Wednesday: Simple Review Editor; Converting Markdown to HTML

Simple Review Editor

Writing our reviews in the textArea field can be tedious. Especially if we have really long reviews. We want to have a simple editor that will make it easier for us to write all our reviews

We will use a plug in called **flask-simplemde**, that will help us create a simple markdown editor allowing us to write our review in markdown. We will then download another python module **markdown2** that will help us convert the markdown to HTML code that we can use in our template.

```
(virtual)$ pip install flask-simplemde markdown2
```

We then need to instantiate it inside our application factory function.*app/__init__.py*

```
..........
from flask_simplemde import SimpleMDE
.....
simple = SimpleMDE()

def create_app(config_name):
    app = Flask(__name__)
    ........
    simple.init_app(app)
```

Next we need to set up configurations for the extension.

config.py

```
class Config:
    # simple mde  configurations
    SIMPLEMDE_JS_IIFE = True
    SIMPLEMDE_USE_CDN = True
```

These two extensions are required by the extension for it function. We can then connect the SimpleMDE CDN to our application template

templates/base.html

```
........

{%block styles %}
  {{super()}}
  <!--Importing the simple mde css and js files  -->
  {{ simplemde.css}}
  {{ simplemde.js }}

{% endblock %}
```

We create a `block styles` block where we use `{{super()}}` to maintain any code that was defined in that code block inside our bootstrap's *base.html* file. We then define the `{{ simplemde.css}}` and `{{ simplemde.js }}`. These two define the link to the simpleMDE CDN.

We can now load the markdown editor where we load our new review form.*templates/new_review.html*

```
        <div class="col-md-4">
            {{ wtf.quick_form(review_form) }}
            {{simplemde.load}}
        </div>
```

We load the simpleMDE editor by calling `{{simplemde.load}}`. This find the text area field and apply changes to it. Now when we launch our application we will see a simple markdown editor on our review page.

Convert Markdown To HTML

Our reviews are saved as markdown and we need to convert it to a readable HTML code that we can use in our template. We first create a route to handle viewing a single review.

main/views.py

```
........
import markdown2
.....
@main.route('/review/<int:id>')
def single_review(id):
    review=Review.query.get(id)
    if review is None:
        abort(404)
    format_review = markdown2.markdown(review.movie_review,extras=["code-friendly", "fenced-c
ode-blocks"])
    return render_template('review.html',review = review,format_review=format_review)
```

First we import the *markdown2* module which will be responsible for the conversion from markdown to HTML.

We then create the `single_review` view function that will be responsible for handling requests for a single review. It takes in the `id` property of the review. We query the database to get a single review with the the same id as the one passed in and we check if the review is available. We then call markdown2's `markdown` function that takes in two arguments.The first is the markdown that is being converted, here, we pass in the `review.movie_review` which is the movie review that is in markdown. The second argument is a lists of styling to style the HTML.

We then render the template and pass in the formatted HTML *review.html*

```
{% extends 'base.html'%}

{% block content%}
<div class="container-fluid">

  <div class="row">
    <div class="col-sm-4">
      <img src="{{review.image_path}}" class="img-responsive">
    </div>
```

```
    <div class="col-sm-8">
      {{format_review|safe}}
    </div>
  </div>
</div>
{%endblock%}
```

We use the `safe` filter to prevent jinja template from formatting our html. Then we create a link to our `single_review` view function inside our macros file.

```
<!-- Displaying reviews macro -->
{% macro displayReviews(review_list) %}

    {% for review in review_list %}

    <div class="row">

      <div class="col-xs-2 col-sm-2 col-md-4 col-lg-4">
        <h4>Author</h4>
        <p>{{review.user.username}}</p>
      </div>
      <div class="col-xs-10 col-sm-10 col-md-8 col-lg-8">
        <h2> <a href="{{url_for('main.single_review',id=review.id)}}">{{review.movie_title}}
</a> </h2>
      </div>
    </div>

    {% endfor %}
{% endmacro %}
```

Here we create an anchor tag that links to the `single_review` view function and passes in the review id.

main/forms.py

```
#.....
class ReviewForm(FlaskForm):

 title = StringField('Review title',validators=[Required()])

 review = TextAreaField('Movie review')

 submit = SubmitField('Submit')
```

Lastly, we modify our forms.py to remove the validator field in the `review` field.

We can launch our application to see everything is properly set.