Web Development

COMP 431 / COMP 531

Back End

Scott E Pollack, PhD

March 10, 2016

Part II – Back End Development

- Homework Assignment 5 (Front-End App)
  - **Due TONIGHT 3/10**

Homework Assignment 6
(Draft Back-End)
Due Thursday 3/24

PART II
~~Web Servers~~
Backend
Architecture
Unit Testing
Web Hosting
Databases

Languages, Platforms, Frameworks

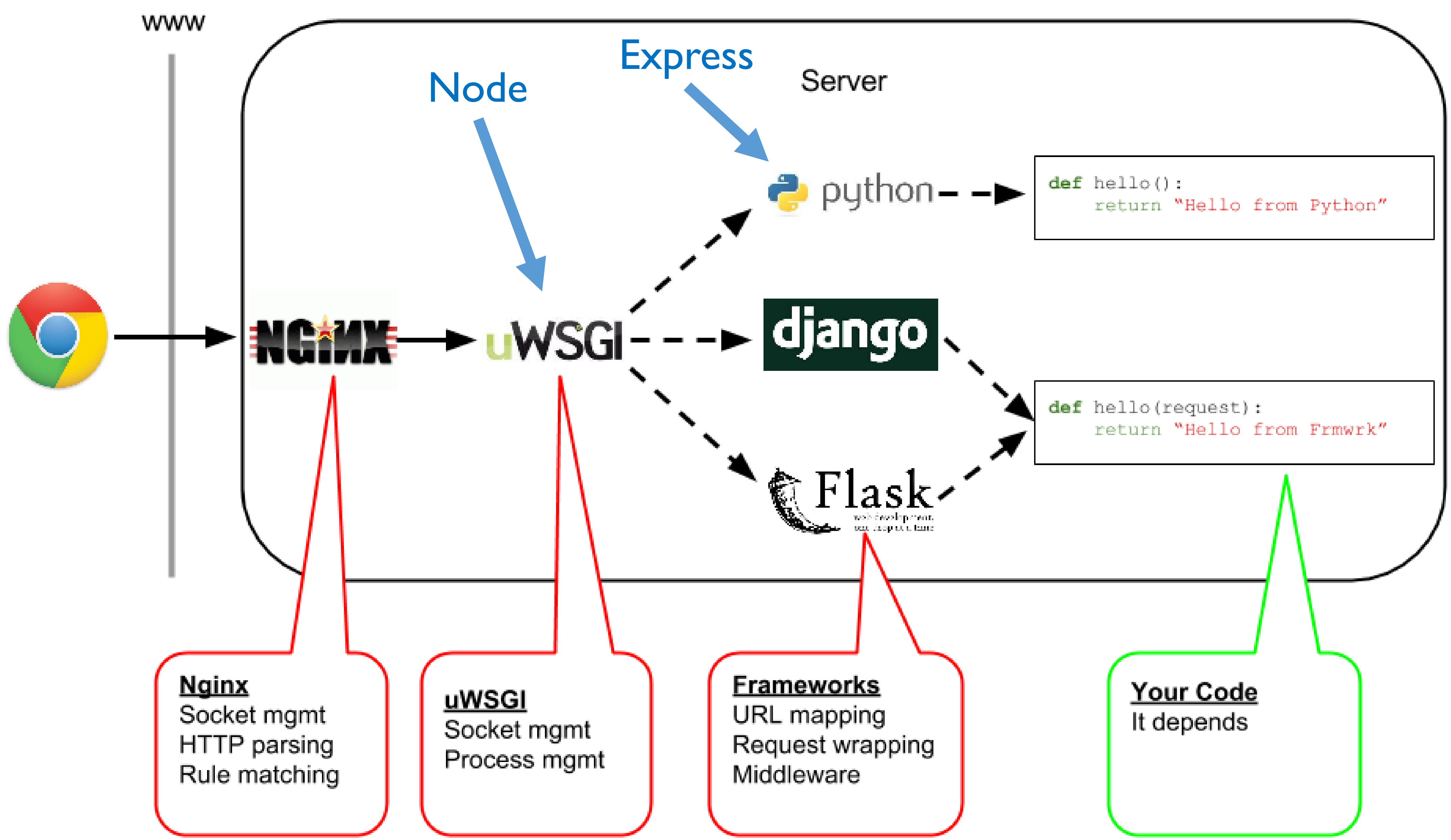
- Lots of choices
- Each have advantages and disadvantages
- Consider the long term solution
- Consider a short term solution
- Write something that you might throw away, but be careful…

