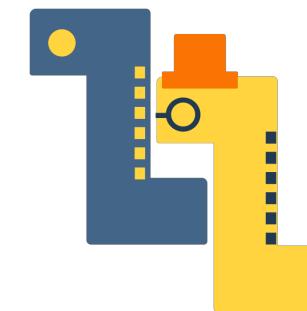




# 自由的Flask

李辉

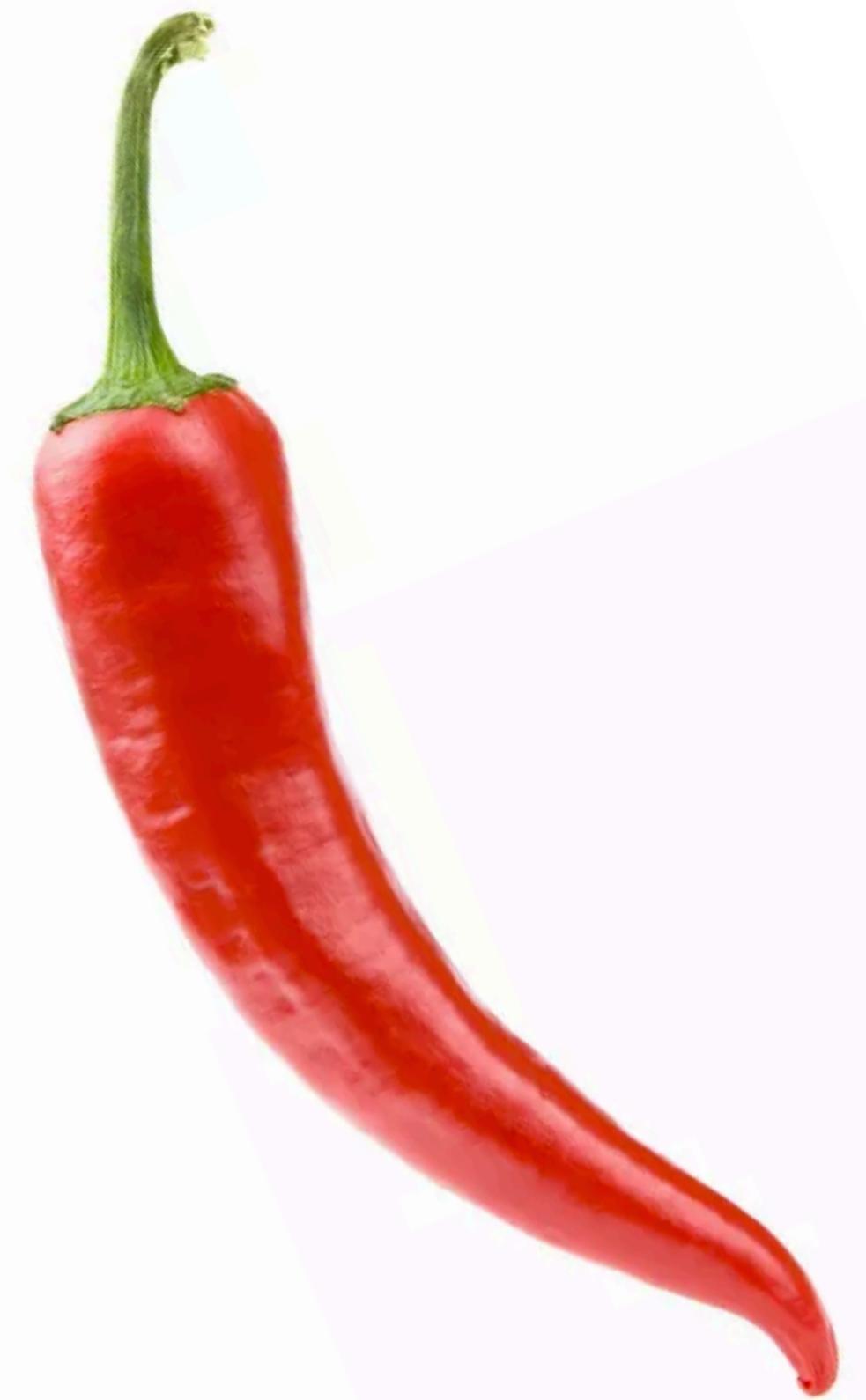
PyCon China 2018 北京





?

