django-markymark

unknown

Sep 12, 2022

CONTENTS

django-markymark provides helpers and tools to integrate markdown into Django.

CHAPTER

ONE

FEATURES

- Django form fields to integrate the bootstrap markdown editor (without the dependency on bootstrap)
- django-filer integration
- django-anylink integration

CHAPTER

TWO

REQUIREMENTS

django-markymark supports Python 3 only and requires at least Django 3.2

CHAPTER

THREE

PREPARE FOR DEVELOPMENT

A Python 3 interpreter is required in addition to Poetry.

```
$ poetry install
```

Now you're ready to start the example project to experiment with markymark.

```
$ poetry run python examples/manage.py runserver
```

CHAPTER

FOUR

RESOURCES

- Documentation
- Bug Tracker
- Code

Contents:

4.1 Installation

- Install with pip:

```
pip install django-markymark
```

- Your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
    # ...
    'markymark',
)
```

4.1.1 Configuration for Django-CMS

If you plan to use django-markymark in a Django-CMS project, you should change the following two settings to make sure there are no iconlibrary problems:

```
MARKYMARK_FONTAWESOME_CSS = None
MARKYMARK_ICONLIBRARY = 'fa4'
```

4.2 Configuration

Markymark uses settings to make the markdown field work in the django admin site.

4.2.1 Django Settings

MARKYMARK_EXTENSIONS

A list with extra extensions for the markdown field.

Default extensions:

```
MARKYMARK_EXTENSIONS = [
    'markymark.extensions.autolink',
]
```

MARKYMARK_FONTAWESOME_CSS

By default, the markdown editor uses FontAwesome to show some nice icons. FontAwesome is vendored and shipped with this library. You can override the path of the FontAwesome css file to use either the CDN version or something completly different.

```
# Override default setting.
MARKYMARK_FONTAWESOME_CSS = 'https://use.fontawesome.com/releases/v5.2.0/css/all.css'
```

MARKYMARK_ICONLIBRARY

By default, the markdown editor uses Font Awesome 5 compatible css classes for icons. In addition, you can change the used icon library to Font Awesome 4.

To change the icon classes to Fa4, use this setting:

```
MARKYMARK_ICONLIBRARY = 'fa4'
```

4.3 Extensions

4.3.1 Python Markdown

Regular extensions from Python Markdown for extra markdown features, can be used with markymark.

```
MARKYMARK_EXTENSIONS = [
    '..',
    'markdown.extensions.codehilite',
    'markdown.extensions.fenced_code',
    'markdown.extensions.tables',
]
```

4.3.2 Django-anylink

An extension for django-anylink a generic linking module for Django.

The following needs to be added to the settings:

```
MARKYMARK_EXTENSIONS = [
    '..',
    'markymark.extensions.anylink',
]
```

This extension has extra dependencies that need to be installed:

```
$ pip install django-markymark[anylink]
```

Note: For this extension additional settings are required that can be found in the django-anylink documentation

Warning: The JavaScript plugin overwrites the functionality of the "Link" button of the markdown editor with it's own implementation. Please be aware of that.

4.3.3 Django-filer

An extension for django-filer a file and image management application for django.

The following needs to be added to the settings:

```
MARKYMARK_EXTENSIONS = [
    '..',
    'markymark.extensions.filer',
]
```

This extension has extra dependencies that need to be installed:

```
$ pip install django-markymark[filer]
```

Note: For this extension additional settings are required that can be found in the django-filer documentation

Warning: The JavaScript plugin overwrites the functionality of the "Link" button of the markdown editor with it's own implementation. Please be aware of that.

4.3.4 Autolink

An extension to automatically create anchor tags from urls inspired by Github Flavored Markdown.

The following needs to be added to the settings:

```
MARKYMARK_EXTENSIONS = [
    '..',
    'markymark.extensions.autolink',
]
```

Example input/output:

```
http://www.example.com will turn into <a href="http://www.example.com">http://www.
↪example.com</a>
```

4.4 Usage

4.4.1 Model Field

```
from django.db import models

from markymark.fields import MarkdownField


class Post(models.Model):
    content = MarkdownField()
```

4.4.2 Template filter

To display the rendered markdown in your template.

