ISBN 978-5-901722-87-9

УДК 656.621

СОЗДАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ НА БАЗЕ ФРЕЙМВОРКА FLASK

Юдин Александр Сергеевич[1], студент
e-mail: sashausu85@gmail.com

[1] Волжский государственный университет водного транспорта, Нижний Новгород, Россия

Аннотация. Описывается методика создания web проекта на основе языков программирования Python и Html. Описаны функции и преимущества фреймворка Flask. Создание проекта с нуля. Описание каталогов и файлов проекта. Реализация клиентской и серверной части приложения. Описывается настройка и подключение к базе данных для хранения данных пользователей и записей в проекте. Описан процесс регистрации и авторизации пользователей в системе. Реализация обновления записей в форме. Используется программное обеспечение PyCharm community edition с открытым исходным кодом, а также язык Python в качестве интерпретатора виртуального окружения версии 3.7.0 и выше.

Ключевые слова Python, Flask, фреймворк, программирование, фабрика приложений, база данных, веб-разработка.

CREATING A MANAGED SYSTEM BASED ON THE FLASK FRAMEWORK

Yudin Alexander Sergeevich[1], Student
e-mail: sashausu85@gmail.com

[1] Volga State University of Water Transport, Nizhny Novgorod, Russia

Abstract. The methodology of creating a web project based on Python and Html programming languages is described. The functions and advantages of the Flask framework are described. Creating a project from scratch. Description of catalogs and files of the project. Realization of client and server parts of the application. Configuring and connecting to a database to store user data and records in the project is described. The process of registration and authorization of users in the system is described. Realization of updating records in the form. PyCharm community edition open-source software and Python language as a virtual environment interpreter version 3.7.0 and higher are used.

Keywords: Python, Flask, framework, programming, application factory, database, web development.

Flask — это отличный фреймворк для тех, кто уже знает основы Python и хочет применить эти знания для создания веб-приложений. Этот фреймворк отличается

простотой и гибкостью, что делает его доступным для новичков и в то же время предоставляет мощные возможности для создания сложных приложений [1, 2].

Ключевыми преимуществами Flask являются:

1) Простота и минимализм: Flask предоставляет основной набор инструментов для веб-разработки, что делает его идеальным для новичков.

2) Гибкость и расширяемость: Возможность легкой интеграции с другими библиотеками и сервисами.

3) Микро - фреймворк: Flask позволяет разработчикам сохранять контроль над своим приложением, выбирая только те инструменты и библиотеки, которые им необходимы [4].

В этой статье описан простой проект с использованием данного фреймворка. В нем пользователи смогут регистрироваться, входить в свой аккаунт (рис. 1), создавать, изменять и удалять свои записи (рис. 2). Этот проект может быть установлен на локальном сервере университета для дальнейшего пользования.