powder horn / flask



No



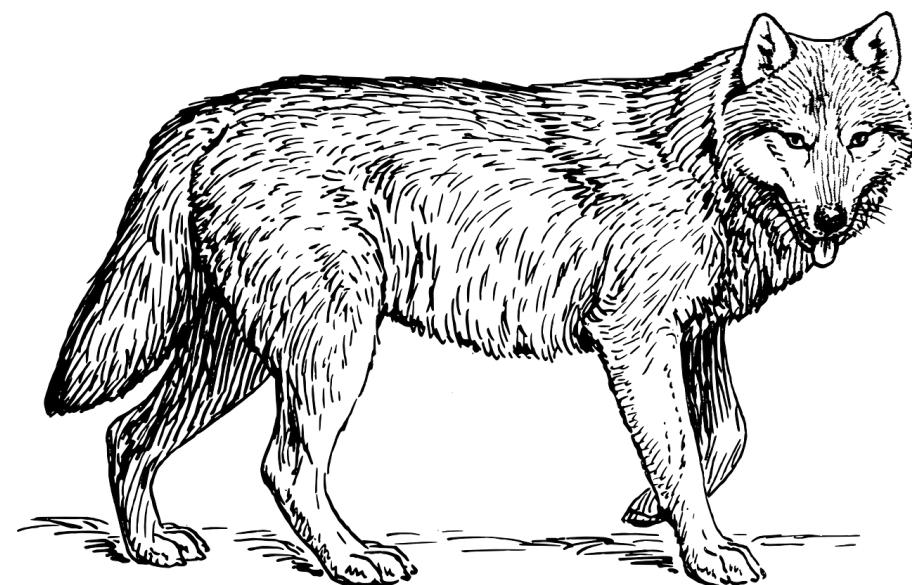
Yes

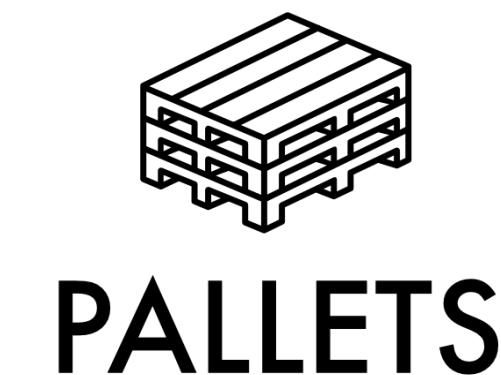
# Werkzeug

['vɛk'tsɔɪk]

# About Me

- Hello! 我是李辉 (Grey Li)
- 知乎专栏 “Hello, Flask!” 作者
- 《Flask Web开发实战》 作者
- Flask开发团队 (Pallets Team) 成员
- 多个 Flask 扩展的维护者





# About Flask

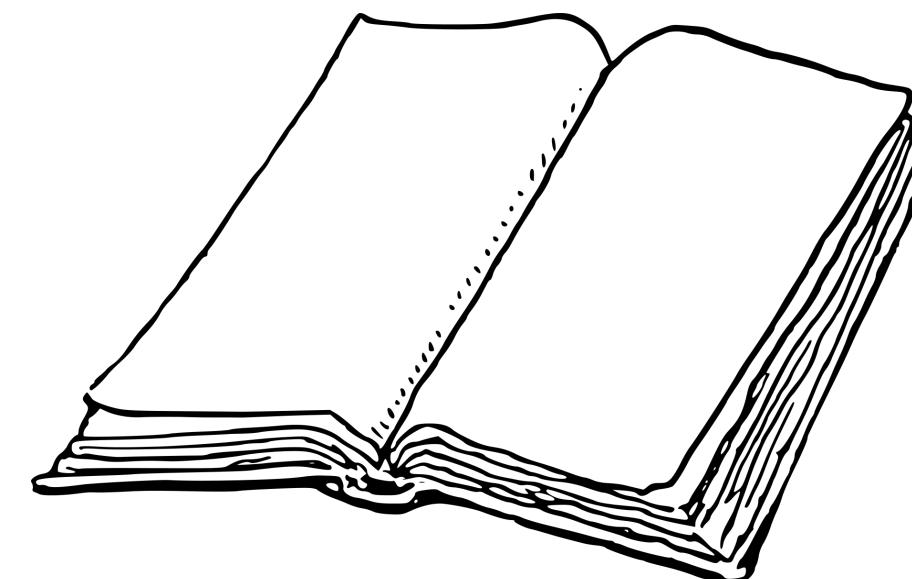
- 由 Armin Ronacher 在 2010 年创建
- 目前由 Pallets Team 维护
- 5个依赖: Werkzeug、Jinja2、Click.....
- 今年 4 月发布了 1.0 稳定版
- 在 GitHub 上有近 4 万 star

~~zer0ver~~



# Contents

- What's new in Flask 1.0?
- Web 程序和 Web 框架
- 自由的功能扩展
- 自由的路由定义
- 自由的项目结构





“大快所有人心的大事，大了又大。”

– Steve Jobs

## Chapter 0

# What's new in Flask 1.0?

更完善的 CLI 系统

# Development Server

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

```
$ export FLASK_APP=hello
$ export FLASK_ENV=development
$ flask run
```

# Application Discovery

1. 不设置 (app.py / wsgi.py)
2. 设为包含程序实例或工厂函数的模块
  - FLASK\_APP=hello
  - FLASK\_APP=project/hello
  - FLASK\_APP=hello.web
3. 直接指定程序实例或工厂函数
  - FLASK\_APP=hello:myapp
  - FLASK\_APP=hello:generate\_app
  - FLASK\_APP="hello:create\_app('dev')"

# python-dotenv Support

```
$ pip install python-dotenv
```

```
$ nano .flaskenv
```

```
FLASK_APP=hello
FLASK_ENV=development
```

```
$ flask *
```

更丰富的测试支持

# 测试 JSON API

```
def test_hello_api():
    client = app.test_client()
    rv = client.post('/api/hello', json={
        'name': 'Grey'
    })
    json_data = rv.get_json()
    assert json_data['message'] == 'Hello, Grey!'
```

# 测试 Flask 命令

```
from myapp import hello_command

def test_hello_command():
    runner = app.test_cli_runner()

    result = runner.invoke(hello_command)
    assert 'Hello!' in result.output
```

