Jmbo Foundry Documentation

Release 2.0.6

Praekelt

March 29, 2016

Contents

Jmbo Foundry ties together the various Jmbo products enabling you to rapidly build multilingual web and mobi sites with the minimum amount of code and customization.

This documentation covers version 2.0.6 of **|jmbo-foundry|**.

Note: **|jmbo-foundry|** 2.0.6 requires Django 1.6. More on *Versions and Compatibility*.

CHAPTER 1

Overview

Jmbo Foundry ties together the various Jmbo products enabling you to rapidly build multilingual web and mobi sites with the minimum amount of code and customization.

Jmbo Foundry strives for a high level of through the web configuration. Much of the site's behaviour is configurable through the admin interface.

CHAPTER 2

Installation

Jmbo Foundry itself is just a Django product and can be installed as such. An easier approach is to follow http://jmbo-skeleton.readthedocs.org/ to get a fully working environment.

CHAPTER 3

Internals

3.1 Sites

Your web presence typically consists of a normal web site and a mobile site. There may be many more types of sites in future and Jmbo Foundry makes it easy to configure them independently. If your main site is served on *www.mysite.com* then go to *Sites* in the admin interface and set *Domain name* and *Display name* accordingly. Then add a site entry for your mobile site and set the values to *m.mysite.com*.

If you have only one site then you may blindly publish everything that is publishable to this site. However, if you have more than one site and in different languages then understanding sites become significant.

At its most basic level publishing to a site means making content appear on a site. This is easy to understand when the content is eg. an article, but content is not always limited to things which are easily translatable into real world objects.

3.2 Preferences

Preferences can be published to certain sites.

3.2.1 General preferences

Check *Private site* to make the site accessible only to visitors who are logged in.

Check *Show age gateway* to enable the age gateway for the site. Visitors must confirm their age before they are allowed to browse the site.

Exempted URLs are URLs which must always be visible regardless of *Private site* or *Age gateway* settings. Certain URLs like */login* are visible by default and do not need to be listed.

The *Analytics tags* field may contain javascript. There is a fallback to enable analytics on low-end browsers but it is not configurable through the web. See the *settings.py* section.

3.2.2 Registration preferences

You can select which fields to display on the registration form. Some fields (eg. *username*) are always visible on the registration form and cannot be removed.

You can select a subset of the displayed fields to be required. Fields which are absolutely required (eg. *username*) cannot be set to be optional. For instance, if the site users my log in using their mobile number then set *mobile_number* as a required field.

Some fields may need to be unique, especially those that may be used to log in to the site. Using the mobile number example above you should set *mobile_number* to be a unique field. It is important to decide beforehand which fields must be unique since it is difficult to remove duplicates if you change this setting. An exeption is raised if you attempt to change this setting and duplicates are detected (friendlier validation still to be added).

3.2.3 Login preferences

Users typically log in to a normal site with their username or email address, whereas a mobile number is a natural login field for a mobile site. Choose from *Username only*, *Email address only*, *Mobile number only* or *Username or email address*.

3.2.4 Password reset preferences

When a user loses his password he may request a password reset. Normally this is accomplished by sending an email to the user, but in the case of a mobile site it is desirable to send a text. Choose between *Email address* or *Mobile number*. Note that a password reset request does not automatically generate a new password for the user since this may lead to malicious people disabling users' accounts.

3.2.5 Naughty word preferences

You can set a list of weighted words. The *report_naughty_words* management command identifies potentially offensive comments. An email containing clickable links for approval or deletion is sent to the *Email recipients*.

3.3 Listings

A *listing* is essentially a stored search that can be rendered in a certain style. A listing can be published to certain sites.

Content type, *Category* and *Content* are criteria which define the items present in the listing. These criteria are optional and logically OR-ed.

Count specifies the maximum number of items in the listing.

Style is the default way in which the listing is rendered. The default styles are vertical, vertical, vertical thumbnail, horizontal, promo and widget. See *Listing styles* for detail.

Items per page is the number of items to display on a single listing page.

3.3.1 Listing styles

Vertical is a vertical listing with no images.

Vertical thumbnail is a vertical listing with images.

Horizontal is a side-by-side listing with images. Each item looks like a baseball trading card.

Promo collates the items in a slideshow.

