УДК 004.423

DOI 10.51691/2541-8327_2023_6_14

РАБОТА С ФОРМАМИ И ШАБЛОНИЗАТОРОМ JINJA2 ФРЕЙМВОРКА FLASK В PYTHON

Халевин Т.А.

студент направления подготовки информатики и вычислительной техники,

Хакасский государственный университет имени Н.Ф. Катанова,

г. Абакан, Россия [1]

Аннотация: Данная статья рассматривает основы работы с формами во фреймворке Flask на языке Python. В статье детально описываются процессы создания и валидации форм. Также предоставляются примеры кода и объяснения по использованию шаблонизатора Jinja2 для рендеринга динамических шаблонов веб-страниц. Рассматриваются важные аспекты, такие как обработка пользовательского ввода, связывание данных с формами, а также вывод ошибок валидации.

Ключевые слова: Python, Flask, фреймворк, формы, шаблонизатор Jinja2, веб-приложения

WORKING WITH FORMS AND THE JINJA2 TEMPLATE ENGINE OF THE FLASK FRAMEWORK IN PYTHON.

Khalevin T.A.

student of computer science and computer engineering department,

N.F. Katanov Khakass State University,

Abakan, Russia

[1] Научный руководитель: Голубничий А.А. старший преподаватель кафедры ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия

Abstract: This paper describes the basics of working with forms in the Flask framework in Python. The article describes in detail the processes of creating and validating forms. It also provides code examples and explains how to use Jinja2 template engine for rendering dynamic web page templates. Important aspects such as handling user input, linking data to forms, and validation error output are covered.
Keywords: Python, Flask, framework, forms, Jinja2 template engine, web applications

В данной статье будет рассматриваться фреймворк Flask и его функциональность. Flask является одним из самых популярных фреймворков для создания веб-приложений на языке Python. Он обладает множеством инструментов для создания высококачественных веб-приложений и может быть использован как для небольших проектов, так и для крупномасштабных систем [1].

Формы являются одним из наиболее важных элементов веб-страниц, поскольку они позволяют пользователям взаимодействовать с веб-приложением и передавать данные на сервер. Формы могут использоваться для различных целей, например, для регистрации пользователей, отправки сообщений, поиска информации и т.д. [2].

Важность форм заключается в том, что они позволяют пользователям предоставлять необходимые данные на сервер, которые затем могут быть использованы для различных целей, например, для создания учетных записей пользователей, обработки платежей, отправки сообщений и т.д. Кроме того, формы позволяют пользователям задавать различные параметры и настройки для веб-приложения, такие как выбор языка, настройки безопасности, настройки конфиденциальности и т.д.

Для создания форм в Flask будет использоваться библиотека WTForms. Она предоставляет простой и удобный способ создания форм с использованием классов Python [3]. Так же важно отметить что Flask должен быть уже установлен на вашем компьютере.

Прежде чем создавать форму, необходимо установить библиотеки WTForms, email-validator и Flask-WTF (Рис. 1).

```
$ pip install WTForms
$ pip install email-validator
$ pip install Flask-WTF
```

Рисунок 1 – Установка библиотек [разработано автором]

Создадим простую форму ввода имени человека и адреса его электронной почты (Рис. 2).

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Email


class ContactForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Submit')
```

Рисунок 2 – Создание формы с помощью WTForms [разработано автором]

В приведенном выше коде создан класс ContactForm, который наследуется от класса FlaskForm. Затем определяются два поля формы name и email, каждое из которых является экземпляром класса StringField. Каждое поле также содержит список валидаторов. В данном случае используется DataRequired для обязательных полей и Email для проверки корректности адреса электронной почты. В завершение добавляется кнопка отправки формы, создав экземпляр класса SubmitField [4].

В Flask можно легко проверять данные, отправленные пользователем в форму, используя метод validate_on_submit() и методы валидации из библиотеки WTForms. В качестве примера рассмотрим функцию обработки формы, которая валидирует данные, введенные пользователем, и возвращает сообщение об успехе или ошибках (Рис. 3).

```
from flask import Flask, render_template, flash
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Email

@app.route('/', methods=['GET', 'POST'])
def contact():
    form = ContactForm()
    if form.validate_on_submit():
        name = form.name.data
        email = form.email.data
        flash('Спасибо за подтверждение, {}! Ваш email {}.'.format(name, email))
    return render_template('contact.html', form=form)
```

Рисунок 3 – Создание функции с получением данных из формы [разработано автором]

В данном примере создана функция contact(), которая обрабатывает запросы GET и POST. При GET-запросе функция возвращает шаблон HTML с формой ввода. При POST-запросе функция валидирует данные, введенные пользователем, используя метод validate_on_submit(). Если данные проходят проверку, имя и адрес электронной почты записываются в переменные и выводится сообщение об успехе, используя flash(). Затем возвращается шаблон HTML с сообщением.

Теперь необходимо объединить код представленный выше (Рис. 4).

