
jmbo-Documentation

Release 2.0.6

Praekelt

May 22, 2017

Contents

1	Overview	3
2	Installation	5
3	Internals	7
3.1	ModelBase	7
3.2	Templates	11
3.3	Customizing templates	13
4	Help	15
4.1	Changelog	15
5	Code	23
6	Versions and Compatibility	25

Jmbo is a lightweight and unobtrusive CMS geared towards the publishing industry. Jmbo introduces an extensible base model and provides segmented templates that are easy to customize.

This documentation covers version 2.0.6 of **ljmbol**.

Note: **ljmbol** 2.0.6 requires Django 1.6. More on *Versions and Compatibility*.

CHAPTER 1

Overview

Jmbo is a CMS built on Django enabling you to rapidly build multilingual web and mobi sites with the minimum amount of code and customization.

The Jmbo base product introduces abstract models and concepts that are used in Jmbo products.

CHAPTER 2

Installation

Jmbo 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.

ModelBase

Jmbo itself has no content type. For our purposes assume a trivial content type called Trivial Content which subclasses ModelBase.

Add trivial content

Title: 1
A short descriptive title.

Subtitle: 2
Some titles may be the same and cause confusion in admin UI. A subtitle makes a distinction.

Description: 3
A short description. More verbose than the title but limited to one or two sentences.

Publishing (Show) 4

Meta (Show) 5

Image 6

Image: No file chosen

Crop from: ▾

Effect: ▾ +

Commenting (Show) 7

Liking (Show) 8

Image overrides 9

Image	Crop from	Effect	Photosize	Delete?
<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Center (Default)"/> ▾	<input type="button" value="-----"/> ▾ +	<input type="button" value="-----"/> ▾ +	
<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Center (Default)"/> ▾	<input type="button" value="-----"/> ▾ +	<input type="button" value="-----"/> ▾ +	
<input type="button" value="Choose File"/> No file chosen	<input type="button" value="Center (Default)"/> ▾	<input type="button" value="-----"/> ▾ +	<input type="button" value="-----"/> ▾ +	

+ Add another Image Override

¹⁰
 ¹¹
 ¹²
 ¹³

1. Title is self-explanatory.
2. Subtitle is an optional field that is only visible in the admin interface. If you have two items with the same title (a common case in multilingual sites) then add extra text in the subtitle to distinguish between the items.
3. Description is an optional field. It is good practice to set it since sites like Facebook read it when content is shared.
4. See below.
5. See below.
6. You should always attempt to set an image since sites look dull without them! All popular formats up to 1MB in size are supported. An aspect ratio of 4:3 is preferred but an attempt is made to convert the uploaded image into a 4:3 format. Applications may override this aspect ratio as they wish.

Crop from and Effect are for advanced use. Crop from is used when the default cropping algorithm is eg. chopping a person's head off.

7. See below.
8. See below.
9. Image overrides are for advanced use. The image you upload in 6 is converted behind the scenes into potentially many sizes. If the algorithm produces undesirable results you can override each size as required.
10. Save and publish saves the item and makes it publicly accessible.
11. Save and add another saves the item. You are redirected to a screen to add a new trivial content item.
12. Save and continue editing saves the item. You stay on the edit screen.
13. Save saves the item. You are redirected to a listing of all trivial content items.

Publishing (Hide)

Sites: 1

Mobi site (basic)
Mobi site (mid)
Mobi site (smart)
Web site

Click an item to select

- **Mobi**
- Web

+ Makes item eligible to be published on selected sites. Hold down "Control", or "Command" on a Mac, to select more than one.

Publish on: 2 **Date:** Today |

Time: Now |

Date and time on which to publish this item (state will change to 'published').

Retract on: 3 **Date:** Today |

Time: Now |



Date and time on which to retract this item (state will change to 'unpublished').

Publishers: 4



Makes item eligible to be published on selected platform. Hold down "Control", or "Command" on a Mac, to select more than one.

1. Jmbo uses the Django sites framework. This means items can be selectively published to one or more sites. Select the applicable sites, or use the shortcuts to select a group of sites.
2. Publish on is an optional date on which to publish the item. If you leave it empty you must publish the item manually, eg. by clicking Save and publish.
3. Retract on is an optional date on which to retract the item. If you leave it empty you must unpublish the item manually, eg. by clicking Save and unpublish.
4. Publishers is marked for deprecation. The idea was to publish an item to eg. Facebook. Unfortunately those platforms often have unstable API's, making it impractical.



Meta (Hide)

Categories: **1**  



Categorizing this item. Hold down "Control", or "Command" on a Mac, to select more than one.

Primary category: **2**  



Primary category for this item. Used to determine the object's absolute/default URL.

Tags: **3**  

Tag this item. Hold down "Control", or "Command" on a Mac, to select more than one.