Widget is the most complex. It is used when each item can be interactive, eg. a listing of polls. Polls you have already voted on are read-only, and the others may change content once you vote on them. The content type being represented as a widget needs to provide code for this functionality.

3.3.2 Implementation

A listing iterates over a set of items and offloads the rendering of each item. This is easier to understand looking at the `Horizontal` style.

A snippet from `templates/basic/foundry/inclusion_tags/listing_horizontal.html`:

```
{% for object in object_list %}
    <div class="item {% if forloop.first %}first{% endif %} {% if forloop.last %}last{% endif %} iter
        {% render_object object.as_leaf_class "list_item_ipod" %}
        <div class="clear"></div>
    </div>
{% endfor %}
```

Note how the template only cares about the layout of the items. Actual rendering of each item is offloaded to `{% render_object object.as_leaf_class "list_item_ipod" %}`. The logic behind `render_object` is fully documented in Jmbo, but in summary the naming convention is `templates/basic/{{ app_label }}/inclusion_tags/{{ model_name }}_list_item_ipod.html`. If you don't have a specific template for a model then it falls back to `templates/basic/jmbo/inclusion_tags/modelbase_list_item_ipod.html`.

Why the seemingly strange name "ipod"? Because the template needs to describe what it looks like. We try to use relatable names.

The convention provides enough flexibility to combine different content types in the same listing and have each item decide how to render itself.

3.3.3 Custom listings

Jmfo Foundry provides many standard listings but you may need to create your own listing. Create `templatetags/listing_styles.py` in your product:

```
from foundry.templatetags.listing_styles import AbstractBaseStyle


class MyListing(AbstractBaseStyle):
    template_name = "myproduct/inclusion_tags/listing_mylisting.html"
```

The listing style is autodetected and can be used in the admin interface and templates. Naming your listing is the hardest part!

3.3.4 Template tags

Render a listing directly in a template:

```
{% listing "my-listing-slug" %}
```

Render a listing on the fly:

```
{% listing queryset style="Horizontal" title="Foo" %}
```

Changing an existing listing's style is a bit more involved:

```
{% get_listing_queryset "my-listing-slug" as "qs" %}
{% listing qs style="Vertical" title="Foo" %}
```

3.4 Menus

A menu is essentially the same as a navigation bar, except it has a vertical layout by default.

A menu with slug *main* is considered special. It is assumed to be the site menu by default.

3.5 Navbars

A navigation bar typically contains a small amount of items since horizontal space is limited. Each item in the navigation bar is represented as a *Link*. A navbar can be published to certain sites.

A navbar with slug *main* is considered special. It is assumed to be the site navbar by default.

3.6 Pages

Page builder documentation tbc.

3.7 Layers

Jmbo Foundry makes use of https://pypi.python.org/pypi/django-layers-hr. It makes it possible to serve a set of templates and static resources as defined in *settings.py*. This means you can serve different HTML, Javascript and CSS to eg. basic mobile devices, smart phones and desktop browsers. These template sets (aka layers) also stack, so if you create foo.html for basic devices it is automatically available for desktop browsers as well. You can override foo.html for desktop browsers.

CHAPTER 4

Help

4.1 Changelog

4.1.1 next

1. Remove CSRF protection from search form since searching is always a readonly operation.

4.1.2 2.0.3

1. Work around deprecated PickleSerializer when setting session expiry.
2. Defensive code in member detail view.

4.1.3 2.0.2

1. Do not attempt to call photologue getters in template if image is not set.
2. Page editor now uses jQuery 1.10.2 and jQuery UI 1.10.4.

4.1.4 2.0.1

1. Change position of api in urls.py so resource registration works properly.

4.1.5 2.0.0

1. Simplify API to use primary keys.
2. Depend on stable versions of Jmbo products.

4.1.6 2.0.0a5

1. Remove references to atlas in migrations.

4.1.7 2.0.0a4

1. Fix sitemap urls.

4.1.8 2.0.0a3

1. Listing fields content and pinned now use a through manager making ordering possible.

4.1.9 2.0.0a2

1. Fix case where *resolve()* would fail if site is run from a subpath.

4.1.10 2.0.0a1

1. Move to Django 1.6 support. Backwards incompatible.
2. Use *django-layers-hr* to handle layering. The FOUNDRY['layers'] setting is now deprecated.
3. Deprecate legacy handling for substring *_LAYER_* in photosize name.
4. Add a *ViewProxy* model enabling views to appear in listings.