Python: Web Server Gateway Interface

- For Python, options are CGI, FastCGI, and mod_python (for Apache)

- WSGI is a low-level interface to promote portability
- Instead of running within a web server, we'll start our own.
- WSGI is only an interface.
  Need utility libraries to provide us implementation for our app

- uWSGI is a universal web server gateway interface
  - Works with python, perl, ruby
- There's also mod_wsgi for those using Apache



```
def hello():
    return "Hello from Python"
```

```
def hello(request):
    return "Hello from Frmwrk"
```

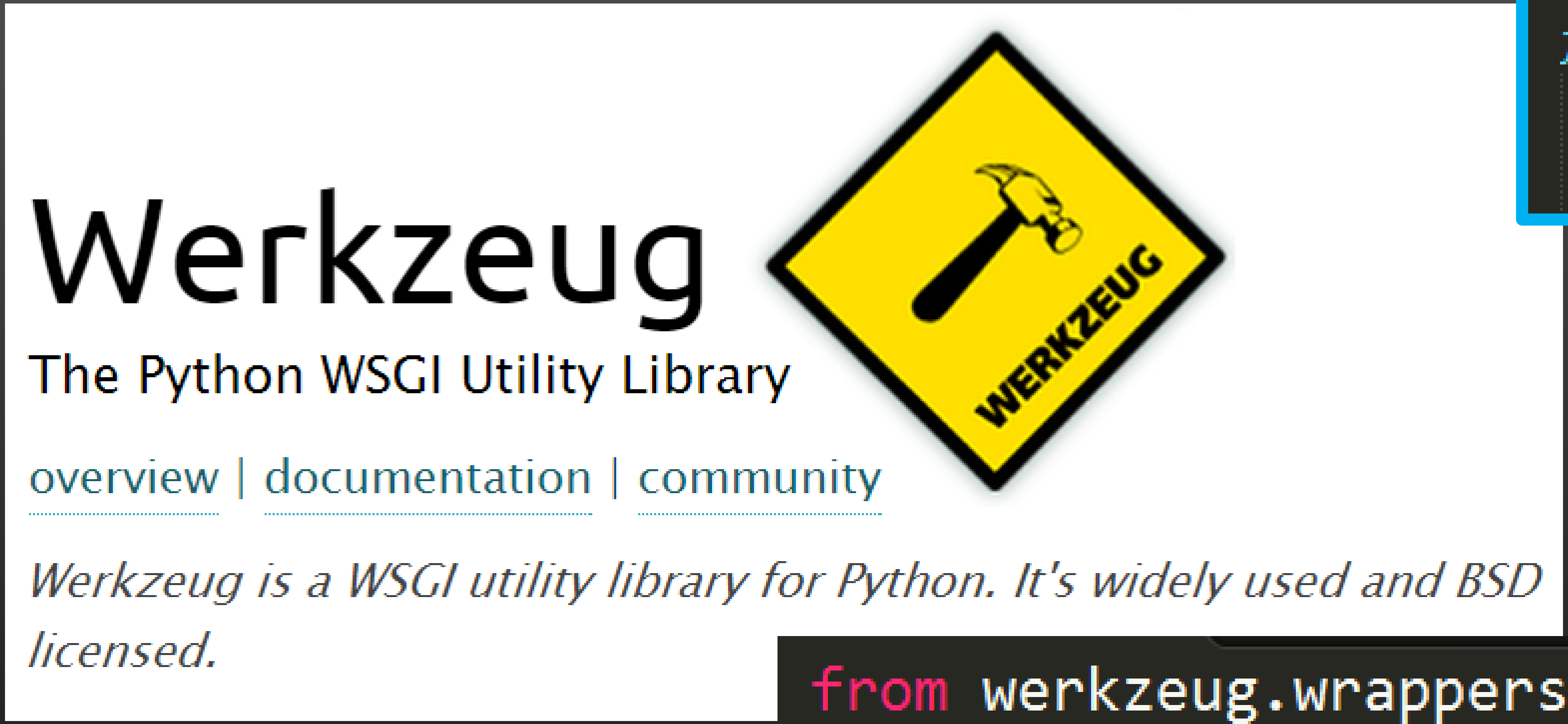
Nginx
Socket mgmt
HTTP parsing
Rule matching