Created Date & Time: **4** **Date:** Today |  **Time:** Now | 

Date and time on which this item was created. This is automatically set on creation, but can be changed subsequently.

Owner: **5**  

1. Jmbo uses categories from `django-category`. Categorizing an item enables you to display it in a certain context. An example is a listing (as defined by `jmbo-foundry`) of all items in a category.
2. The primary category is considered the most important category. Once again it enables you do display the item in a certain context.
3. Freeform tagging of an item allows more ways of searching for it.
4. Created date and time is when the item was created. If you leave it empty it is automatically set to the current time.
5. The owner of the item. The dropdown may change into an autocomplete field if there are many people to choose from.

Commenting (Hide)	
<input checked="" type="checkbox"/> Commenting Enabled 1	Enable commenting for this item. Comments will not display when disabled.
<input checked="" type="checkbox"/> Anonymous Commenting Enabled 2	Enable anonymous commenting for this item.
<input type="checkbox"/> Commenting Closed 3	Close commenting for this item. Comments will still display, but users won't be able to add new comments.
Liking (Hide)	
<input checked="" type="checkbox"/> Liking Enabled 4	Enable liking for this item. Likes will not display when disabled.
<input checked="" type="checkbox"/> Anonymous Liking Enabled 5	Enable anonymous liking for this item.
<input type="checkbox"/> Liking Closed 6	Close liking for this item. Likes will still display, but users won't be able to add new likes.

1. Enable commenting for this item. Comments will not display when disabled.
2. Enable anonymous commenting for this item.
3. Close commenting for this item. Comments will still display, but users won't be able to add new comments.
4. Enable liking for this item. Likes will not display when disabled.
5. Enable anonymous liking for this item.
6. Close liking for this item. Likes will still display, but users won't be able to add new likes.

Templates

Jmbo provides a generic detail template `templates/jmbo/modelbase_detail.html` for subclasses. This template has been aliased to `object_detail.html` to follow Django naming conventions:

```
{% extends "base.html" %}
{% load jmbo_inclusion_tags jmbo_template_tags %}

{% block extratitle %} - {{ object.title }}{% endblock %}

{% block extrameta %}
    {% jmbocache 1200 'object-detail' object.id object.modified %}
    <link rel="canonical" href="{{ object.get_absolute_url }}" />
    <meta name="description" content="{{ object.description|default_if_none:'' }}" />
    <meta name="keywords" content="{{ tags|join:', ' }}" />
    <meta property="og:title" content="{{ object.title }}" />
    <meta property="og:type" content="article"/>
    <meta property="og:url" content="http{% if request.is_secure %}s{% endif %}://{{ request.get_host }}{{ object.get_absolute_url }}" />
    <meta property="og:image" content="http{% if request.is_secure %}s{% endif %}://{{ request.get_host }}{{ object.image_detail_url }}" />
{% endblock %}
```

SOME TRIVIAL CONTENT

(0 COMMENTS) | LIKE 0



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

 Share
 Tweet

0 COMMENTS **ADD YOURS**

ADD COMMENT

Fig. 3.1: A trivial item as it is seen by the public.


```

        {% endif %}
        <meta property="og:description" content="{ object.description|default_if_
↪none:'' }}" />
        {% endjmbocache %}
    {% endblock %}

{% block content %}
    <div class="object-detail {{ object.class_name.lower }}-detail">

        {% with object.as_leaf_class as object %}
            {% object_header object %}
            {% render_object object "detail" %}
            {% object_footer object %}
            {% object_comments object %}
        {% endwith %}

    </div>
{% endblock %}

```

Jmbo breaks a detail template into a number of elements that are customizable per content type by creating specially named templates:

1. Header provided by `templates/jmbo/inclusion_tags/object_header.html`.
2. Body (the meat of the item) provided by `templates/jmbo/inclusion_tags/modelbase_detail.html`. This template has been aliased to `templates/jmbo/inclusion_tags/object_detail.html` to follow Django naming conventions.
3. Footer provided by `templates/jmbo/inclusion_tags/object_footer.html`.
4. Comments provided by `templates/jmbo/inclusion_tags/object_comments.html`.

Customizing templates

Custom templates can be created for the following:

- Individual models that extend `ModelBase`
- All application models that extend `ModelBase`

Objects extending `ModelBase`

Create `templates` and `inclusion_tags` folders.

In your app folder you require the following folder structure:

- `<app name>/templates/jmbo/inclusion_tags/`

Create templates to override

Create template HTML files inside:

- `<app name>/templates/jmbo/inclusion_tags/object_<type>.html`

Creating object templates in this folder will override all objects that extend `ModelBase`.

Eg. Creating `object_header.html` and editing its content will change what is displayed on your page headers.

