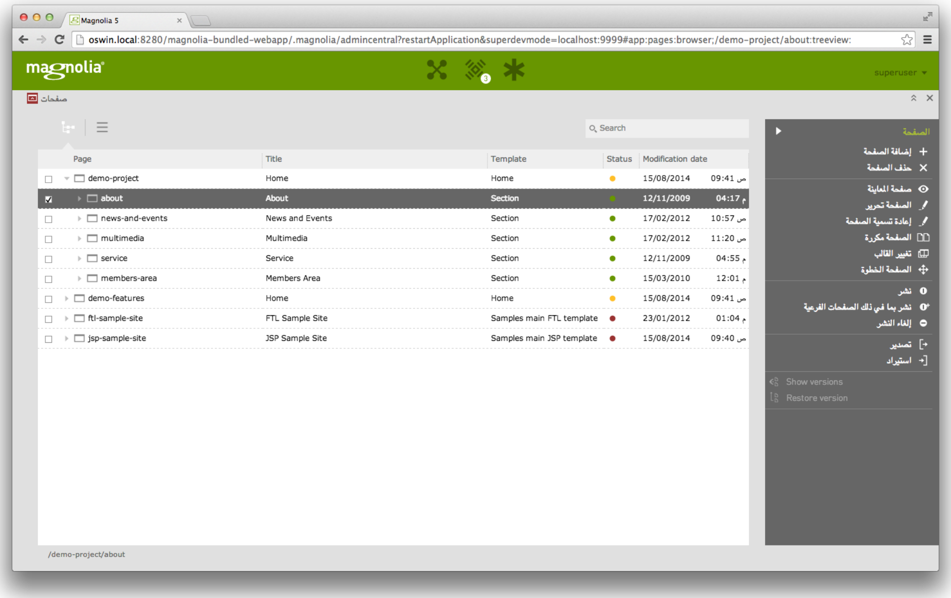
Image Location	left of the text
Image Caption	طثف ن بسيشمتاسش منال يسهع لسيمنت يبستقخهضص
Image Copyright	سيبا ثصقضصخشاءخضهنال يسب تمسينتضصهخث خض هتسخهيش هخيب
Image Description	سيبا نيستابن ثصخثصضاثيهضا لقن بسيبسقضص Magnolia CMS يسبت ميسنبتيسم تيس يسب يسبتيسدلسيسل سابناسينت ستييثع ا خسشبيعا قثصعق خعصشبقخشسي سش ه

Block B: Support for RTL i18nization

- identify critical UI components
- might include one most crucial layout reversal
- ...

Estimate: *4w*

Below is the result of a focused experiment on the action bar *(google translations — do not share blindly!)*



Page	Title	Template	Status	Modification date
demo-project	Home	Home	🟡	15/08/2014 09:41 ص
about	About	Section	🟢	12/11/2009 04:17 م
news-and-events	News and Events	Section	🟢	17/02/2012 10:57 ص
multimedia	Multimedia	Section	🟢	17/02/2012 11:20 ص
service	Service	Section	🟢	12/11/2009 04:55 م
members-area	Members Area	Section	🟢	15/03/2010 12:01 م
demo-features	Home	Home	🟡	15/08/2014 09:41 ص
ftl-sample-site	FTL Sample Site	Samples main FTL template	🔴	23/01/2012 01:04 م
jsp-sample-site	JSP Sample Site	Samples main JSP template	🔴	15/08/2014 09:40 ص

What is not considered right now

- Reverse ordering of table colums
- General layout "reversals" (e.g. content apps)

Technical study

1. RTL support across the web stack

Short version: We favor the document-based "dir=auto" attribute wherever possible; works best combined with CSS "text-align: start".

There are two known ways to make input elements display and enter text in RTL fashion: by means of CSS rules and by means of the **dir** attribute of the elements.

CSS "direction"

RTL input CSS

```
input {
  text-align: start !important;
  direction: rtl;
}
```

The **`text-align`** rule should be used in order to override the left-alignment set by Vaadin by default. The **`direction`** rule actually sets the direction of the content flow.

Pros of the CSS approach:

1. Easy to apply: by simply adding a corresponding style name to an arbitrary component it is possible force the inputs to use RTL.
2. As a consequence of 1. - rather easy to implement the basic support.
   a. The style could be conditionally applied to the field component within the `FormBuilder`.
   b. The decision whether the style could come from the definitions.
3. Seems to be supported by all the modern browsers (even IE6+)
4. Cascades well.