uWSGI
Socket mgmt
Process mgmt

Frameworks
URL mapping
Request wrapping
Middleware

Your Code
It depends

http://www.redsuncube.com/2015/08/host-your-django-website-in-ubuntu.html

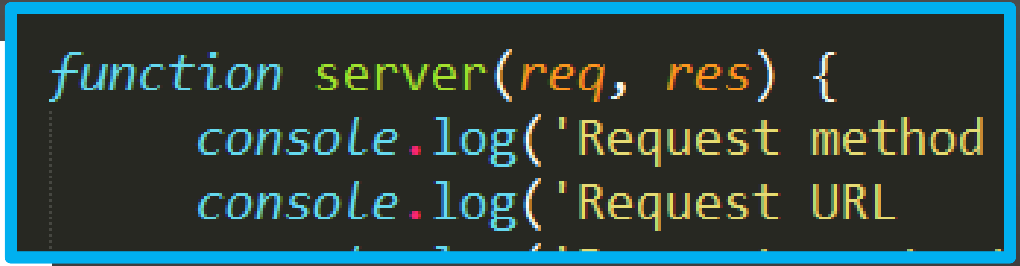


Werkzeug

The Python WSGI Utility Library

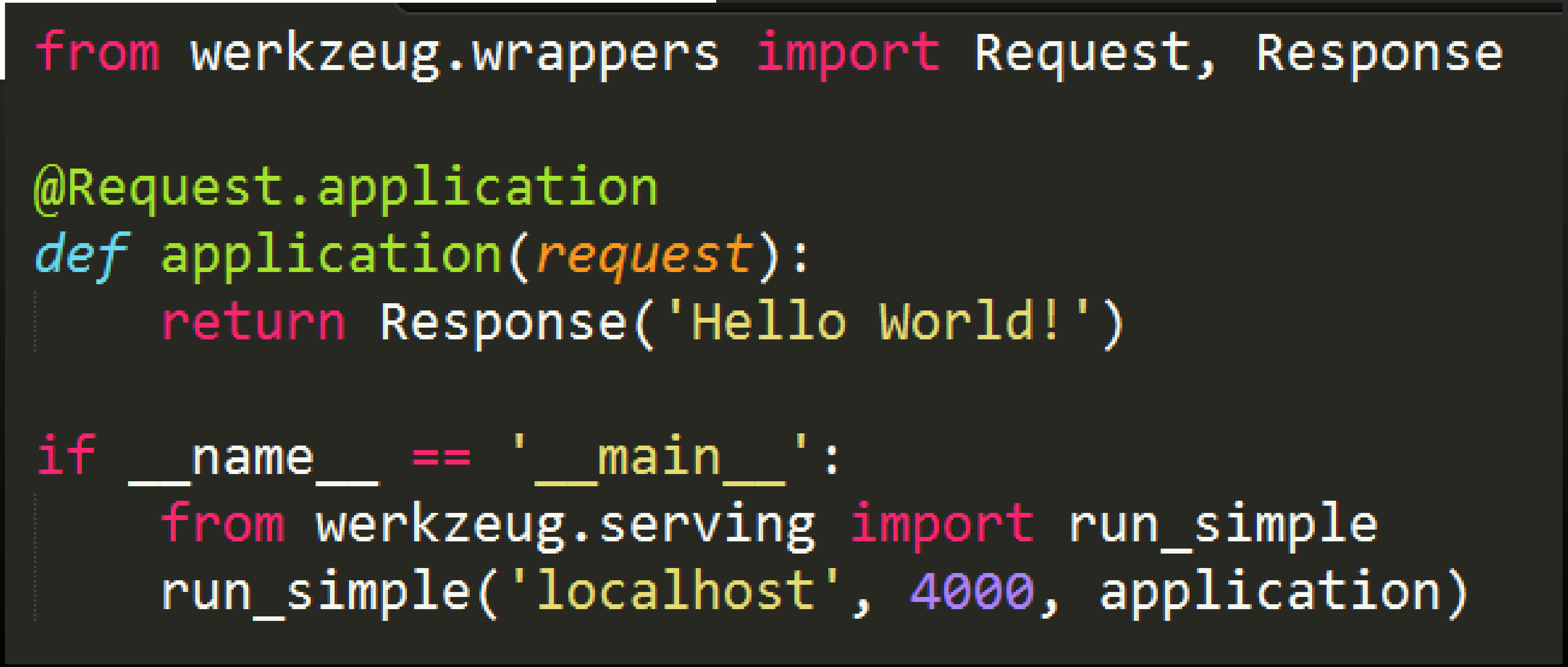
overview | documentation | community

Werkzeug is a WSGI utility library for Python. It's widely used and BSD licensed.



```
function server(req, res) {
      console.log('Request method
      console.log('Request URL
```

NodeJS



```
from werkzeug.wrappers import Request, Response

@Request.application
def application(request):
    return Response('Hello World!')

if __name__ == '__main__':
    from werkzeug.serving import run_simple
    run_simple('localhost', 4000, application)
```

