

Разработка Web приложений с Python и Flask

Дмитрий Кушцов
2021

Оглавление

| | | |
|-------|--|----|
| 1 | Компоненты | 7 |
| 1.1 | Среда окружения | 7 |
| 1.2 | Безопасность | 8 |
| 1.2.1 | CSRF | 8 |
| 1.2.2 | Аутентификация и авторизация | 10 |
| 1.2.3 | Шифрование данных с SSL | 10 |
| 1.3 | Базы данных | 10 |
| 1.3.1 | MySQL и SQLAlchemy | 10 |
| 1.3.2 | MongoDB и NoSQL | 10 |
| 1.4 | Веб формы | 10 |
| 1.5 | Jinja | 10 |
| 1.6 | Blueprints: Структурируем большое приложение | 10 |
| 1.7 | Bootstrap | 10 |
| 1.8 | Клиентский код: Vue.js | 10 |
| 2 | Разрабатываем блог | 11 |

Веб приложения сегодня являются одним из самых распространённых способов создания сервисов, которые используются миллионами пользователей. Веб приложения легко обновлять, переносить, и распространять - они не требуют установки специальных средств на рабочей машине пользователей, кроме веб браузера.

В этой книге мы рассмотрим как создавать большие и структурированные веб приложения используя Python и библиотеку Flask. Мы рассмотрим такие вещи как безопасность веб приложений, базы данных, веб формы и структурирование больших веб приложений.

Данная книга предназначена для начинающих веб программистов, а также для всех тех, кто желает познакомиться с веб программированием. Мы предполагаем, что читатель уже ознакомился с нашей книгой об основах Python и алгоритмах. Данную книгу можно скачать бесплатно здесь [?].

ГЛАВА 1

Компоненты

1.1 Среда окружения

Часто на рабочей машине может быть запущено несколько проектов одновременно. Каждый проект должен иметь свои зависимости и установленные библиотеки. Для того, чтобы не было путаницы в библиотеках на рабочей машине, для разработки, устанавливают виртуальное окружение и устанавливают нужные библиотеки.

На дистрибутиве Linux Ubuntu 20.04 (мы предполагаем наличие Python 3.6) среда может быть установлена следующим образом:

```
$ python3 -m venv myproject
```

После нужно активировать виртуальную среду разработки:

```
$ source myproject/bin/activate
```

Далее можно устанавливать необходимые библиотеки и они не будут пересекаться с другими проектами:

```
$ pip3 install pycryptodome
```

1.2 Безопасность

Безопасность в веб приложениях является одним из основных вопросов. Популярность веб приложений и их повсеместность накладывает определённые требования к безопасности. В данной главе мы познакомим читателя с такими атаками как Cross Site Request Forgery (CSRF), неаутентифицированный и неавторизованный доступ к ресурсам, а также рассмотрим методы защиты от таких атак. И конечно, мы затронем тему шифрования канала от пользователя до веб сервера с помощью Secure Socket Layer (SSL).

1.2.1 CSRF

Очень часто можно послать запрос на сайт замаскированный как сторонний и произвести какую либо транзакцию так, чтобы пользователь этого не заметил. Например, пусть пользователь авторизовался на вашем сайте, получил куки и не вышел из системы. Позже злоумышленник может прислать пользователю на почту картинку с изображением, например, кошечки и попросил перейти по ссылке. Ссылка же на самом деле ведёт к вашему сайту и автоматически пошлёт все куки файлы с запросом.

Это опасно тем, что если система не защищена от CSRF атак, то запрос выполнится и атакующий сможет изменить содержимое базы данных. Для того чтобы этот тип атаки не смог быть реализован, необходимо в форме HTML вставлять некий токен безопасности. Вместе с тем, тот же токен необходимо хранить в зашифрованном виде в куке файле. Если на сервер придет куки файл, и токен в нем будет отличаться от токена, полученного в форме, то запрос стоит отвергнуть, так как он не безопасный.

Приведём пример того, как можно использовать CSRF защиту в Flask приложении:

```
from flask_wtf.csrf import CSRFProtect

csrf = CSRFProtect(app)
```

В веб форме прописываем скрытое поле, которое будет содержать наш CSRF токен:

```
<form method="post">
    {{ form.csrf_token }}
</form>
```

Если вы используете jQuery AJAX запросы то можно с запросом посылать и токен:


```

<script type="text/javascript">
    var csrf_token = "{{ csrf_token() }}";

    $.ajaxSetup({
        beforeSend: function(xhr, settings) {
            if (!/^(GET|HEAD|OPTIONS|TRACE)$/i.test(settings.type) && !
                this.crossDomain) {
                xhr.setRequestHeader("X-CSRFToken", csrf_token);
            }
        }
    });
</script>

```

Когда проверка верности токена будет неудачной Flask выбросит ошибку `CSRFError`. По умолчанию, Flask вернет HTTP с кодом 400 и объяснением причины ошибки. Если вы хотите отправить своё сообщение об ошибке то стоит зарегистрировать соответствующий обработчик ошибки:

```

from flask_wtf.csrf import CSRFError

@app.errorhandler(CSRFError)
def handle_csrf_error(e):
    return render_template('csrf_error.html', reason=e.description),
        400

```

Стоит заметить CSRF защита требует секретный ключ для подписи токена. По умолчанию Flask будет использовать `SECRET_KEY` переменную для этих целей. Если же вы захотите использовать отдельный ключ для этих целей, то можно установить переменную `WTF_CSRF_SECRET_KEY`.

<https://towardsdatascience.com/how-to-secure-your-machine-learning-app-with-csrf-protection-506c3383f9e5>

1.2.2 Аутентификация и авторизация

1.2.3 Шифрование данных с SSL

1.3 Базы данных

1.3.1 MySQL и SQLAlchemy

1.3.2 MongoDB и NoSQL

1.4 Веб формы

1.5 Jinja

1.6 Blueprints: Структурируем большое приложение

1.7 Bootstrap

1.8 Клиентский код: Vue.js

ГЛАВА 2

Разрабатываем блог

Литература