```
{% load markymark %}

{{ obj.content|markdown }}
```

4.4.3 Running tests

To run the tests, you need to install the test requirements with Poetry:

```
$ poetry install
```

Now you can run the tests from the root folder of the package:

```
$ make tests
```

4.5 Changelog

4.5.1 3.0.0 (2022-09-12)

- Drop support for Django < 3.2, add support for 3.2
- Drop support for Python < 3.8

4.5.2 2.1.0 (2018-09-11)

- Add setting to switch icon library for markdown editor to FontAwesome 4

4.5.3 2.0.0 (2018-07-27)

- THIS IS A BREAKING RELEASE, if you need Django < 1.11 please stick to the 1.1.0 release.
- Drop Python < 3 support
- Drop Django < 1.11 support
- Cleanup code and add some more documentation

4.5.4 1.1.0 (2017-04-21)

- add table support including new button in editor

4.5.5 1.0.5 (2017-04-20)

- drop py26 support
- Fix anylink pop up, make it more flexible for custom project urls

4.5.6 1.0.4 (2017-02-21)

- Fix styling when using markymark in inlines with django-cms / djangocms-admin-style
- Force Pillow version for py26

4.5.7 1.0.0 (2016-11-24)

- Add support for Django 1.9
- Fix styling when using markymark with django-cms / djangocms-admin-style
- Improve autolink extension to prevent double-linking
- Javascript bugfixes

4.5.8 0.9.3 (2016-10-05)

- fix issue - MarkdownField should handle widget instances as well #7
- adjust install requirements to use 'official' python Markdown package
- some improvements

4.5.9 0.9.2 (2016-02-03)

- fix javascript - js was renamed from dismissAddAnotherPopup to dismissAddRelatedObjectPopup

4.5.10 0.9.1 (2015-09-24)

- fix MardownTextarea widget

4.5.11 0.9.0 (2015-09-24)

- add django 1.8 and python 3.5 support

4.5.12 0.8.3 (2015-07-08)

- fix - make "$" available in function only in markdown-editor.js

4.5.13 0.8.2 (2015-04-22)

- Add python 2.6 support
- Add support for south

4.5.14 0.8.1 (2015-03-16)

- Small style fix for preview button

4.5.15 0.8.0 (2015-03-12)

- Refactor JavaScript part of plugins to replace existing icons
- Fix bug in anylink js integration to avoid blank window after save

4.5.16 0.7.2 (2015-03-09)

- Fix anylink javascript to link to 'add' view

4.5.17 0.7.1 (2015-03-09)

- Remove padding from textarea

4.5.18 0.7 (2015-03-09)

- Fix release

4.5.19 0.6 (2015-03-09)

- Rework extensions to allow js/css files to be defined directly on each extension
- The return value of `render_markdown()` is now marked as safe
- Allow template-names to be overwritten
- Made settings easier to be overwritten, you can now import default settings from `markymark.defaults`
- Fixed contrib.anylink to avoid name clashes with other extensions named "link"
- Fix fullscreen icon integration

4.5.20 0.5 (2015-02-13)

- Removed anylink and filer extensions from being autoloaded.
- Removed dependency on floppyforms.

4.5.21 0.2..0.4 (2015-01-22)

- General cleanups and bugfixes.

4.5.22 0.1 (2015-01-22)

- Initial release.

Api documentation:

4.6 markymark.renderer

markymark.renderer.**initialize_renderer**(*extensions=None*)

Initializes the renderer by setting up the extensions (taking a comma separated string or iterable of extensions). These extensions are added alongside with the configured always-on extensions.

Returns a markdown renderer instance.

markymark.renderer.**render_markdown**(*value*, *extensions=None*)

Takes a text and a optional list of extensions and returns the rendered markdown text.

The result is marked safe.

4.7 markymark.fields

class markymark.fields.**MarkdownFormField**(**args*, ***kwargs*)

Bases: `CharField`

Form field to configure the markdown textarea widget if not already set.

__init__(**args*, ***kwargs*)

Update the kwargs to set the markdown widget. Special note: the provided widget should be a subclass of MarkdownTextarea, if not the provided widget will be ignored.

class markymark.fields.**MarkdownField**(**args*, *db_collation=None*, ***kwargs*)

Bases: TextField

Model field based on TextField with enabled markdown form field.

formfield(*form_class=<class 'markymark.fields.MarkdownFormField'>*, ***kwargs*)

Return a django.forms.Field instance for this field.

4.8 markymark.widgets

class markymark.widgets.**MarkdownTextarea**(**args*, ***kwargs*)