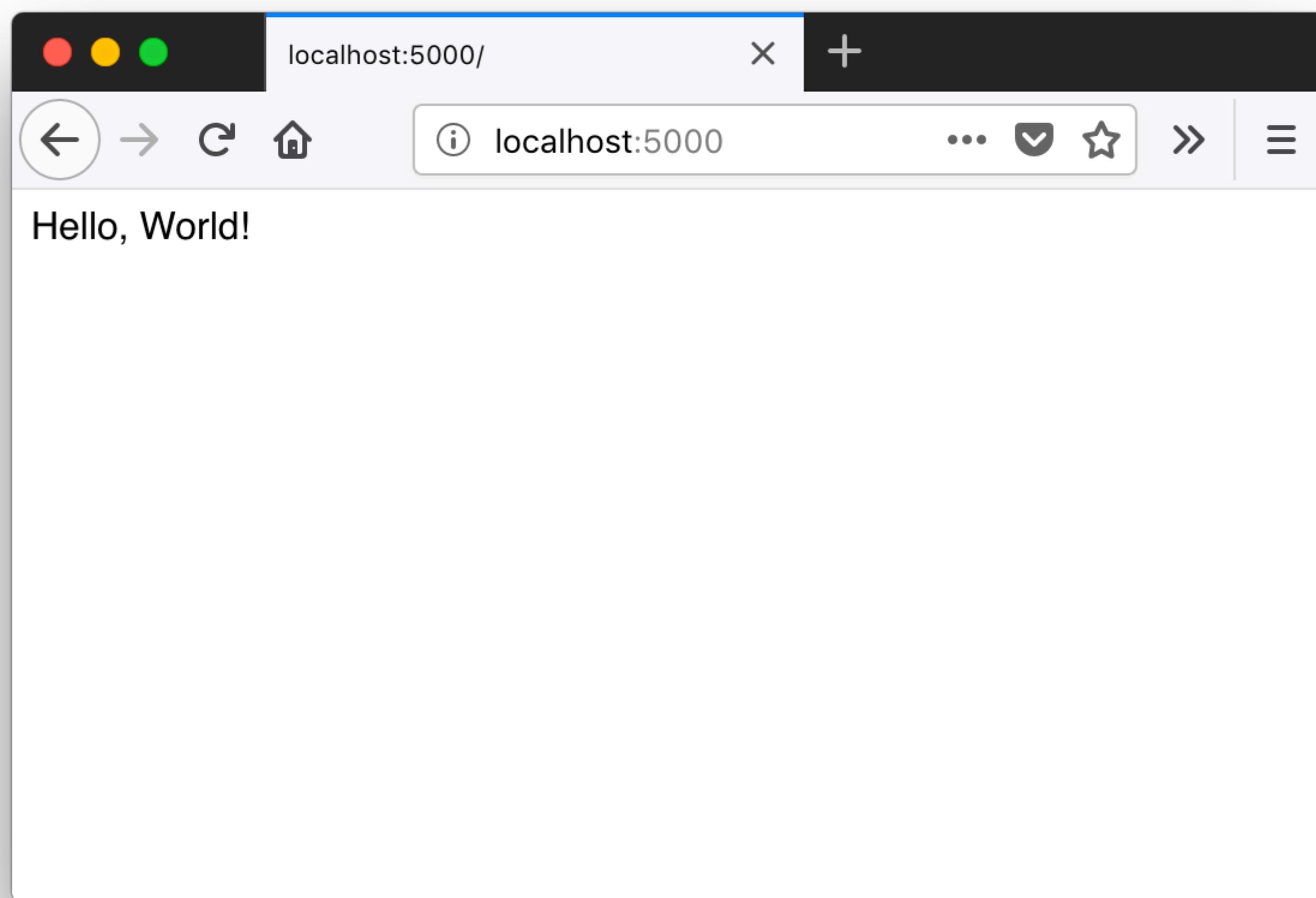
etc.