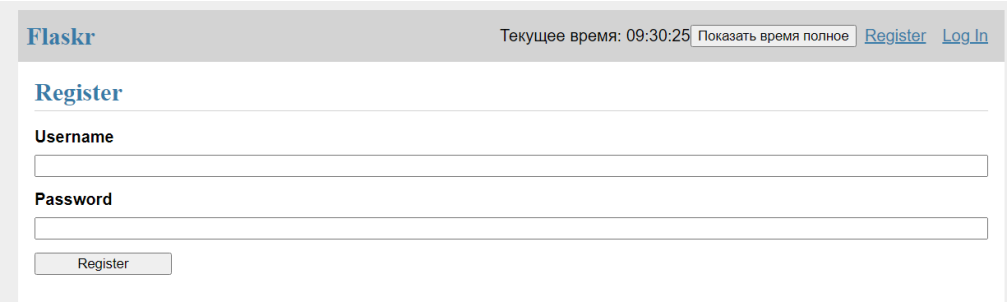

Рисунок 1 – Окно регистрации пользователя в системе

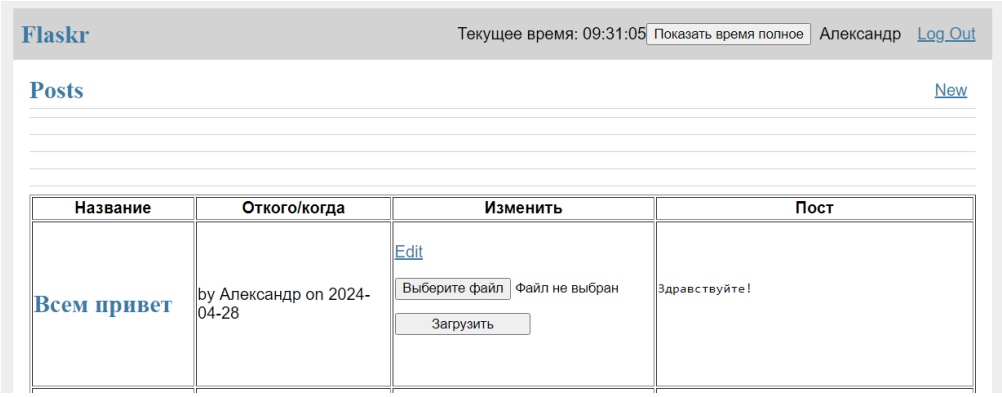

Рисунок 2 – Интерфейс работы системы

Макет проекта

В данной статье предполагается, что с этого момента вы работаете из каталога flask-tutorial. Имена файлов в верхней части каждого блока кода относятся к этому каталогу. Также подразумевается, что Вы знаете, что такое виртуальное окружение в проекте и установили его для проекта.

Приложение Flask может состоять из одного файла.

Однако по мере того, как проект становится больше, хранить весь код в одном файле становится неудобно. В проектах Python используются пакеты для организации кода в несколько модулей, которые можно импортировать при необходимости.

Каталог проекта будет содержать:

flaskr/ — пакет Python, содержащий код и файлы вашего приложения.

tests/ — каталог, содержащий тестовые модули.

.venv/ — виртуальная среда Python, в которой установлены Flask и другие зависимости.

Установочные файлы, сообщающие Python, как установить ваш проект [1, 3].

Настройка приложения

Приложение Flask является экземпляром класса Flask. Все, что касается приложения, например конфигурация и URL-адреса, будет зарегистрировано в этом классе.

Самый простой способ создать приложение Flask — создать глобальный экземпляр Flask непосредственно в верхней части кода. Хотя в некоторых случаях это просто и полезно, по мере роста проекта это может вызвать некоторые непростые проблемы.

Вместо того, чтобы создавать экземпляр Flask глобально, вы создадите его внутри функции. Эта функция известна как фабрика приложений. Любая конфигурация, регистрация и другие настройки, необходимые приложению, будут выполнены внутри функции, после чего приложение автоматически их применит как настройки по умолчанию.

Фабрика приложений

Создайте каталог flaskr и добавьте файл __init__.py. Файл __init__.py выполняет двойную функцию: он содержит фабрику приложений и сообщает Python, что каталог flaskr следует рассматривать как проект.

create_app — это функция фабрики приложения. Вы добавите его позже.

app = Flask(__name__, instance_relative_config=True) создает экземпляр Flask.

__name__ — имя текущего модуля Python. Приложению необходимо знать, где оно находится, чтобы настроить некоторые пути, и __name__ — удобный способ сообщить ему об этом.

SECRET_KEY используется Flask и расширениями для обеспечения безопасности данных. Для него установлено значение «dev», чтобы обеспечить удобное значение во время разработки, но при развертывании его следует переопределить случайным значением.

DATABASE — это путь, по которому будет сохранен файл базы данных SQLite. Он находится в app.instance_path — пути, выбранном Flask для папки экземпляра.

@app.route() создает простой маршрут, чтобы вы могли увидеть работу приложения, прежде чем переходить к остальной части статьи. Он создает связь между URL-адресом /hello и функцией, которая возвращает ответ — строку с текстом в этом случае.

Запуск приложения

Теперь вы можете запустить свое приложение с помощью команды flask. В терминале сообщите Flask, где найти ваше приложение, а затем запустите его в режиме отладки. Помните, что вы все равно должны находиться в каталоге flask-tutorial верхнего уровня, а не в пакете flaskr.

В режиме отладки интерактивный отладчик отображается всякий раз, когда страница вызывает исключение, и перезапускает сервер всякий раз, когда вы вносите изменения в код. Вы можете оставить его включенным и просто перезагрузить страницу браузера.

Определение базы данных и доступ к ней

Приложение будет использовать базу данных SQLite для хранения пользователей и сообщений. Python поставляется со встроенной поддержкой SQLite в модуле sqlite3.