4.1.11 1.3.0

1. Deprecate *compute_settings* function.
2. Ignore result of celery tasks as appropriate.
3. Up *jmbo* requirement to 1.2.0.
4. Up *jmbo-post* requirement to 0.4.
5. SEO improvements in templates.

4.1.12 1.2.6.1

1. Delegate photologue dependency to Jmbo.

4.1.13 1.2.6

1. Fix naughty word task emoji handling.

4.1.14 1.2.5.1

1. Fix typo leading to unassigned variable.

4.1.15 1.2.5

1. RSS2 feed now includes canonical image.

4.1.16 1.2.4

1. Validate member profile image strictly.
2. Allow = in username.
3. Friendly error message when attempting to use the same slug for overlapping sites.

4.1.17 1.2.3.1

1. Use new version of *django-ckeditor* with prettier toolbars.
2. Make ajax pagination more robust. It now always targets the correct listing.
3. Fix password setting on member change form.

4.1.18 1.2.2.3

1. Hotfix - fix missing import.

4.1.19 1.2.2.2

1. Hotfix - image layer fallback functionality restored.

4.1.20 1.2.2.1

1. Hotfix - added dependency link to photologue.

4.1.21 1.2.2

1. Fix forms.css rule for required fields.
2. Adapt monkey patch because of *django-photologue* version 2.8.praekelt.
3. Make it possible to define custom listings.

4.1.22 1.2.1

1. Use *next* parameter when redirecting to age gateway. On successfully passing the age gateway, the user is redirected to *next*.
2. Allow a partner site to automatically pass the age gateway for a user by providing age gateway data in a JWT token.

4.1.23 1.2

1. Move to jQuery 1.10.2 as recommended version. If you have customized and static Javascript resources you will have to update them manually.
2. Use a newer version of AnythingSlider.
3. Allow form class to be passed to join view.

4. Cache individual comments on comment list.
5. Minor performance improvements.

4.1.24 1.1.23

1. Fix template error in *modelbase_list_item_ipod.html*.

4.1.25 1.1.22

1. Fix bug where it was possible for an event handler to change the default avatar during user registration.
2. Cache individual listing item templates.

4.1.26 1.1.21

1. Content type, categories and tags fields on listings are now ANDed when evaluating the listing.
2. Do not allow comments containing only spaces.

4.1.27 1.1.20

1. Provide two more custom listing styles.
2. Listings can now be filtered by tag.

4.1.28 1.1.19.3

1. Use *django-setuptest* 0.1.4. It handles South migrations correctly.
2. Use workaround so *jmbo-sitemap* works correctly again.

4.1.29 1.1.19.2

1. Really do what is stated in 1.1.19.1.

4.1.30 1.1.19.1

1. Found a critical error in legacy Jmbo code that is triggered by *jmbo-sitemap* URL pattern. Remove *jmbo-sitemap* URL patterns.

4.1.31 1.1.19

1. Remove potential *get_preference* cache key collision.
2. Port XML sitemap over to *jmbo-sitemap*.

4.1.32 1.1.18.2

1. Protect comment creation against manually crafted POSTs.

4.1.33 1.1.18.1

1. Hotfix. Fix bug where page change form did not display rows.

4.1.34 1.1.18

1. Change listing to accept multiple categories. A South data migration is involved and should work without issue, but it is recommended to backup your database.
2. Generate intentionally simple XML sitemap from the main navigation elements.
3. Offer Google Oauth2 login.

4.1.35 1.1.17

1. Web promo listing now displays pinned items.
2. Make ajax pagination more robust.
3. Allow @ in username.
4. Friendlier admin form when setting required fields in Registration Preferences.

4.1.36 1.1.16.1

1. Hotfix. foundrycache template tag was using wrong class to compute key.

4.1.37 1.1.16

1. Ensure that *user_logged_in* signal is dispatched when a user joins.
2. Don't allow the creation of a *BlogPost* where the *content* field contains scripting.
3. Map as many fields as possible to member when doing Facebook Connect.
4. Twitter Oauth is now standard functionality.
5. *base_inner.html* provides now has an extratitle block.
6. A comment posted to eg. basic will now show up in the other layers comprising the same logical site.
7. Flatpages are now part of our standard set of products.
8. Through-the-web configurable caching for rows, columns, tiles, menus and navbars.
9. Identify poorly performing areas and optimize code.