Default templates

Default Inclusion Tags:

This is what the templates look like by default.

object_header.html

object_detail.html

object_footer.html

App specific models

The folder structure changes slightly.

- *<app name>/templates/<app name>/inclusion_tags/object_<type>.html*

This will override all Models that extend ModelBase in the app, while external ModelBase objects can still be overridden separately.

Alternatively each app Model can also be overridden to use their own separate templates.

Individual models

The template naming convention changes slightly.

- *<app name>/templates/<app name>/<model>/inclusion_tags/object_<type>.html*

Instead of generic object templates, model specific templates are now used.

Template resolution

ModelBase templates get resolved in the following order:

1. *<app name>/inclusion_tags/<model>_<type>.html*
2. *<app name>/<model>/inclusion_tags/object_<type>.html*
3. *<app name>/inclusion_tags/object_<type>.html*
4. *jmbo/inclusion_tags/object_<type>.html*

The list from top to bottom lists the order in which templates are found and resolved. Paths that are higher up take priority and are rendered. If template is not found in any of these paths a default template is used.

Changelog

2.0.6

1. Make object detail template resolution follow the standard Django naming conventions. Backward compatibility is preserved.
2. Move test templates into tests directory.
3. Add dependency on `django-ultracache` and defer `jmbocache` template tag to `ultracache` template tag.

2.0.5

1. Gracefully handle missing images in the API.
2. Change the site information in the `unicode` method to be less overwhelming.
3. Use built-in jQuery for autosave function.

2.0.4

1. Patch `ImageModel` delete to handle null image.

2.0.3

1. Limit photologue to <3.2 because they have stopped supporting Django 1.6.

2.0.2

1. Disable more filters so Oracle can work.

2.0.1

1. Disable advanced admin change list filtering if Oracle is the database. The Oracle adapter is buggy.

2.0.0

1. Allow per content type customization of object header and footer.
2. Select all sites initially for new items.
3. The API now dereferences resource URI to the leaf class if possible.
4. Ensure image field is optional on ModelBase database table as well.

2.0.0a1

1. Move to Django 1.6 support. Django 1.4 support is deprecated. For Django 1.4 use Jmbo 1.x.
2. Add *Clone this item* button to change forms.
3. Deprecate gizmo, “wide” template, Pin class.
4. Deprecate own class based generic views in favour of Django’s equivalent.
5. Deprecate views related to show objects per category. *jmbo-foundry* offers a much more powerful solution and scales better.
6. Limit Relation change form to only ModelBase subclasses.
7. Deprecate smart_url template tag because Django url template tag does the same now.
8. Move back to mainline *django-photologue*.
9. API now includes image detail url.

1.2.0

1. Use renamed django-photologue-praekelt.
2. SEO optimizations in templates.
3. Make it possible to reach a detail page through a category.

1.1.7

1. Bump to resolve missing version bump in setup.py.

1.1.6

1. API now includes image detail url.
2. URL pattern to resolve detail page through category.

1.1.5

1. Ignore result of celery tasks as appropriate.

1.1.4

1. Relax uniqueness constraint on slugs.

1.1.3

1. Fix modelbase editing where location field was added to wrong fieldset.

1.1.2

1. Add logging to *jmbo* cache template tag.

1.1.1

1. Add a template *base.html* so unit tests that render detail pages work.
2. Reshuffle the test layout.

1.1

1. Location aware functionality now only takes effect if both 'django-atlas' and *django.contrib.gis* are installed.
2. *django-photologue* 2.10 is now the minimum version.

1.0.14

1. Add *rel="nofollow"* on view modifier links.
2. Fix *render_object* where context was copied instead of using push and pop.
3. Simplify sharing link creation.

1.0.13

1. Fix a broken find link in *setup.py*.

1.0.12

1. Fix incorrect file permissions.

1.0.11

1. Add functionality to periodically autosave certain fields on the change form.
2. Change change list ordering to be *-publish_on, -created*.
3. Change *get_related_items* ordering to be *-publish_on, -created*.
4. Use a celery task to publish content.
5. Permalink now links to all sites.

1.0.10

1. Change secretballot usage so it does not hijack the objects manager anymore.
2. Add *owner_override* and *image_attribution* fields.

1.0.9

1. Change permitted manager and generic object detail so staff can preview unpublished content.
2. Aggregate total comments and likes onto *ModelBase* to prevent expensive queries.

1.0.8

1. Add caching template tag *jmbocache* which automatically adds the *SITE_ID* as part of the cache key.

1.0.7

1. Generic caching on detail templates.
2. Share on Google.

1.0.6

1. Add a list filter in admin to filter *ModelBase* objects by site and site group.
2. *ModelBase.__unicode__* includes the site name - non-admin templates that rely on *__unicode__* will have to be updated.
3. Set title, description and keywords meta tags on detail page.
4. *comment_count* is now aware that multiple sites may comprise a logical site.