Python

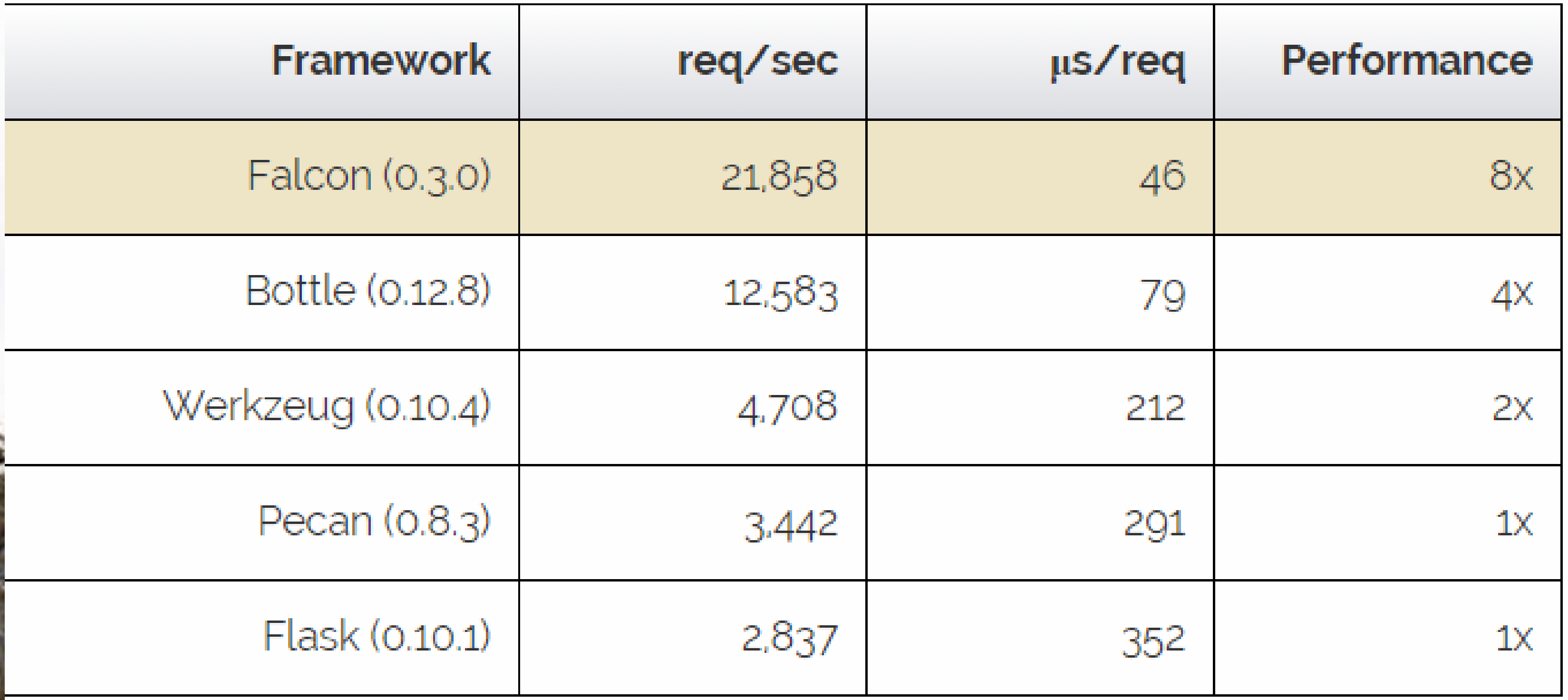
Middleware

- Middleware is *anything* you put in between the server/gateway and the final application/framework
- Middleware is compliant,
  - they accept a request and pass it along
  - they accept a response and pass it along
