django-fack Documentation

Release 1.0

Kevin Fricovsky, Jacob Kaplan-Moss

Sep 27, 2017

Contents

This is a simple FAQ application for a Django powered site, featuring:

- The basic Q&A model you'd expect.
- Question are grouped into topics.
- Topics can be ordered and arranged, as can questions within topics.
- Built-in views to drill down by topic and question, and individual question detail pages (for permalinks).
- A view for users to submit new questions (with or without answers). These go into moderation queue and need to be marked "active" before they'll show up on the site.

There's an example app (distributed with the source) to try out if you'd like to get a taste of the app.

Django-Fack requires Django 1.3+ and Python 2.5+.

For more details, read on:

CHAPTER 1

Getting starting with Django-Fack

Installation

1. `pip install django-fack`
2. Add `"fack"` to your `INSTALLED_APPS` setting.
3. Wire up the FAQ views by adding a line to your URLconf:

```
url('^faq/', include('fack.urls'))
```

If you'd like to load some example data then run `python manage.py loaddata faq_test_data.json`

FAQ entries can be edited in the admin, and also via the submit view (at `<prefix>/submit`).

The app's written with quite a bit of customization in mind; see *Customizing Django-Fack* for details. You'll almost certain want to *customize the templates* to make the app "fit in" to your site.

Example Site

There is a stand-alone example site distributed with the source in the `example/` directory. To try it out:

1. Install Django-Fack (see above).
1. Run `python manage.py syncdb`

   This assumes that sqlite3 is available; if not you'll need to change the `DATABASES` setting first.
3. Load some example data by running `python manage.py loaddata faq_test_data.json`
4. Run `python manage.py runserver` and you will have the example site up and running. The home page will have links to get to the available views as well as to the admin.

The capability to submit an FAQ is available and works whether or not you are a logged in user. Note that a staff member will have to use the admin and review any submitted FAQs and clean them up and set them to active before they are viewable by the end user views.

CHAPTER 2

Customizing Django-Fack

Customizing templates

The one thing you'll probably want to do quickly is to make some custom templates so that the FAQ app "fits" with the rest of your site. Django-Fack uses the following templates:

Template name	Use
`faq/base.html`	The base template that all the other included templates extend by default. For a quick and dirty customization you can just override this one template, making sure to provide a `content` block, and everything'll look OK. But to get a more custom feel you'll probably want to move on
`faq/question_detail.html`	Used by an individual "question" detail page. The context has one variable, `question`, which is the `Question` object.
`faq/submit_question.html`	Used by the view that allows users to submit a question. The context has the obligatory `form` variable, which is a form containing `topic`, `text`, and `answer` fields.
`faq/submit_thanks.html`	Shown after a successful FAQ submission. No context.
`faq/topic_detail.html`	Shows all the FAQs in a given topic. **Context:** • `topic` - a `Topic`. • `questions` - list of `Question` objects in the given topic.
`faq/topic_list.html`	Shows a list of all topics. Context contains a `topic` variable, which is a `Topic` object.

Subclassing FAQ views

All of the views in Django-Fack use class-based generic views and are designed to be extended. Consulting the source is your best bet here: the code's fairly clear, and prose will only complicate things.

If you do choose to subclass views, you'll most likely want to write your own URLconf (instead of `fack.urls`). Remember that if you maintain the names given by the default URLconf all the existing links will continue to work.