Frontpage User Guide

S. Hamblett

Table of Contents

1. Introduction

The Frontpage component is a front end content editor and creator. It's intended audience is web users who have their own page set for blogs, news articles and the like and want to manage their own content without having to log in through the manager interface to do this.

The component itself is an amalgamation of Evolution's QM extra from where we get the jQuery based toolbar and colorbox interface and the Evolution Pubkit package which allows content creation through the front end. Unlike Evolutions QM however this component stays entirely in the front end, there is no connection to the manager.

Content editing is provided via the jQuery TinyMCE plugin or via the Aloha HTML5 editor which can be toggled on or of to allow raw HTML content editing if needed. Other resource fields are supported such as title, summary, alias etc. along with check boxes for publish, hide from menus and folder on the create page.

Permissions are based entirely on Revolutions permissions system, so logged in web users must have the correct edit/create permissions or the Frontpage toolbar will not appear. An additional permissions check is also supported whereby users must have certain nominated roles such as 'Editor' for instance to be allowed access to the toolbar. This supports the concept of content managers who can be a variant of the site administration user with access to the whole site. This functionality can be toggled on or off.

2. Installation

The Frontpage package is installed as any other, through package manager.

On installation the following additions to your site will appear :-

1. A container in the document tree named Frontpage, under this you will have three resources named Frontpage Create, Frontpage Edit and Frontpage AJAX. These are the edit, create and AJAX resources respectively, also present are three resources named as above appended with 'Aloha' for use by the integrated Aloha editor.
2. A template named Frontpage. This is the template used on the pages above and pulls in the interface to jQuery, the TinyMCE plugin and the form css file. A template named Frontpage Aloha. This is the template used to integrate the Aloha editor.
3. Two chunks named frontpageEdit and frontpageCreate. These contain the form layouts for the edit and create pages. Two chunks named as above appended with 'Aloha' for use by the integrated Aloha editor. Two further chunks are are also installed for Aloha editor integration.
4. Three snippets named frontpageEdit, frontpageCreate and frontpageAJAX. These contain the edit and create logic for the edit and create pages and the AJAX link logic. Three further snippets named as above appended with 'Aloha' for use by the integrated Aloha editor.
5. A plugin named Frontpage. This is attached to the OnWebPagePrerender event and draws the Frontpage toolbar for logged in web users. This plugin has a number of customizable properties, see the Customization section below
6. Four system settings, three that hold the resource id's of the edit, create and ajax pages and one that holds the template to use on content creation. A further three settings are supplied with the string 'aloha' appended to the key name for use by the integrated Aloha editor. See the Customization section below for more details.

All the above elements are resident under the newly created Frontpage category.

3. Usage

On installation you will see no changes to your site until a web user logs in through the front end. When this happens the plugin will check what permissions the web user has in relation to content creation and editing.

This is based on normal Revolution permissions for that web user, set by document/user group membership, access policies etc. Depending on the outcome of these permissions checks either no toolbar will be shown, a toolbar with the Edit button will be shown, a toolbar with the Create button will be shown or a toolbar with both these buttons will be shown. There is another role based permissions check that can be invoked over and above this, as well as global Create/Edit button access, see the Customization section below.

From here, if the user elects to edit the current document or create a new one by pressing the Edit or Create buttons an expanding colorbox overlay will appear containing the Edit or Create page. These pages are further described below :-

Edit

The edit page invokes the frontpageEdit resource which displays a form with the following fields :-

1. Title
2. Long title
3. Description
4. Alias
5. Summary
6. Menu Title
7. Menu Index
8. Hide From Menus checkbox
9. Publish checkbox

Following these fields is the main content area itself. The content area is a textarea field linked to the TinyMCE jQuery plugin. The TinyMCE editor can be toggled on or off as needed by pressing the 'Add/Remove editor' link underneath the content area. Several toolbars and plugins are are supported by this editor, customization of these additions and the setting of paths to user areas etc. is not supported in this release, see the Further Development section for more details on this.

The content area contains the document content field which can be edited as normal.

Because of the way Revolution works any content that contains tags, i.e. '[[' will be shown with these tags separated into this '[['. This stops Revolution parsing these and displaying their real output, we want to see this in the content '[[Wayfinder? &start=`0`]] not the output of the actual call which would occur if we left the tags intact. Editors can ignore this, any expanded tags found are closed again when the document is saved so editors can type in normal tag syntax, they do not have to separate the tags themselves.

