django-flatpages-x Documentation

Release 0.0.2

Christopher Clarke

August 06, 2013

CONTENTS

An extension to provide easy integration between `django.contrib.flatpages` and Django-filer

> "A file management application for django that makes handling of files and images a breeze".

Thw `django-flapages-filer` app aims to provide a seamless experience to users of the standard flatpages app. It enhances the standard flatpages Admin with inline forms that allow you to include references to your filer based images and files (attachments). It also allows you to easily maintain content using a markup format such as `Markdown`.

However, the `contrib.Flatpage` model (content, titles) is not be affected by installing or removing this app. Your templates do not have to be modified as they will still utilize the `{{ flatpage.content }}` and `{{ flatpage.title }}` context variables. Once it is installed, content changes are actually stored in a related model `flatpages_filer.models.Revisions` usually in a markup format. The Revision model which also keeps track of content changes making it easy to revert to an earlier versions. The modified `Admin ChangeForm` ensures you can view the latest version of your content in the appropriate markup format and when you save a `flatpage` this markup content will be rendered to to html via a specified parser.

Additionally, `django-flatpages-filer`:

- Comes with a default `Markdown` parser. By default the `codehilite` and `extra` extensions are supported but you can specify your own list of extensions in your Django settings
- You can easily write your own parser to support to support markup formats such as rst or creole.
- Provides optional support for the excellent **markItUp** widget. This requires the installation `django-markitup`.
- Provides templatetags to support *HTML metatags* such as keywords and descriptions in flatpages.

CHAPTER

ONE

CONTRIBUTORS

- Christopher Clarke
- Lendl R Smith
- Mikhail Andreev
- Raumkraut

CHAPTER

TWO

QUICKSTART

Create a virtual environment for your project and activate it:

```
$ virtualenv mysite-env
$ source mysite-env/bin/activate
(mysite-env)$
```

Next install `flatpages_filer`

```
(mysite-env)$ pip install django-flatpages-filer
```

Add `flatpages_filer` to your INSTALLED_APPS setting.

Inside your project run:

```
(mysite-env)$ python manage.py syncdb
```

Django-flatpages-filer comes with support for Markdown

To include filer images in your content use a standard markdown image reference

```
![Atl text][filer_image.pk]
```

For a link to a file

```
[Atl text][filer_file.pk]
```

Where `pk` refers to the primary key of the filer file or image. To facilitate easy inclusion of the images and file attachments in your markdown content the `Flatpages Admin` now contains Inline image and file attachment forms which allow you to to associate the filer images or files with the `flatpage` once you save the `flatpage` the correct markdown image/file reference should is visible in Inline image form.

2.1 markItUp support

If you want to use the excellent markItUp! editor widget. Install django-markItUp:

```
(mysite-env)$ pip install django-markitup
```

You need a few configuration steps

1. Add 'markitup' to your INSTALLED_APPS setting.
2. Add the following to settings:

```
MARKITUP_SET = 'markitup/sets/markdown'
MARKITUP_SKIN = 'markitup/skins/markitup'
MARKITUP_FILTER = ('markdown.markdown', {'safe_mode': True})
```

3. You need to use the AJAX-based preview for the admin widget:

```
url(r'^markitup/', include('markitup.urls'))
```

in your root URLconf.

2.2 Markup Parsers

Django-flatpages-filer includes a Markdown parser that supports `codehilite` and `extra` extensions. You can support additional extensions by first installing the extension an adding the following to you Django settings

```
FLATPAGES_FILER_PARSER= ["flatpages_filer.markdown_parser.parse",
                         {'extensions': ['codehilite','extra', 'abbr']}]
```

You can also supply your own parser by setting the value for `FLATPAGES_FILER_PARSER` to point to your parser

```
FLATPAGES_FILER_PARSER= ["myproject.myapp.creole_parser.parse",
                         {'emitter': FilerEmmiter}]
```

Note we expect that your `parse` would have the following signature

```
parse(text, [list of `extensions, emitters etc.])
```

2.3 Migrating From Flatpages-x

Before installing to `django-flatpage-filer` dump the data in the Revison model

```
python manage.py dumpdata flatpages_x.Revision > revision.json
```

Remove `flatpage_x` from you settings and install django-flatpages-filer and add `flatpages_filer` to the settings. To import your revision data edit `revision.json` and change all occurences of `flatpages_x.revision` to `flatpages_filer.revision`. Finally, run

```
python manage.py loaddata ~/usr/folder/revision.json
```