## Chapter 1

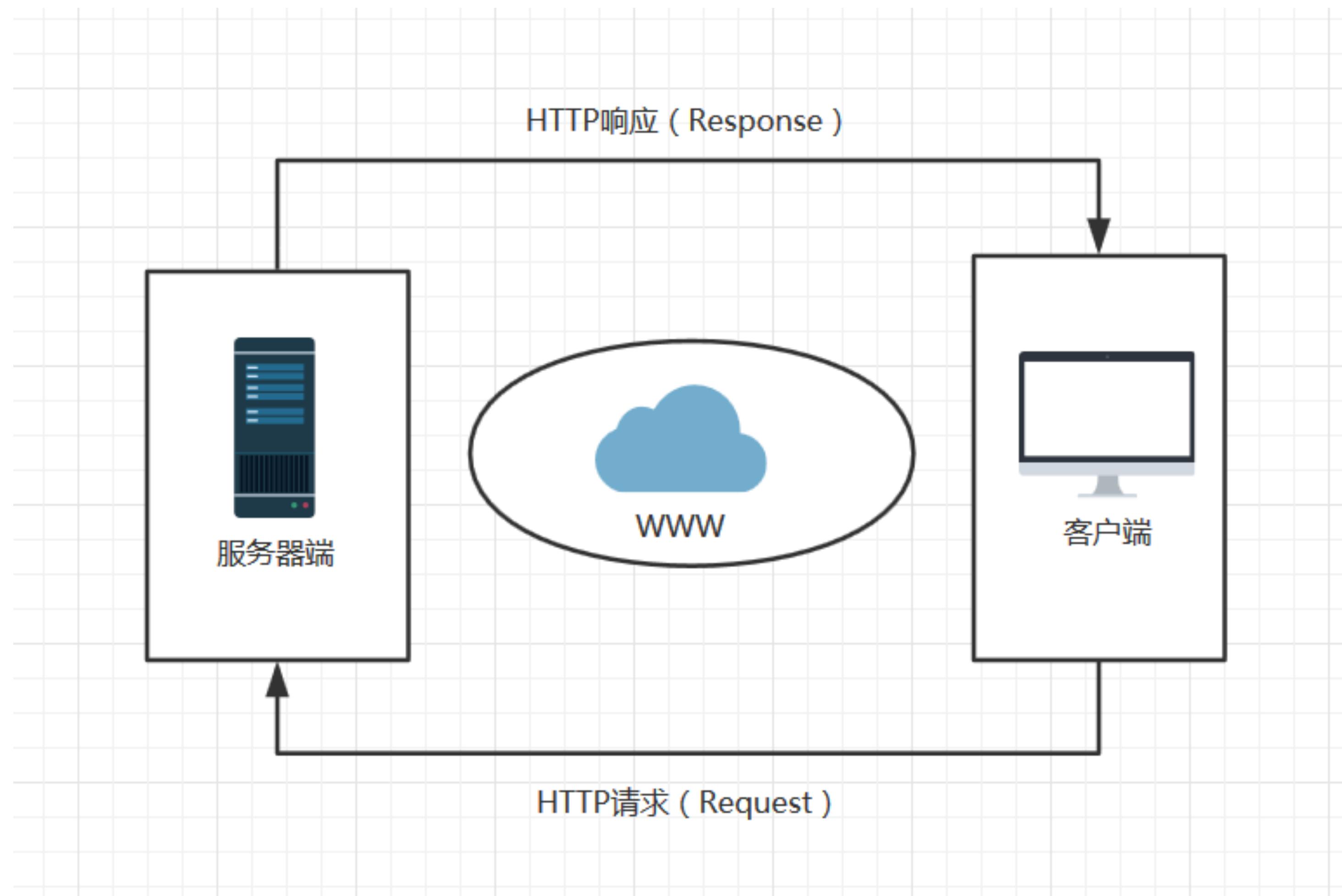
# Web 程序和 Web 框架

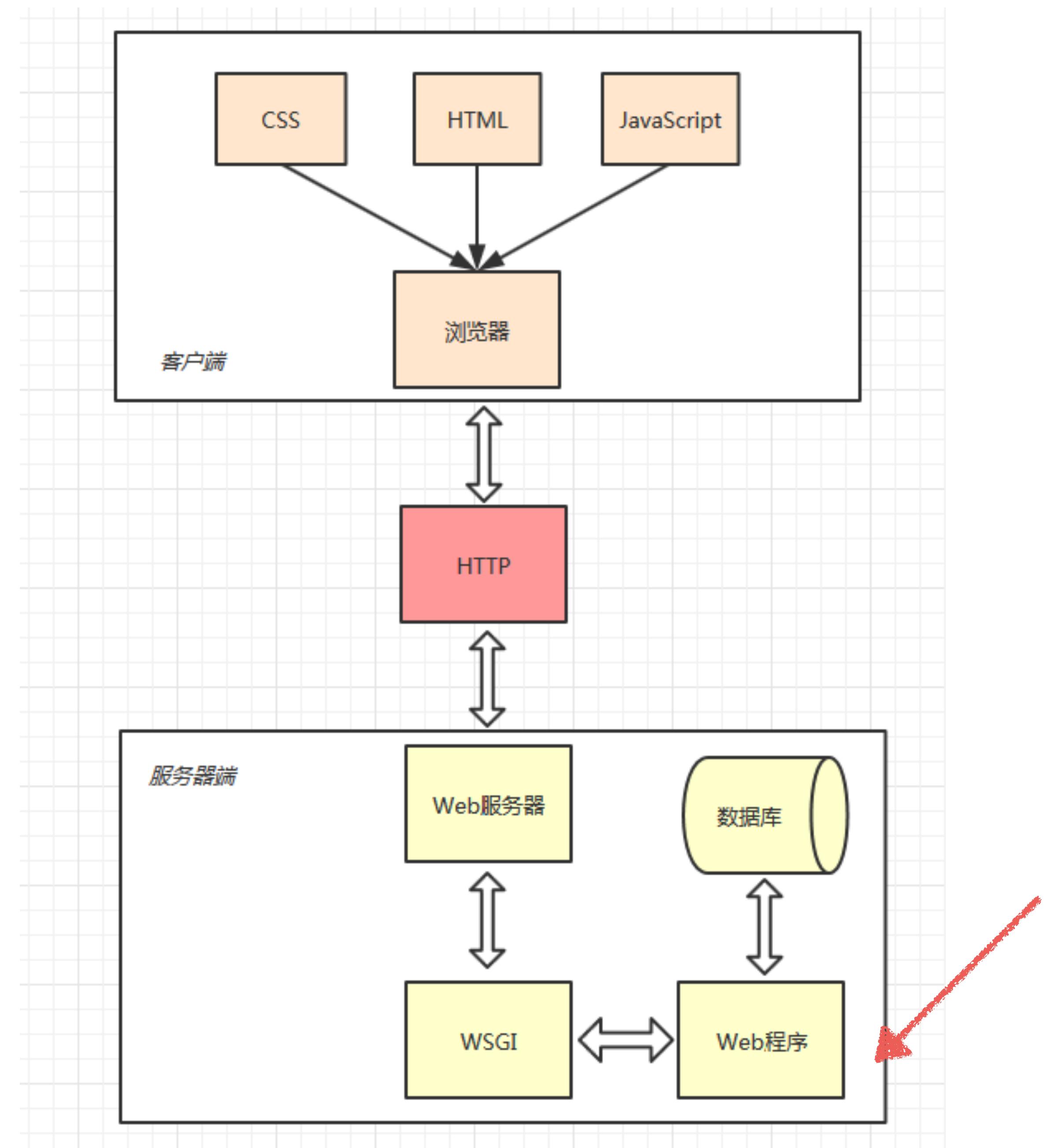
# 编写 Hello World 程序

- 用户访问 / 时，返回一行“Hello, World!”问候。

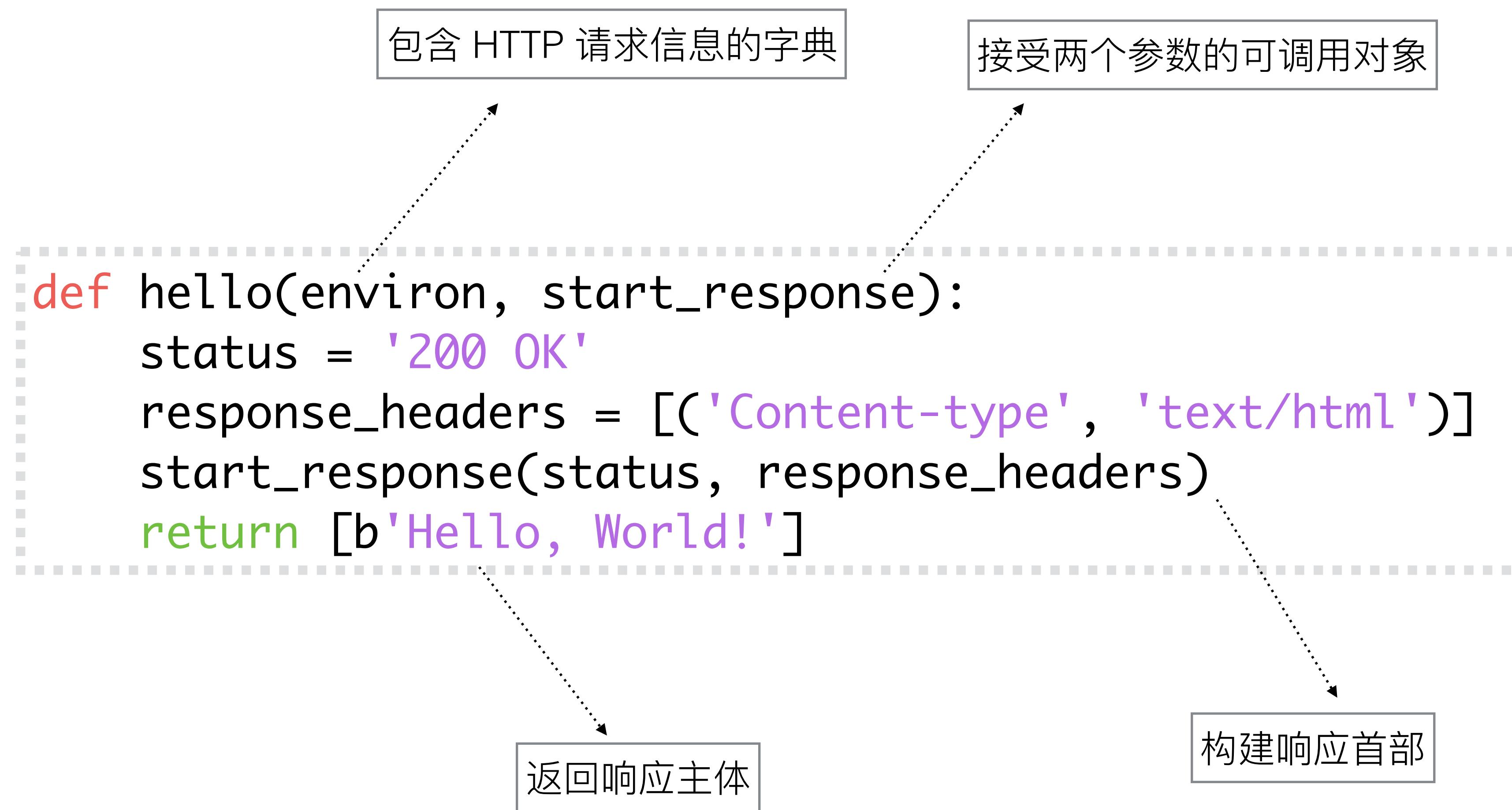


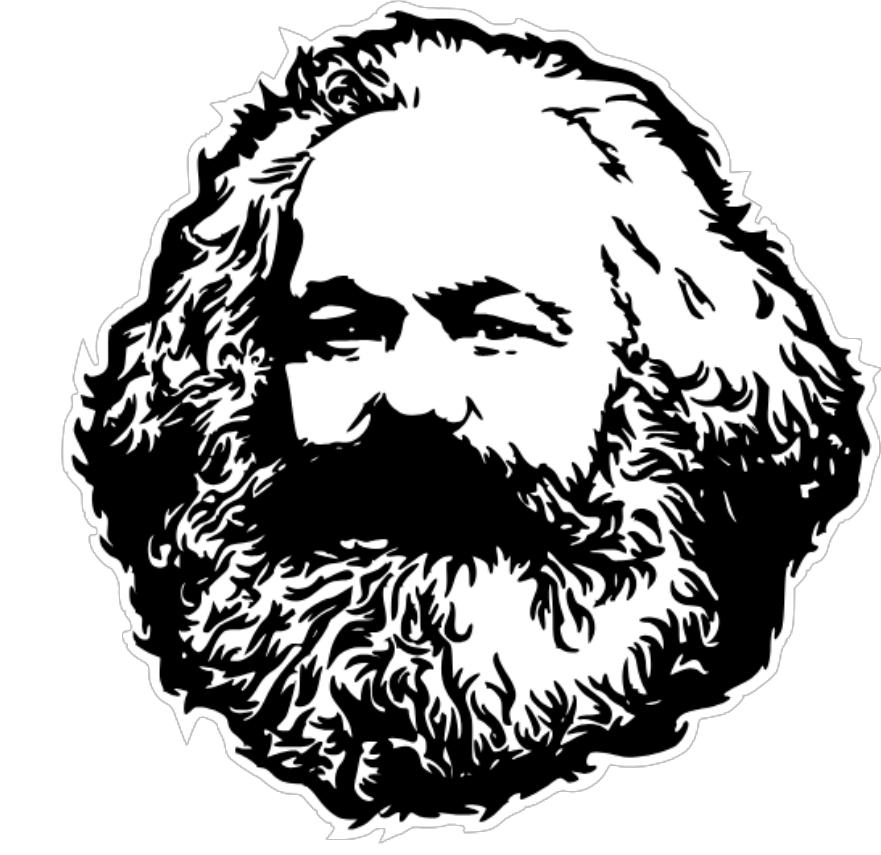
# “请求 - 响应”循环





# WSGI Application





“Web 框架把 Web 开发者从繁琐的底层实现中解放出来，从而有更多的时间思考午饭吃什么。”

– Karl Marx

# Flask

```
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def hello():  
    return 'Hello, World!'
```