- As middleware, they can modify the request or response, e.g.,
  - check for authentication
  - add or strip headers
  - format or transform content

Python Frameworks

- Bottle
- CherryPy
- Django
- Falcon
- Flask
- PoorWSGI
- Pyramid (Pylons)
- Web.py
- Web2py

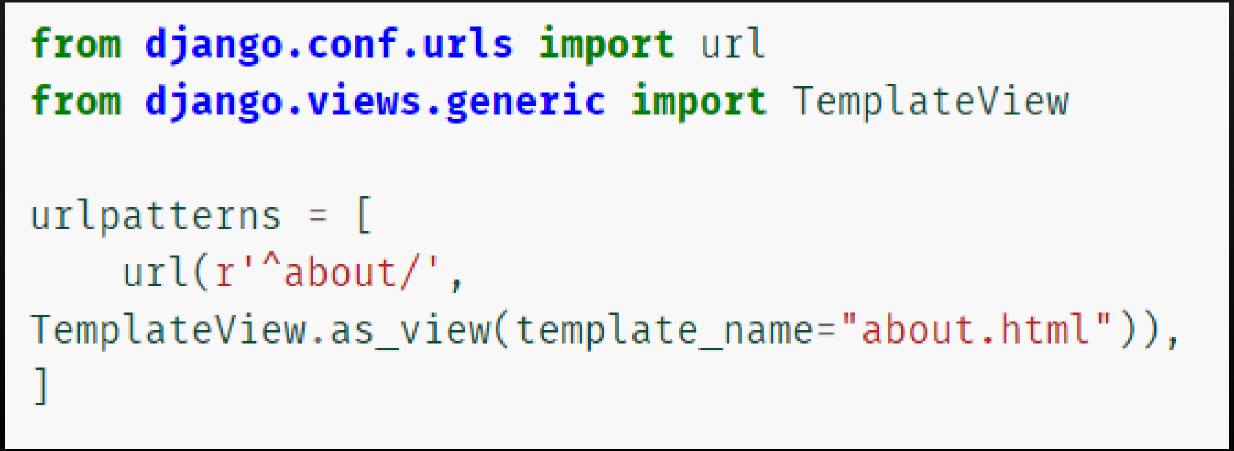
Falcon is FAST!!!