SQLite удобен тем, что не требует настройки отдельного сервера базы данных и встроен в Python. Однако если одновременные запросы попытаются выполнить запись в базу данных одновременно, они будут замедляться, поскольку каждая запись происходит последовательно. Маленькие проекты этого не заметят. Когда вы станете расширяться, вы, возможно, захотите переключиться на другую базу данных.

Подключение к базе данных

Первое, что нужно сделать при работе с базой данных SQLite (и большинством других библиотек баз данных Python), — это создать к ней соединение. Любые запросы и операции выполняются с использованием соединения, которое закрывается после завершения работы.

В веб-приложениях это соединение обычно привязано к запросу. Он создается в какой-то момент при обработке запроса и закрывается до отправки ответа.

g — специальный объект, уникальный для каждого запроса. Он используется для хранения данных, к которым могут получить доступ несколько функций во время запроса. Соединение сохраняется и используется повторно вместо создания нового соединения, если get_db вызывается второй раз в том же запросе.

current_app — еще один специальный объект, указывающий на приложение Flask, обрабатывающее запрос. Поскольку вы использовали фабрику приложений, при написании остальной части кода объект приложения отсутствует. get_db будет вызываться, когда приложение создано и обрабатывает запрос, поэтому можно использовать current_app.

sqlite3.connect() устанавливает соединение с файлом, на который указывает конфигурационный ключ БАЗЫ ДАННЫХ. Этот файл еще не обязательно должен существовать и не будет, пока вы позже не инициализируете базу данных.

sqlite3.Row сообщает соединению, что оно должно возвращать строки, которые ведут себя как dicts. Это позволяет получить доступ к столбцам по имени.

Создание таблицы базы данных

В SQLite данные хранятся в таблицах и столбцах. Их необходимо создать, прежде чем вы сможете хранить и извлекать данные. Flaskr будет хранить пользователей в таблице пользователей, а сообщения — в таблице сообщений. Создайте файл с командами SQL, необходимыми для создания пустых таблиц:

Инициализация файла базы данных

Теперь, когда init-db зарегистрирован в приложении, его можно вызвать с помощью команды flask, аналогичной команде run с предыдущей страницы.

Чертежи и виды

Flaskr будет иметь два проекта: один для функций аутентификации и один для функций сообщений в блоге. Код для каждого проекта будет помещен в отдельный модуль. Поскольку блог должен знать об аутентификации, сначала вы должны создать регистрацию пользователей.

Регистрация пользователя

Когда пользователь посещает URL-адрес /auth/register, представление реестра возвращает HTML-код с формой для заполнения. Когда они отправят форму, она проверит введенные данные и либо снова отобразит форму с сообщением об ошибке, либо создаст нового пользователя и перейдет на страницу входа.

Проект блога

В блоге должны быть перечислены все сообщения, разрешено вошедшим в систему пользователям создавать сообщения, а автору сообщения должно быть разрешено редактировать или удалять его.

При реализации каждого представления оставляйте работающим сервер разработки. Сохранив изменения, попробуйте перейти по URL-адресу в браузере и протестировать их.

Создание записей в форме

Представление создания работает так же, как представление регистра аутентификации. Либо отображается форма, либо опубликованные данные проверяются и сообщение добавляется в базу данных, либо отображается ошибка.

Декоратор login_required, который вы написали ранее, используется в представлениях блога. Пользователь должен войти в систему, чтобы посетить эти представления, в противном случае они будут перенаправлены на страницу входа.

Обновление записей в форме

Представления обновления должны будут получить сообщение по идентификатору и проверить, соответствует ли автор вошедшему в систему пользователю.

Представления создания и обновления выглядят очень похоже. Основное отличие состоит в том, что представление обновления использует объект post и запрос UPDATE вместо INSERT. С помощью рефакторинга вы можете использовать одно представление и шаблон для обоих действий.

Список литературы:

1. Официальный сайт фреймворк Flask – URL: https://flask.palletsprojects.com/en/latest/ (дата обращения: 30.04.2024)

2. Официальный сайт программного обеспечения Python – URL: https://www.python.org/ (дата обращения: 30.04.2024)

3. Официальный сайт программного обеспечения Jetbrains – PyCharm – URL: https://www.jetbrains.com/pycharm/download/?section=windows (дата обращения: 30.04.2024)

4. Корпоративный блог Hubr.com – URL: https://habr.com/ru/articles/783574/ (дата обращения: 30.04.2024)