▼ 创建程序实例

▶ 绑定对应的url规则

▶ 返回响应主体

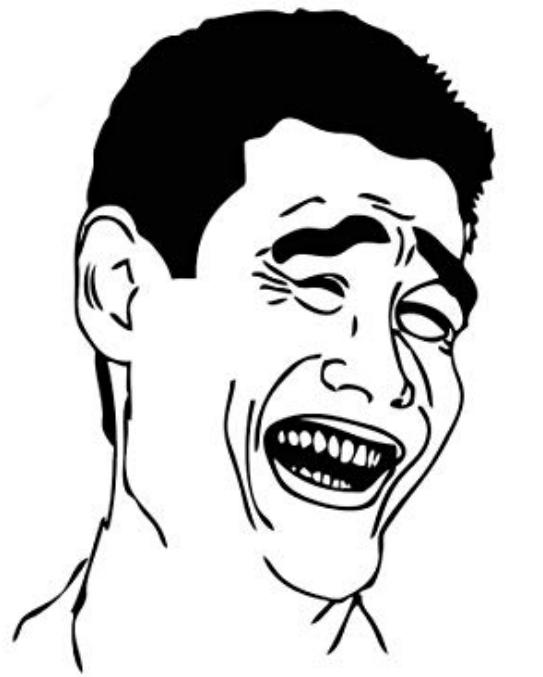
# Django

```
from django.conf.urls import url
from django.http import HttpResponseRedirect

DEBUG = True
SECRET_KEY = '4l0ngs3cr3tstr1ngw3lln0ts0l0ngw41'
ROOT_URLCONF = __name__

def hello(request):
    return HttpResponseRedirect('Hello, World!')

urlpatterns = [
    url(r'^$', hello),
]
```



“没有对比就没有伤害。”

– Yao

## Chapter 2

# 自由的功能扩展

# 微核心 + 扩展

- 保留核心 (**一个初始状态的芭比娃娃**)
- 通过扩展与第三方库进行集成 (**附送几百套衣服**)