Bases: Textarea

Extended forms Textarea which enables the javascript markdown editor.

__init__(**args*, ***kwargs*)

Sets the required data attributes to enable the markdown editor.

property media

Returns a forms.Media instance with the basic editor media and media from all registered extensions.

4.9 markymark.templatetags

markymark.templatetags.markymark.**markdown_filter**(*value*, *extensions=None*)

Template which converts a provided value using markdown to html. Accepts additional extensions when rendering with markdown.

4.10 markymark.extensions

4.10.1 markymark.extensions.autolink

class markymark.extensions.autolink.**AutoLinkPostprocessor**(*md=None*)

Bases: Postprocessor

Post processor to look for valid URIs and converts them to html links.

AUTOLINK_RE = re.compile('(href="|src="|<a.*>)?(?:(https?|ftps?|file|ssh|mms|svn(?:
\+ssh)?|git|dict|nntp|irc|rsync|smb|apt|telnet|s?news|sips?|skype|apt)://
|(mailto:))([-A-Za-z0-9+&@#/%?=~_()|!:,.;]*[-A-Za-z0-9+&@#/%=~_()|]))

run(*text*)

Subclasses of Postprocessor should implement a *run* method, which takes the html document as a single text string and returns a (possibly modified) string.

class markymark.extensions.autolink.**AutoLinkExtension**(***kwargs*)

Bases: *MarkymarkExtension*

Extension to insert html links for certain URIs.

postprocessors = (<class 'markymark.extensions.autolink.AutoLinkPostprocessor'>,)

property media

markymark.extensions.autolink.**makeExtension**(***kwargs*)

4.10.2 markymark.extensions.base

class markymark.extensions.base.**MarkymarkExtension**(***kwargs*)

Bases: Extension

Base class for all markymark extensions.

Actually its just a markdown.Extension class with some media attached.

preprocessors = None

inlinepatterns = None

postprocessors = None

extendMarkdown(*md*)

Every extension requires a extendMarkdown method to tell the markdown renderer how use the extension.

property media

4.10.3 markymark.extensions.clean

class markymark.extensions.clean.**CleanExtension**(***kwargs*)

Bases: *MarkymarkExtension*

Extension to enable the cleanup plugin for the markdown editor.

class Media

Bases: object

js = ('markymark/extensions/clean.js',)

property media

markymark.extensions.clean.**makeExtension**(***kwargs*)

4.10.4 markymark.extensions.anylink

class markymark.extensions.anylink.**AnyLinkPostprocessor**(*md=None*)

Bases: Postprocessor

Post processor to look for anylink link tags, replaces these tags with html links.

LINK_RE = re.compile('(\\[link\\:(?P<id>\\d+)\\])', re.IGNORECASE)

run(*text*)

Subclasses of Postprocessor should implement a *run* method, which takes the html document as a single text string and returns a (possibly modified) string.

class markymark.extensions.anylink.**AnyLinkExtension**(***kwargs*)

Bases: *MarkymarkExtension*

Extension to insert django-anylink links into a markdown document.

postprocessors = (<class 'markymark.extensions.anylink.AnyLinkPostprocessor'>,)

class Media

Bases: object

js = ('markymark/extensions/anylink.js',)

property media

markymark.extensions.anylink.**makeExtension**(***kwargs*)

4.10.5 markymark.extensions.filer

class markymark.extensions.filer.**FilerPostprocessor**(*md=None*)

Bases: Postprocessor

Filer markdown extension for django-filer to show files and images.

FILE_RE = re.compile('(\\[file\\:(?P<id>\\d+)\\])', re.IGNORECASE)

run(*text*)

Subclasses of Postprocessor should implement a *run* method, which takes the html document as a single text string and returns a (possibly modified) string.

class markymark.extensions.filer.**FilerExtension**(***kwargs*)

Bases: *MarkymarkExtension*

Extension to look for file tags, replaces them with html tags. In case of image, the image is added as img-tag, files are added as download links.

postprocessors = (<class 'markymark.extensions.filer.FilerPostprocessor'>,)

class Media

Bases: object

js = ('markymark/extensions/filer.js',)

css = {'all': ('markymark/extensions/filer.css',)}

property media

markymark.extensions.filer.**makeExtension**(***kwargs*)

CHAPTER

FIVE

WHO'S MARKY MARK?

Marky Mark on Wikipedia

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

Symbols

A

C

E

F

I

J

L

M

P

R