Flask-MDE

Release 1.2.1

Mar 21, 2021

Contents

Release v1.2.1 *Install* | Example application

Pagedown Editor with Google code-prettify for Flask. WTForms integration supported. From v1.2.0 Flask-MDE supports pagedown-extra.

Pagedown is a Markdown editor and previewer popularised by its use on StackOverflow. You can use the *Flask-MDE* extension to integrate the Pagedown editor into your Flask application.

Read the *User Guide* to get started.

CHAPTER 1

Table of Contents

1.1 User Guide

1.1.1 Installation

Installing Flask-MDE is simple with pip. To install Flask-MDE, run the command

```
pip install Flask-MDE
```

in your terminal.

1.1.2 Basic usage

Flask-MDE can be used without wtforms integration.

For one Flask application, create the *Mde* object by passing it the application.

```
from flask import Flask, render_template
from flask_mde import Mde

app = Flask(__name__)
mde = Mde(app)

@app.route('/')
def index():
    return render_template(
        "index.html"
    )
```

This will allow us to access the object from the Jinja templates, without passing it as an argument to *render_template*.

Let's take a look at the various components that need to be included in the template

- The *editor method* of the object will render the Pagedown editor in the template.

- The *css property* will include the css files.
- The *js property* will include the js files.
- The *preview property* will render the preview panel.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    {{mde.css}}
</head>
<body>
    {{mde.editor()}}
    {{mde.preview}}
    {{mde.js}}
</body>
</html>
```

Optionally a name and default value can be passed to the editor:

```
{{mde.editor(name='e1', default='# Heading')}}
```

Another method editor_with_preview will render the editor and preview panel in one go. This is less flexible if you want finer control over the appearence of editor and preview.

```
{{mde.editor_with_preview(name='e1', default='# Heading')}}
```

If you are using an application factory, the following pattern is also supported.

```
mde = Mde()

def create_app():
    app = Flask(__name__)
    mde.init_app(app)
    return app
```

1.1.3 WTForms integration

For WTForms compatibilty, use the *MdeField* class. MdeField extends wtforms.fields.TextAreaField. MdeField can be customized via the Field definition. See: WTForms - Field definitions.

You must NOT however change the `id` of the field. This is used by the css and js files.

A Minimal example

app.py

```
from flask import Flask, render_template
from flask_mde import Mde, MdeField
from flask_wtf import FlaskForm
from wtforms import SubmitField

app = Flask(__name__)
mde = Mde(app)
app.config['SECRET_KEY'] = "your_secret_key_here"
```

(continues on next page)

(continued from previous page)

```
class MdeForm(FlaskForm):
    editor = MdeField()
    submit = SubmitField()


@app.route('/')
def index():
    form = MdeForm()
    return render_template(
        "index.html",
        form=form
    )
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    {{mde.css}}
</head>
<body>
    <form method="POST">
        {{form.csrf_token }}
        {{form.editor}}
        {{form.submit}}
    </form>
    {{mde.preview}}
    {{mde.js}}
</body>
</html>
```

Note that `form.editor` does not take any arguments. All the arguments need to be passed during field definition.

1.1.4 Converting to HTML

Submitted text will be markdown. If you need to convert to HTML, you can use the markdown library.

1.1.5 Sanitizing HTML

HTML sanitization can be used to protect against cross-site scripting (XSS) attacks by sanitizing any HTML code submitted by a user. Once you have converted markdown to HTML, it is a good idea to sanitize the HTML before displaying it on your site.

Bleach library can be used for this.

1.1.6 Making HTML pretty

When you convert the markdown to HTML and displayit back on your application, it won't be similar to the preview shown. This is because Google code-prettify's `prettyPrint` function is not called.

Flask-MDE provides the `make-pretty` css class to make your HTML similar to that shown in the preview.

```
<!DOCTYPE html>
<html lang="en">
<head>
    {{mde.css}}
</head>
<body>
    <div class="make-pretty">
        {{output_html|safe}}
    </div>
    {{mde.js}}
</body>
</html>
```

1.1.7 Pagedown Extra

From v1.2.0 Flask-MDE supports pagedown-extra. Pagedown Extra is a collection of Pagedown plugins to enable support for Markdown Extra syntax (such as tables). Checkout the example application code in the Github repo see how you can integrate using *python-markdown* and *pymdown-extensions*.

1.1.8 Example application

An example application is included in the Github repository. This application includes code for WTForms integration, HTML sanitization and displays prettified HTML back to the user. You can use this as a rough guide if you get stuck.

See a live demo of the example application.

1.2 API

class flask_mde.models.**MdeField**(*id='wmd-input'*, ***kwargs*)

MdeField extends wtforms.fields.TextAreaField

`id` defaults to `wmd-input`. Do NOT change this. You can change the other parameters in the WTForms Field base class.

class flask_mde.models.**Mde**(*app=None*)

Pagedown Editor Class.

Usage Model

```
app = Flask(__name__)
mde = Mde(app)
```

or

```
mde = Mde()

def create_app():
    app = Flask(__name__)
    mde.init_app(app)
    return app
```

editor(*name=''*, *default=''*)

Loads the Pagedown editor (without preview).

Parameters

- **name** (*str, optional*) – Name of the textaread field.
- **default** (*str, optional*) – Default value in the the textaread field.

Returns Editor markup.

Return type markupsafe.Markup

Example of usage in Jinja2

```
{{mde.editor(name='e1', default='# Heading')}}
```

preview
Loads the preview panel.

Accessible as `{{mde.preview}}` in Jinja2.

editor_with_preview(***kwargs*)
Loads the Pagedown editor and preview.

Parameters

- **name** (*str, optional*) – Name of the textaread field.
- **default** (*str, optional*) – Default value in the the textaread field.

Returns Editor markup.

Return type

markupsafe.Markup

Example of usage in Jinja2

```
{{mde.editor_with_preview(name='e1', default='# Heading')}}
```

This will give (almost) the same effect as using

```
{{mde.editor(name='e1', default='# Heading')}}
{{mde.preview}}
```

js
Loads the Javascript files.

Accessible as `{{mde.js}}` in Jinja2.

css
Loads the css files.

Accessible as `{{mde.css}}` in Jinja2.

1.3 Flask-MDE Changelog

FLask-MDE follows Semantic Versioning 2.0.0

1.3.1 1.2.1

- Replaced *HTMLString* with *markupsafe.Markup* for *MdeField*. Github Issue # 7 - Button Bar Not Displaying.

1.3.2 1.2.0

- Added support for pagedown-extra.

1.3.3 1.1.2

- Fixed application factory usage issue.

1.3.4 1.1.1

- `make-pretty` was not pretty printing `<pre>` tags. Fixed.

1.3.5 1.1.0

- `make-pretty` css class added to make the output HTML pretty.

1.3.6 1.0.0

- This is the Public API.

Python Module Index

f

Index