- 打造你自己的 Web 框架

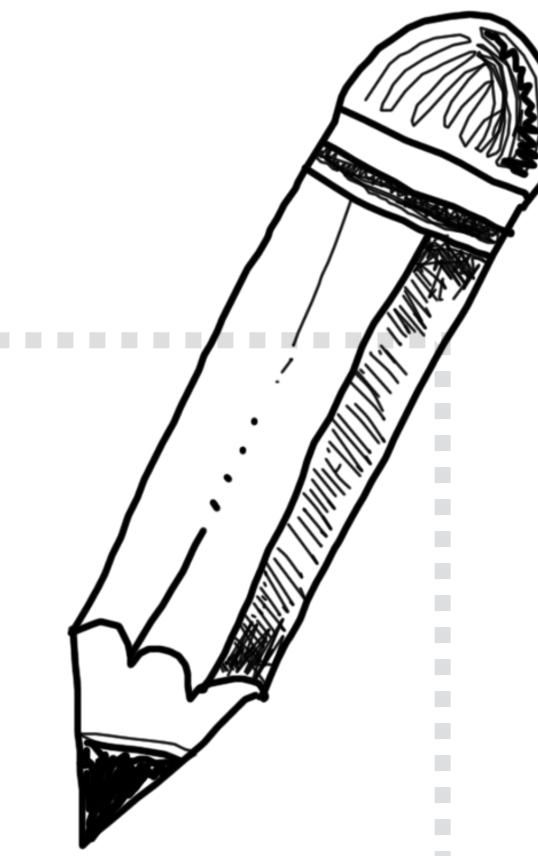


# 高度可自定义

```
from flask import Flask

class SmileFlask(Flask):
    def make_response(self, rv):
        body = rv + ' :)'
        response = self.response_class(body)
    return response

app = SmileFlask(__name__)
```



## Chapter 3

自由的路由定义

# 方式 1

函数 + 单独注册

```
from flask import Flask  
app = Flask(__name__)  
  
def hello():  
    return 'Hello, Flask!'  
  
app.add_url_rule('/', hello)
```