4.1.38 1.1.15

1. The Open Graph site description can now be set under General Preferences.
2. Allow dot in username.

4.1.39 1.1.14

1. Exclude gallery images from search results.
2. Include URLs from *jmbo-gallery*.

4.1.40 1.1.13

1. Use *django-banner>=0.2.2*. DFP banners loaded by ajax will now work.

4.1.41 1.1.12

1. Fire *onListingRefresh* event when listing is updated via ajax. Extra *target* parameter is passed to handler.
2. Basic ajax comment loading until jQuery-replacement is added.
3. Add name attribute to logo anchor so it is possible to jump to top of page.
4. Ajaxify view modifier navigation on listings.
5. Use *django-dfp>=0.2* which works across all browsers.

4.1.42 1.1.11

1. Add an index on Member.last_seen - useful for fast online user queries.

4.1.43 1.1.10

1. The *jmbo-banner* migration dependency was not in the correct migration step. Fixed.

4.1.44 1.1.9

1. Initial migration now depends on *jmbo-banner* migrations.

4.1.45 1.1.8

1. Restore version of *jmbo-banner* to 0.2.

4.1.46 1.1.7

1. Hotfix release. Use safe method to get HTTP_USER_AGENT in middlewares since it might not be present.
2. Deprecated. Use 1.1.8.

4.1.47 1.1.6

1. Hotfix release. An url import went missing.
2. Deprecated. Use 1.1.8.

4.1.48 1.1.5

1. Newer version of *jmbo-banner* implies a DFP header to be added to the base template.
2. Deprecated. Use 1.1.8.

4.1.49 1.1.4

1. Add optional CSS classes to page rows and columns.
2. Add last_seen field to Member and a middleware to update this timestamp at most every 5 minutes.

4.1.50 1.1.3

1. Use *django-social-auth* to authenticate against external providers. You must add *social_auth* to *INSTALLED_APPS* and set *SOCIAL_AUTH_USER_MODEL = 'foundry.Member'* at the very least. See the django-social auth documentation for more settings.
2. Drop the wizard style of registration. This is required for consistent UX when registering via Facebook.
3. Listings no longer include unpublished items that are referenced by the Content or Pinned fields.

4.1.51 1.1.2

1. Fix migration 0045 which would case South to complain about a previous set not being frozen.
2. Page objects can now be styled with extra CSS. This is useful when using a page as a campaign.

4.1.52 1.1.1

1. Filter Foundry comments by content type in admin.
2. Remove redundant chatroom detail template. It caused a comment count bug.
3. Allow social sharing of content even if it is a private site.
4. Remove jquery from basic layer since it causes out of memory errors on some devices. We will in future look for an API compatible replacement.
5. Add *jmbo-twitter* as dependency.
6. Provide three customizable listings to enable developers to easily add more listings.

4.1.53 1.1

1. Rename potentially confusing photosizes used in listing item templates. Old photosizes are retained for backward compatibility. If your app redefines a photosize for *listing_** then you must update those photosize names.
2. Handle favicon.ico requests so they do not 404.
3. Include *jmbo-gallery* admin urls.

4.1.54 1.0.1

1. Make fields in registration form reorderable.
2. Set initial values for location and age in registration form, when possible.
3. Remove hack to django-autopaginate to allow last page as default view. We have our own replacement autopaginate tag now.

4.1.55 1.0

1. Patch django.contrib.sites.models.Site.__unicode__ so it returns name and not domain. The UI gets confusing since we have up to three sites comprising one logical mobi site.
2. Listings now have automatic RSS feeds.
3. Comment form now fires up correct virtual keyboard for a smart phone.
4. Logged in members can now flag offensive comments. After three flags a moderator is notified.
5. Some IP addresses can now be allowed to bypass the age gateway / private site.
6. Listing gets an optional RSS feed.
7. Simplified paginator. No more breadcrumbs.
8. Show less metadata in mobi listings.
9. Ditch addthis sharing widget. It is too slow.
10. Simplified commenting and chatroom. Removed some navigation links.
11. Some user agents can now be allowed to bypass the age gateway. This allows bots to crawl the site.
12. Up required jmbo to 1.0.