Cons of the CSS approach:

1. No way to implement the automatic resolution of RTL/LTR cases:
   a. The fields that should use RTL have to be manually marked as such (additional definition properties are needed).

HTML5 "dir" attribute

Dir attribute

```
<input type="text" dir="auto"|"rtl">
```

Pros of the HTML approach:

1. Has a magic **auto** value
   - rather smartly guesses how to treat an input (based on the first char of the text)
   - setting it properly requires no additional definition properties
2. At least according to the linked article - more robust then CSS.
3. Cascades well (but not so well for the case of **auto**).

Drawbacks:

1. Harder to apply from server-side
   a. Special extension required. It would traverse the component DOM structure and set the attribute where needed.
   b. Some widgets will need dedicated tweeking

2. RTL support in CKEditor

With CKEditor is seems to be rather straightforward to control the text/layout direction both in the toolbar and in the text editor.

In order to tweak the direction of the CKEditor toolbar layout the **cke_rtl & cke_ltr** style names should be applied to the `info.magnolia.ui.vaadin.richtext.MagnoliaRichTextField`.

The editor text direction is controlled by the **contentsLangDirection {*value options: rlt | ltr*}** property. It can be applied the following way:

Setting CKEditor text direction to Right-to-Left

```
config.addExtraConfig("'contentsLangDirection'", "'rtl'");
```

3. "Bidifying" UI components

The extension way

Vaadin extension proves to be a neat way of adding the `dir` attribute throughout multiple components (e.g. `TextField`, `TextArea`), without extending each and every client-side widget.

- Worked like a charm for e.g. (server-side) `TextField`
  - its client-side widget counterpart is directly the HTML `input` where we want to set the attribute
- However, other widgets may be "wrapped" (e.g. in a `div`)
  - The extension *should* most likely also drill down the DOM structure for text-inputs where the attribute should be applied
- The extension *could* also be smarter and handle the RTL settings for the CKEditor

Impact on Vaadin components and GWT widgets

Another focused experiment was carried out, on the **RTL i18nization** of the action bar.

- There, the client-side DOM hierarchy is more complex than just inputs
  - Sections, groups, actions, are not mirrored by Vaadin components on the server
  - Not *as easy* to set up a simple extension approach (we have to target specific nested DOM elements of the action bar)
- Enabling RTL for the action bar required to implement it all the way through:
  - Configuring RTL first from the `ActionbarPresenter` (Magnolia)
  - propagating it to the `ActionbarView` interface and implementation (Magnolia)
  - `Actionbar` raw Vaadin component, including:
    - shared state
    - client-side connector
  - `ActionbarWidgetView` and impl (GWT), plus:
    - `ActionbarSectionWidget` for sections title
    - `ActionbarItemWidget` for actions
  - Admittedly the action bar is one example where the client-side complexity is rather high
    - but that would definitely be an effort to do it that way for most of our common widgets
- We *could* use an extension too for such i18inzation (has sort of a different purpose from authoring)

Fetching text orientation from the locale

It is rather straight forward to "derive" text orientation from the current locale:

```
Locale userLocale = MgnlContext.getLocale();
ComponentOrientation textOrientation = ComponentOrientation.getOrientation(userLocale);
if (textOrientation.isHorizontal() &amp;&amp; !textOrientation.isLeftToRight()) {
    view.setBidi(true);
}
```

The code

Code for these couple experiments (extension, action bar) lives on the following branch: https://git.magnolia-cms.com/users/mgeljic/repos/ui/commits?until=refs%2Fheads%2Ffeature%2Frtl-mge

Corresponding JIRA epic: BL-303 - Getting issue details... STATUS

Resources on the topic from the web

- http://css-tricks.com/almanac/properties/d/direction/ - good summary/tutorial on how/when to use CSS and attribute solutions.
- W3C spec pages:
  - http://dev.w3.org/csswg/css-writing-modes/#direction - CSS property spec
  - http://www.w3.org/International/articles/inline-bidi-markup/ - Bi-directional mark-up explained
  - http://www.w3.org/International/questions/qa-html-dir - Structural Mark-up and Right-to-Left text in HTML
- http://www.i18nguy.com/markup/right-to-left.html
- UI guidelines
  - https://developer.apple.com/library/prerelease/ios/documentation/MacOSX/Conceptual/BPInternational/SupportingRight-To-LeftLanguages/SupportingRight-To-LeftLanguages.html