Jmbo Listing Documentation

Release 2.0.6

Praekelt

May 02, 2017

Contents

CHAPTER 1

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.

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.

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/listing/templatetags/horizontal.html`:

```
{% for object in object_list %}
    <div class="item {% if forloop.first %}first{% endif %} {% if forloop.last %}last{
↪% endif %}
                        item-{{ object.content_type.app_label }}
                        item-{{ object.content_type.app_label }}-{{ object.content_
↪type.model_name }}">

        {% 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/{{ 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/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.

Custom listings

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

```
from listing.styles import AbstractBaseStyle


class MyListing(AbstractBaseStyle):
    template_name = "myproduct/templatetags/mylisting.html"
```

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

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" %}
```