At the bottom of the form is a Save button. When pressed this saves the edited document content and refreshes the page view so that editing can continue if needed. When this button is pressed your changes are saved to the database and the MODX cache is cleared.

Next to this is a Complete button. When pressed this closes the colorbox overlay and refreshes the underlying page allowing your previously saved edits to be seen. Note that this button does NOT save the content before closing the colorbox overlay, the Save button must be used to do this.

The large X symbol in the right bottom corner of the box also performs the complete action.

Because we are editing in the front end please beware of the publish button, if you set this to 'not published' and save the document you won't be able to see this document again until you re-publish it through the manager interface.

Create

The create page contains the same fields as the edit page above with the addition of a 'Folder ' checkbox to indicate if the document is to be a container and a ' Create as Child of' text box to allow the parent of the document to be specified.

Creation follows the 'create here' principle, so your created document will be initially created in the document hierarchy as a peer of the underlying page. The processing establishes whether the underlying page has a parent, i.e. is not a top level document. If this is the case then the template used by the parent is used as the template for the newly created document or depending on the 'default_template' system setting any nominated template. Also the created document is placed in the same document groups as its parent. So the created page would be allocated to the users page set automatically if you had set your users up this way.

If the created document has no parent then its template is set to the value in the 'default_template' system setting or 0 if this indicates that the parent should be used. No resource groups are set in this case.

The created documents parent can be changed any time after this initial setting by entering the identifier or alias of the parent document in the 'Create as Child of' entry box, if this differs from what is currently set by the above processing the document and its resource groups are updated to reflect this. The template to use is determined as above.

The page operates in the same way as the edit page with the exception that the document create operation occurs when you press the Create button, not on Save. This means that if you press Create and immediately press the complete button you will create an empty document.

The Complete button operates in the same manner as the edit page with the exception except that the browser is re-directed to the newly created page rather than the underlying page allowing you to re-edit the page as you wish.

The above mechanism is installed by default, also selectable is the Aloha editor interface, see the section named 'Aloha Integration' for further details.

4. Customization

Customization of the component is achieved by a combination of plugin properties and system settings. Taking each in turn :-

Properties

The plugin has the following customizable properties :-

1. loadJQuery. This property allows the loading of jQuery into your front end pages should you need it, i.e. you are not using jQuery. Default is 'no', you shouldn't need this as the colorbox overlay uses its own JavaScript file.
2. JqueryNoConflict. Sets jQuery into no conflict mode if you need to load it, default is 'no'.
3. ShowCreate. This property is a global setting to show or hide the Create button regardless of any underlying revolution permissions. Default is 'yes'.
4. ShowEdit. This property is a global setting to show or hide the Edit button regardless of any underlying revolution permissions. Default is 'yes'.
5. AutohideToolbar. This property allows the toolbar to be permanently shown or not. Default is 'yes' so the toolbar is auto hidden.
6. PerformRoleCheck. Activates or deactivates role based permissions checks, see below. The default is 'not activated'.
7. ContentManagerRoles. This property contains a comma separated list of roles for the role based permissions checks above. Default is empty.
8. 'boxWidth. This property sets the width of the colorbox overlay. Default is '90%'.
9. boxHeight. This property sets the height of the colorbox overlay. Default is '90%'.
10. editMethod. Either 'classic'(default) or 'aloha' to use Aloha editor integration(see below).
11. ActiveAloha. True or false to activate the Aloha editor 'raw' page mod(see below).
12. jQueryPath. This property sets the path to use if you wish to load your own jQuery. Default is https://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js

The role based permissions check allows for only users with nominated roles to access the toolbar. This can be used to allow general site wide editing from the front end whilst locking this down to certain users only.

For instance you may wish to create an administration user that can edit any site content, you assign the role 'Editor' to this user and set other permissions and access policies as appropriate. Then activate the PerformRoleCheck property above and set the ContentManagerRoles field to 'Editor'

When logging in through the front end the users role is checked, if the user is a nominated content manager then the processing proceeds to check normal Revolution

permissions for page access, if not then no toolbar will be displayed.

Note because this check is done before normal permissions checks if you activate this all your web users must also have the editor role. Super Users are not affected by this check.

System settings

The component has the following system settings under the namespace 'frontpage' :-

1. edit_resource. This contains the resource id of the edit page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
2. create_resource. This contains the resource id of the create page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
3. ajax_resource. This contains the resource id of the ajax page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
4. edit_resource-aloha. This contains the resource id of the Aloha editor integration edit page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
5. create_resource-aloha. This contains the resource id of the Aloha editor integration create page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
6. ajax_resource-aloha. This contains the resource id of the Aloha editor integration ajax page. If you need to change the resource id's of your Frontpage pages for any reason then this setting must be updated with the new resource id.
7. default_template. This field dictates how templates are assigned to newly created resources. If it is set to the value 'parent' newly created resource templates will be set the value of the parent resource if one exists, otherwise 0. If it contains a value then this value will be used.