# 方式 2

## 装饰器

```
from flask import Flask  
app = Flask(__name__)  
  
@app.route('/')  
def hello():  
    return 'Hello, Flask!'
```

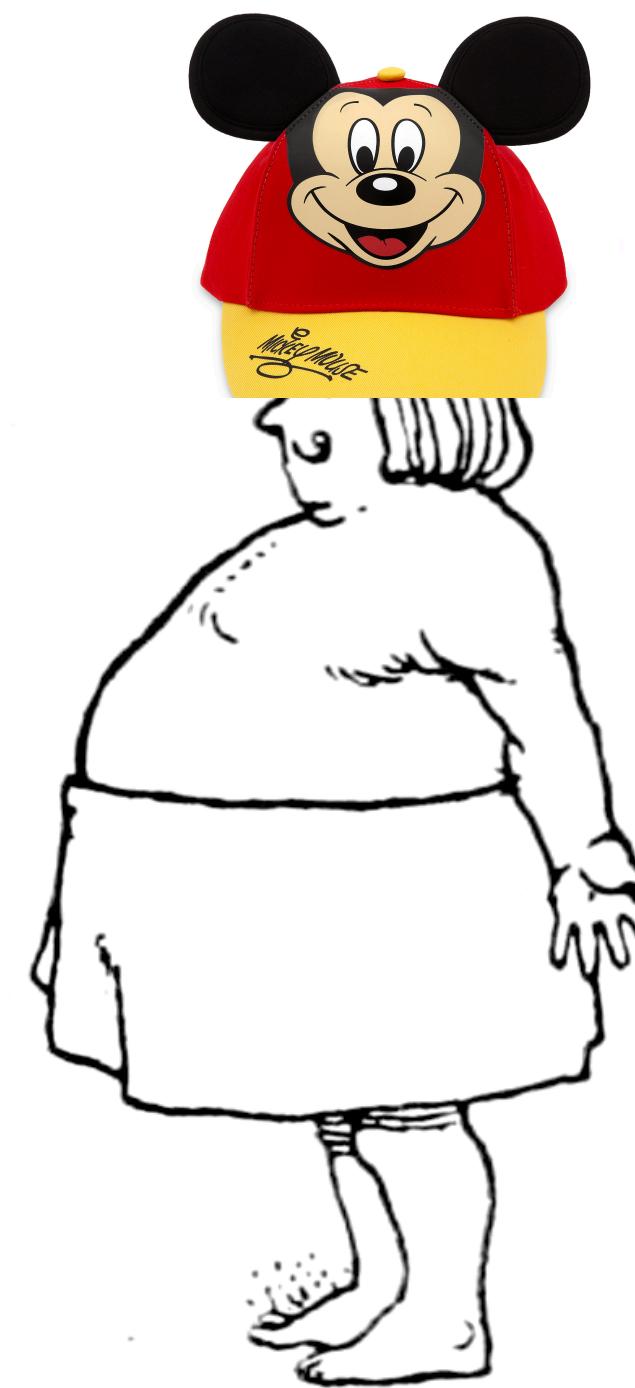


“装饰器是一个绝妙的好主意，我们应该把它发扬光大。”

