

ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Структурировать

practicum.yandex.ru

РЕКЛАМА · 16+ · Я · ⋮

Курс «Специалист по Data Science». Начните с нуля.

Обучаем специалистов в Data Science с нуля. 20 часов практики – бесплатно.

Узнать больше

Страница 3. / Jinja2 - движок шаблонов для Python

Модуль Jinja2 – быстрый и удобный язык шаблонов для Python, созданный по образцу шаблонов Django. Он быстр, т.к. широко используется и безопасен благодаря дополнительной среде выполнения изолированных шаблонов.

Преимущества Jinja2:

- экранирование HTML для предотвращения XSS.
- поддержка макросов.
- генерация оптимального кода Python (можно отключить при отладке).
- исключения точно указывают на неправильную строку в шаблоне.
- много встроенный фильтров.
- методов стандартных типов Python в шаблонах.
- синтаксис Python в шаблонах.

Установка Jinja2 в виртуальное окружение:

для своего бизнеса на hh.ru

Найти

Реклама. ООО «Хэдхантер». Подробнее на сайте hh.ru

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# ставим модуль Jinja2
(VirtualEnv):~$ python -m pip install -U Jinja2
```

## Инициализация движка шаблонов Jinja2 в Python.

Модуль Jinja использует центральный объект, называемый шаблоном `jinja2.Environment()`. Экземпляры этого класса используются для хранения конфигурации и глобальных объектов, а также для [загрузки шаблонов](#) из файловой системы или других мест. Даже если создавать шаблоны из строк с помощью конструктора класса `jinja2.Template()`, среда Environment создается автоматически, только она будет совместно используемая.

Большинство приложений создают один объект Environment при инициализации приложения и используют его для загрузки шаблонов. Однако в некоторых случаях полезно иметь несколько сред рядом, если используются разные конфигурации.

Самый простой способ настроить Jinja для загрузки шаблонов для приложения выглядит примерно так:

```
from jinja2 import Environment, PackageLoader, select_autoescape
env = Environment(
    loader=PackageLoader('yourapplication', 'templates'),
    autoescape=select_autoescape(['html', 'xml'])
)
```

Это создаст шаблонную среду с настройками по умолчанию и загрузчиком loader, который будет искать шаблоны в папке шаблонов templates внутри пакета python yourapplication. [Доступны разные загрузчики](#), также можно написать свой собственный, если необходимо загружать шаблоны из базы данных или других ресурсов. Код примера, так же, определяет автоматическое экранирование файлов с расширениями .html и .xml.

Чтобы загрузить шаблон из среды env, которая определены в примере выше, нужно просто вызвать метод env.get\_template(), который затем возвращает загруженный шаблон [Template](#):

```
template = env.get_template('mytemplate.html')
```

Чтобы отобразить его с некоторыми переменными, просто вызовите метод Template.render():

```
print(template.render(the='variables', go='here'))
```

Использование [загрузчика шаблонов](#) вместо передачи обычных [строк](#) в Template или Environment.from\_string() имеет несколько преимуществ. Помимо того, что загрузчик намного проще в использовании, он также позволяет [наследование шаблонов](#).

При использовании загрузчика шаблонов автоматическое экранирование будет включено по умолчанию из соображений безопасности.

В базовом режиме экранирование включено по умолчанию. В режиме [автоматического экранирования](#), а не полагаться на значение по умолчанию.

При использовании загрузчика шаблонов в шаблонах.

Jinja2 поддерживает экранирование Python. Допустимые идентификаторы могут быть любой комбинацией символов Юникода, за исключением пробелов и специальных символов. Идентификаторы могут содержать символы подчеркивания и имеют слегка измененный синтаксис идентификатора. Фильтры могут содержать символы подчеркивания. Например, вполне допустимо добавить метод в фильтр и вызвать его как .unicode. Регулярное выражение для идентификаторов фильтров: [a-zA-Z\_][a-zA-Z0-9\_]\*(\.[a-zA-Z\_][a-zA-Z0-9\_]\*)\*.

## Базовое использование движка шаблонов Jinja2.

В этой статье мы рассмотрим базовое использование в Python API для шаблонов Jinja.

Самый простой способ использовать шаблон и отрендерить его - использовать класс [jinja2.Template\(\)](#). Такой способ работы не требует загрузки шаблонов, а загружаются не из строк, а из файловой системы или другого источника данных:

```
>>> from jinja2 import Template
>>> tpl = Template('Сумма чисел {{ a }} и {{ b }} равна {{ a + b }}')
>>> tpl.render(a=5, b=2)
# 'Сумма чисел 5 и 2 равна 7'
```

## Пример разбора шаблона с циклом.

```
>>> import jinja2
# шаблон с циклом
>>> tpl = """
... {{ '-' * title|length }}
... {% for n, user in enumerate(users, 1) %}
...   {{ n }}. {{ user.name }} - должность: {{ user.status }}, оклад: ${{ user.salary }}
... {% endfor %}
... """
# собираем данные для шаблона
>>> content = {}
>>> content['title'] = 'Итерация по пользователям'
>>> content['users'] = []
>>> content['users'].append({'name': 'Маша', 'status': 'Менеджер', 'salary': 1500})
>>> content['users'].append({'name': 'Света', 'status': 'Дизайнер', 'salary': 1000})
>>> content['users'].append({'name': 'Игорь', 'status': 'Программист', 'salary': 2000})
# В словаре передаем в шаблон функцию Python
>>> content['enumerate'] = enumerate
# Смотрим, что получилось
>>> print(jinja2.Template(tpl, trim_blocks=True).render(content))
# Итерация по пользователям
# -----
# 1. Маша - должность: Менеджер, оклад: $1500
# 2. Света - должность: Дизайнер, оклад: $1000
# 3. Игорь - должность: Программист, оклад: $2000
```

## Загрузка шаблонов из файловой системы.

Сохраним шаблон из предыдущего примера в директорию ~/temp/main.txt.

```
{% from 'main.txt' import user %}
{{ user.name }}
{# Добавим условие #}
```

```
{% if title %}
{# Если существует переменная `title`, то будем ее подчеркивать #}
{{ '-' * title|length }}
{% endif %}

{# ... #}
{{ users, 1) %}
...
лжность: {{ user.status }}, оклад: ${{ user.salary }}
```

Основная задача — как работает с сохраненным шаблоном main.txt. Для понимания, что происходит, код снабжен комментариями.

```
import os
import jinja2

# Загружаем шаблоны из файловой системы
# Сохраненный шаблон 'main.txt'
loader = jinja2.FileSystemLoader('temp')
env = jinja2.Environment(loader=loader, trim_blocks=True)

# Загружаем шаблон 'main.txt'
tpl = env.get_template('main.txt')
# рендерим шаблон в переменную `result`
result = tpl.render(content)
# Сохраним получившийся текст
with open('result.txt', 'w') as fp:
    fp.write(result)

# Прочитаем записанный файл
with open('result.txt', 'r') as fp:
    print(fp.read())

# Итерация по пользователям
# -----
# 1. Маша - должность: Менеджер, оклад: $1500
# 2. Света - должность: Дизайнер, оклад: $1000
# 3. Игорь - должность: Программист, оклад: $2000
```



### Содержание раздела:

Вверх

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Класс Environment\(\) модуля jinja2](#)
- [Класс Template\(\) модуля jinja2](#)

РЕКЛАМА

hh

Ищите работников для своего бизнеса на hh.ru

Найти

DOCS-Python.ru™, 2023 г.

Реклама ООО «Хэдхантер». Подробнее на сайте hh.ru.

0+

[онах Jinja](#)

[ja2](#)

[шаблонов модулем jinja2](#)

[a2](#)

[аблонов jinja2](#)

[тесты для jinja2](#)

[функции модуля jinja2](#)

[ах jinja2](#)

[ja2](#)

[\\_jinja2](#)

[ах jinja2](#)

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs\\_python\\_ru](#)