Deeper customization of the pages can be carried out if needed by editing the frontpageEdit and frontpageCreate chunks. And their respective Aloha counterparts. These chunks contain the form layout HTML for the edit and create pages with CSS layout achieved through the 'forms.css' file and its Aloha counterpart.

Obviously only the placeholders present in these chunks are supported by the back end code so adding fields would entail editing the correspondent edit and create snippets but you could use these to remove fields for instance, or make fields read only, publish may be a god candidate here.

5. Aloha Editor Integration

The Aloha editor is now integrated into Frontpage. Aloha is an advanced browser HTML5 based WYSIWYG editor. Aloha is an AJAX based editor thus removing the need for the traditional form/submit edit method used by Frontpage. Two methods of integration are supported, an edit method selection of 'aloha' and a 'raw' page editing method, both these options are detailed below.

Aloha Edit Method selection

This option allows editing of content using the Aloha editor based method and is selectable by setting the editMethod property on the Frontpage plugin to 'aloha'.

When edit or create are now invoked on a document the colorbox opens as normal and the documents fields are presented as normal, operation utilises the same mechanisms outlined above except no save or complete buttons are present.

When the user clicks in one of the fields the Aloha editor 'floating box' appears and the user can type text into the fields selected. As soon as the user moves out of focus on the selected field the update is saved by an AJAX call. The colorbox can be closed as normal by pressing the 'X' in the bottom right hand corner, all updates applied to the document should now be visible. Each field can be edited independently of others using this method.

It should be noted that the create page doesn't support re-parenting in the Aloha edit method, nor do is redirect to the newly created page on closure, these deficiencies will be addressed in a later release.

Aloha Page editing selection

This option allows editing of resources in situ, i.e without opening a colorbox, the user just edits selected fields of the document and sees the changes happen in real time as they type, once again saving updates is done when the user clicks out of a highlighted field using AJAX calls. To activate this option set the ActiveAloha property on the Frontpage plugin to true.

Obviously the Aloha editor has no concept of your page layout, nor how this corresponds to the MODX document fields and the database backend so to allow these connections to be made your page must be marked up to allow this. This is simply achieved by the addition of an enclosing div that instructs Aloha what to do with the page content.

For instance, most MODX templates contain this structure :-

[[*content]]

to encapsulate the documents content. To allow Frontpage to activate Aloha on this

content the following markup must be added :-

** **

[[*content]]

** **

The additional div tells Frontpage that this section of the document is editable by Aloha using the class=”alohaeditable” construct and which field of the MODX document hierarchy the edits should be saved to using the id=”alohaContent” construct. When the user now clicks the page content area it is highlighted in a yellow box subsequent edits being saved as content updates. Simple really.

The table below shows the mapping of the div id's to MODX document fields :-

alohaTitle	Title
alohaLongTitle	Long Title
alohaDescription	Description
alohaAlias	Alias
alohaSummary	Summary
alohaMenuTitle	MenuTitle
alohaContent	Resource Content

The user will not see these elements highlighted until they click in the appropriate area of the on screen document, how you present these fields to the user is of course up to you. Edits are saved in real time when the user leaves the fields focus.

This method allows users to edit content using their pre-existing page styles a real WYSIWIG experience.

Considerations using Aloha

Because the updates are performed via AJAX calls there's no real going back here, when the user leaves focus the update occurs, some users may not want this, the 'classic' edit method of Frontpage allows corrections to be made before the form as a whole is submitted. I intend to implement a simple undo/redo stack to alleviate this in future releases.

This is the first release with Aloha integrated so I wouldn't rush this in front of clients just yet until the inevitable tranche of bugs has been ironed out, also its worth noting that the formatting options available are limited to what the Aloha guys provide, not what MODX can do. At the moment this is fairly rudimentary and may not be enough for a lot of users, its nowhere near as comprehensive as TinyMCE for instance. Hopefully the Aloha guys will address this as we progress.

Please feel free to discuss any of this on the MODX forums and/or raise issues on Frontpage in github.

6. Further development

This is the fourth release of this component and although functional there is still further development needed.

Other than that any other further development will be dictated by community feedback as the component is deployed more widely.