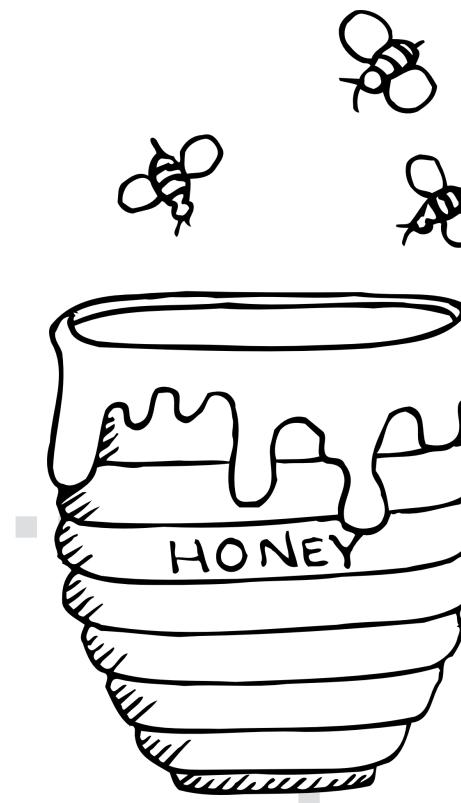
– Zen of Flask

# 百变装饰器

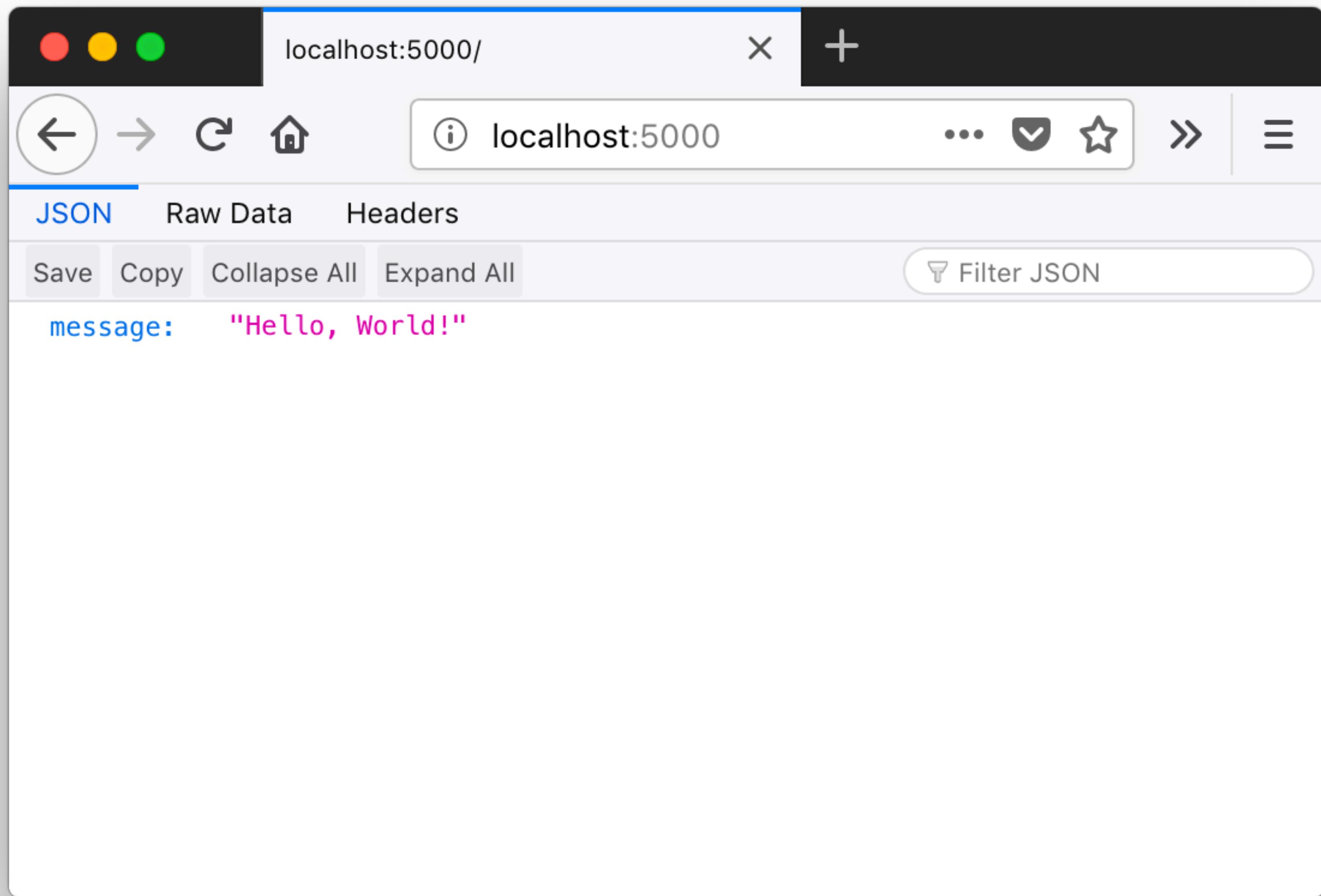
```
@etc.  
def hello():  
    pass # do something
```



# 秒变 Web API



```
from flask import Flask, jsonify  
app = Flask(__name__)  
  
@app.route('/')  
def hello():  
    # return 'Hello, Flask!'  
    return jsonify({'message': 'Hello, Flask!'})
```



# 方式 3

类方法 + 单独注册

```
from flask import Flask, MethodView
app = Flask(__name__)

class Hello(MethodView):
    def get():
        return 'Hello, Flask!'

app.add_url_rule('/', Hello.as_view('hello'))
```

# API 资源类

```
from flask import Flask, MethodView, jsonify
app = Flask(__name__)

class Hello(MethodView):

    def get():
        return jsonify({'message': 'Hello, Flask!'})

    def post():
        pass

    def delete():
        pass

app.add_url_rule('/', Hello.as_view('hello'),
                  methods=['GET', 'POST', 'DELETE'])
```

## Chapter 4

# 自由的项目结构

# 小型项目

单脚本

```
- app.py
```

```
- templates/  
- static/  
- app.py
```

\$ flask run

# 中型项目

包

```
$ export FLASK_APP=hello  
$ flask run
```

```
hello/  
- __init__.py  
- templates/  
- static/  
- views.py  
- forms.py  
- errors.py  
- models.py  
- commands.py  
- settings.py
```

# 大型项目结构 1

功能式架构

```
$ export FLASK_APP=hello  
$ flask run
```

```
:hello/  
    blueprints/  
        - __init__.py  
        - blog.py  
        - auth.py  
    forms/  
        - __init__.py  
        - blog.py  
        - auth.py  
    templates/  
        - auth/  
        - blog/  
        - base.html  
        - macros.html  
    static/  
        __init__.py  
models.py  
extensions.py
```

# 大型项目结构 2

分区式架构

```
$ export FLASK_APP=hello  
$ flask run
```

```
:hello/  
  blog/  
    - __init__.py  
    - views.py  
    - forms.py  
auth/  
  - __init__.py  
  - views.py  
  - forms.py  
templates/  
  - auth/  
  - blog/  
  - base.html  
  - macros.html  
static/  
  __init__.py  
models.py  
extensions.py
```

etc.



“Life is Short, Run Naked Use Flask!”

– Grey Li

# Resources

H

- 幻灯片以及相关源码：[github.com/greyli/freeflask](https://github.com/greyli/freeflask)
- 更多Flask开源项目和文章：[helloflask.com](https://helloflask.com)

# Bonus



# Thanks!

- 主页: [greyli.com](http://greyli.com)
- GitHub: [github.com/greyli](https://github.com/greyli)
- Email: [withlihui@gmail.com](mailto:withlihui@gmail.com)
- 微信 & Telegram: greylihui