```
from flask import Flask, render_template, flash
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired, Email

app = Flask(__name__)
app.secret_key = 'super_secret_key'

class ContactForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Submit')

@app.route('/', methods=['GET', 'POST'])
def contact():
    form = ContactForm()
    if form.validate_on_submit():
        name = form.name.data
        email = form.email.data
        flash('Спасибо за подтверждение, {}! Ваш email {}.'.format(name, email))
    return render_template('contact.html', form=form)

if __name__ == '__main__':
    app.run()
```

Рисунок 4 – Полный код приложения [разработано автором]

Так же необходимо создать папку в проекте с названием templates и создать в ней HTML документ с названием contact.html, после чего вставить в этот документ код (Рис. 5).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Contact Form</title>
  </head>
  <body>
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        <ul class="flashes">
          {% for message in messages %}
            <li>{{ message }}</li>
          {% endfor %}
        </ul>
      {% endif %}
    {% endwith %}
    <form method="POST" action="{{ url_for('contact') }}">
      {{ form.hidden_tag() }}
      <p>
        {{ form.name.label }}<br>
        {{ form.name(size=32) }}<br>
        {% if form.name.errors %}
          <ul class="errors">
            {% for error in form.name.errors %}
              <li>{{ error }}</li>
            {% endfor %}
          </ul>
        {% endif %}
      </p>
      <p>
        {{ form.email.label }}<br>
        {{ form.email(size=32) }}<br>
        {% if form.email.errors %}
          <ul class="errors">
            {% for error in form.email.errors %}
              <li>{{ error }}</li>
            {% endfor %}
          </ul>
        {% endif %}
      </p>
      {{ form.submit() }}
    </form>
  </body>
</html>
```

Рисунок 5 – Код HTML документа [разработано автором]

Далее необходимо сохранить наши файлы и запустить Flask-приложение. После запуска, перейдите по адресу http://127.0.0.1:5000 в веб-браузере, и на экране отобразится созданная форма (Рис. 6).

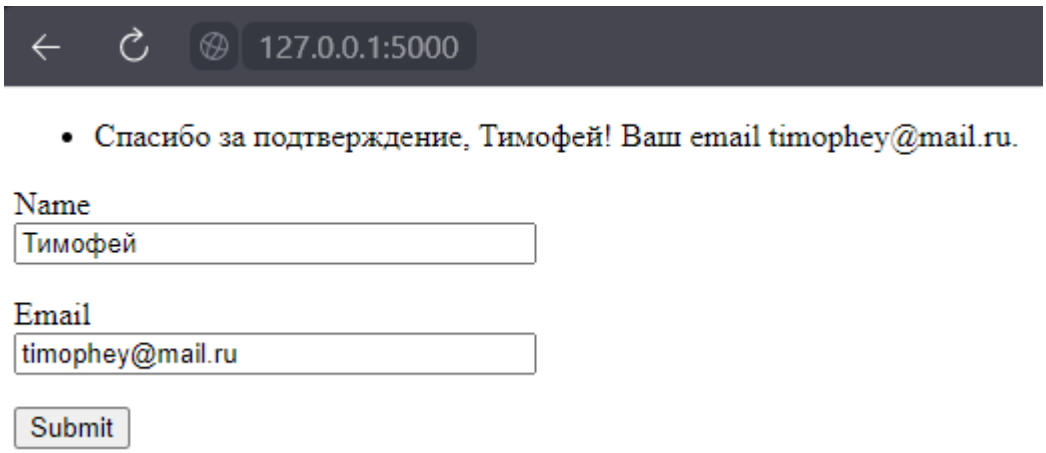

Рисунок 6 – Отображение формы в веб-браузере и вывод текста после нажатия кнопки Submit [разработано автором]

Заключение

Работа с формами является одним из важных аспектов веб-разработки. Она позволяет пользователям взаимодействовать с веб-приложением и передавать данные на сервер. Flask предоставляет нам множество инструментов для работы с формами, включая модуль WTForms. С их помощью мы можем создавать мощные и интерактивные веб-приложения, которые могут обрабатывать пользовательский ввод.

Библиографический список:

1. Доусон М. Программируем на Python [Текст] / Доусон М. – СПб.: Издательский Дом ПИТЕР, 2022. – 416 с.

2. Гэддис Т. Начинаем программировать на Python [Текст] / Т. Гэддис. – СПб.: БХВ-Петербург, 2021. – 768 с.

3. Васильев А.Н. Программирование на Python в примерах и задачах [Текст] / Васильев А.Н. – М.: Бомбора, 2021, 2022.

4. Гуриков, С.Р. Основы алгоритмизации и программирования на Python [Текст] / С.Р. Гуриков. – М.: ИНФРА-М, 2023. – 343 с.

Оригинальность 81%