4.1.56 0.7.2

1. Hotfix. Apps with empty URL patterns cause infinite recursion when adding a page.

4.1.57 0.7.1

1. Hotfix. Remove references deprecated *jmbo-gallery* views.

4.1.58 0.7

1. A listing now has an optional view modifier. This makes it possible to filter or order the listing.
2. *compute_settings* function is now redundant thanks to the introduction of *foundry.finders.FileSystemLayerAwareFinder*. Add this finder to STATICFILES_FINDERS as the first item.
3. Gallery specific code ported to *jmbo-gallery*. *base_inner.html* has a new link to gallery CSS and JS. If you have a customized template then update accordingly.
4. Up required *jmbo-gallery* to 0.1.

4.1.59 0.6.4

1. Replace deprecated message_set call.

4.1.60 0.6.3

1. Move FileSystemStorage listdir monkey patch to __init__.py so it is applied for collectstatic.

4.1.61 0.6.2

1. Django 1.4 incompatibilities with login and password reset fixed.
2. More tests.

4.1.62 0.6.1

1. Change admin static file urls to use 'static' filter instead of deprecated 'ADMIN_MEDIA_PREFIX'.

4.1.63 0.6

1. Up required jmbo to 0.5. Django 1.4 now implicitly required. You may get errors on template loaders not being found. See the Django 1.4 changelog in that case.

4.1.64 0.5.1

1. Clean up ajax batching of listings for basic and smart layers.
2. View modifiers and modelbase_list.html style templates are not ajaxified anymore.
3. Country model has new field country code.
4. Up required jmbo to 0.4.

4.1.65 0.5

1. "More" style batching for smart layer.
2. Listings now have optional pinned items which are anchored to the top of a listing.
3. Default photosizes for basic, mid, smart and web. Some old settings have changed so existing images may be scaled differently.

4.1.66 0.4

1. *layered* decorator so you can write different views for different layers without cluttering urls.py.

4.1.67 0.3.10

1. Translation for search form.
2. Member profile editing regression fixed.

4.1.68 0.3.9

1. Searching now working.

4.1.69 0.3.8

1. Bug fix for regression introduced into 0.3.7.

4.1.70 0.3.7

1. Listings being used within a tile can now choose whether to display a title.
2. Columns now have an optional title.

4.1.71 0.3.6

1. Demo is now part og jmbo-skeleton.
2. Minimum jmbo version required is now >= 0.3.4.
3. Management command load_photosizes loads photosizes in a sane way.

4.1.72 0.3.5

1. Adjust South migration dependencies.
2. Simplify and extend demo.

4.1.73 0.3.4

1. Batching on tastypie listing API.
2. Remove django-ckeditor dependency. Handled by jmbo-post.
3. Patch CsrfTokenNode.render so the input is not wrapped in a hidden container.

4.1.74 0.3.3

1. Version pins for jmbo and jmbo-post.

4.1.75 0.3.2

1. Use slug for lookups in tastypie API.

4.1.76 0.3.1

1. Chatrooms and normal comments can now have distinct appearances. jmbo>=0.3.1 required.

4.1.77 0.3

1. Reduce ajax polling when user is inactive
2. django-tastypie support added. jmbo and jmbo-post have minimum version requirements.

4.1.78 0.2.2

1. Pin django-ckeditor to >= 3.6.2
2. Remember me field now on login and join forms. Checked by default.
3. Any call to get_XXX_url is now layer aware.
4. Comment posting now ajaxified depending on browser capabilities.

4.1.79 0.2.1

1. Remove dependency links.

4.1.80 0.2

1. Add a base_inner.html template so it is easier to override base.html.
2. Patch listdir so collectstatic does not fail on custom layers for third party foundry-based products.

4.1.81 0.1

1. Use Jaro Winkler for matching naughty words.

4.1.82 0.0.2 (2011-09-27)

1. Detail view.
2. Element preferences.

4.1.83 0.0.1 (2011-09-21)

1. Initial release.

CHAPTER 5

Code

https://github.com/praekelt/jmbo-fouundry

CHAPTER 6

Versions and Compatibility

Jmbo Foundry is currently developed against the previous stable version of Django.

- Jmbo Foundry 2.x: Compatible with Django 1.6
- Jmbo Foundry 1.x: Compatible with Django 1.4