1.0.5

1. Make *jmbo_publish* command timezone-aware, ensuring that it works with old, naive timestamps.

1.0.4

1. Restore crop from field to a more prominent position.

1.0.3

1. Simplify the change form. Move advanced fields into their own section.

1.0.2

1. Ensure the leaf object is passed to template tags in *modelbase_detail.html*.
2. *get_related_items* parameter *name* is now optional. The sorting has changed to reverse on modified (our default sorting).

1.0.1

1. *as_leaf_class* method would break if two models had the same name. Fixed.

1.0

1. Jmbo is now location aware. This requires a manual upgrade of libraries and existing databases. DO NOT UPGRADE to 1.0 without preparation. If you are on Ubuntu then it is as simple as running the interactive `convert_to_geodb_ubuntu.sh` script.

0.5.5

1. *modelbase_detail* inclusion template now has a block for easier re-use.
2. Simplified paginator. No more breadcrumbs.
3. Introduce *object_footer* template which shows sharing links.
4. `can_comment` has an API change. It has always only been used internally and should not cause problems.
5. README.rst gets friendlier documentation.

0.5.4

1. Pin Django on 1.4.x range.

0.5.3

1. Add *Save and publish* and *Save and unpublish* buttons to edit form.

0.5.2

1. Use `django.jQuery` instead of `$` to trigger publish ajax call. `$` is not necessarily available.

0.5.1 (2012-08-20)

1. `on_likes_enabled_test` and `on_can_vote_test` signal receivers now only checks `ModelBase` based objects. Also updated for compatibility with `django-likes` 0.0.8, which updated its signal's `obj` param to conventional `instance`. `django-likes >= 0.0.8` is now required for correct operation.

0.5

1. Django 1.4 compatible release. Django 1.4 is now required.

0.4

1. Detail templates can now be customized per model. Create `{app_label}/{model}_detail.html`.
2. `publish_on` and `retract_on` filters are now applied via management command `jmbo_publish`. Run it via cron.
3. Published state is not directly editable through change form anymore. It is now an action.

0.3.4 (2012-06-26)

1. Natural key support for dumping and loading data.

0.3.3 (2012-06-20)

1. Use Pillow instead of PIL.

0.3.2

1. Use slug for lookups in tastypie API.

0.3.1 (2012-06-15)

1. Add a decorator `register_tag` that can accept a softcoded list of templates.

0.3 (2012-06-14)

1. `django-tastypie` support added

0.2.6 (2012-06-07)

1. Add `image_list_url` to `Modelbase`.
2. Pin `django-setuptest` to 0.0.6 because of issue in 0.0.7

0.2.5 (2012-05-11)

1. Admin category filtering now filters on both `categories` and `primary_category` fields.

0.2.4

1. Remove dependency links in `setup.py`.

0.2.3 (2012-05-08)

1. render_object tag now fails with clear TemplateDoesNotExist exception.

0.2.2

1. Include category filtering in admin.

0.2.1

1. Find links in setup.py

0.2

1. Add Opengraph metadata tags to detail view.
2. Add dependency on django-sites-groups.
3. Setup South migration chain.

0.1.20

1. Bring pager HTML and CSS in line with django-pagination.
2. Add wrapping div to comments UI.
3. Fix admin interface bug where some fields were duplicated.
4. Reverse lookup for <content_type>_object_detail now works for model names that may contain spaces, eg. 'Blog Post'.
5. Add fallback to modelbase detail view to get_absolute_url.
6. Add ability to limit size of queryset for generic views.
7. Afrikaans and French translations.
8. Make it possible to specify a custom photosize per content type.
9. Introduce a new optional field 'subtitle' for friendlier admin UI.
10. Add South migrations. Existing installations must be upgraded using ./manage.py migrate jmbo 0001 --fake and then ./manage.py migrate jmbo.

0.1.9 (2011-09-27)

1. Added primary category field on ModelBase.
2. Allow for modifier on humanize time diff tag.
3. Added category pin model and admin override.

0.1.7 (2011-06-15)

1. Jmbo rename.

0.1.6

1. Added state admin bulk actions.

0.1.5

1. Use photologue 2.6.praekelt

0.1.4

1. Generate slug optimization.

0.1.3

1. Refactored ModelBase.comment_count to resolve comments for leaf class or modelbase content types.

0.1.2

1. Generic form issues corrected.

0.1.1

1. Use django-photologue 2.5.praekelt

0.1.0

1. Improved generate_slug utils method.
2. Removed ModelBaseAdminForm.

CHAPTER 5

Code

<https://github.com/praekelt/jmbo>

Versions and Compatibility

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

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