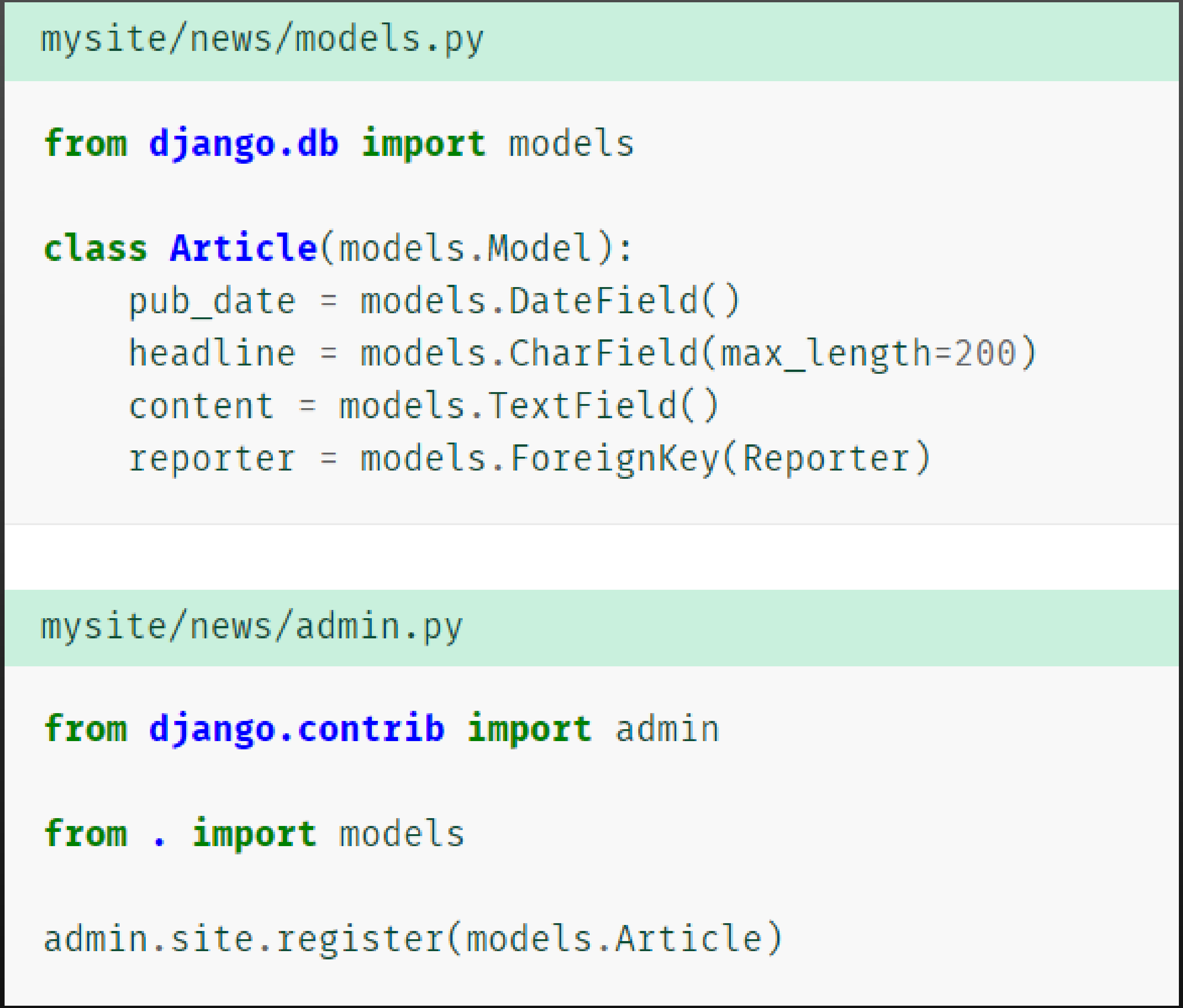
Framework	req/sec	µs/req	Performance
Falcon (0.3.0)	21,858	46	8x
Bottle (0.12.8)	12,583	79	4x
Werkzeug (0.10.4)	4,708	212	2x
Pecan (0.8.3)	3,442	291	1x
Flask (0.10.1)	2,837	352	1x

Django

- Full stack
- Opinionated
- Highly developed
- "Fat" framework
- a "sea" of functionality
- … not quick to go…



```
from django.conf.urls import url
from django.views.generic import TemplateView

urlpatterns = [
    url(r'^about/',
TemplateView.as_view(template_name="about.html")),
]
```



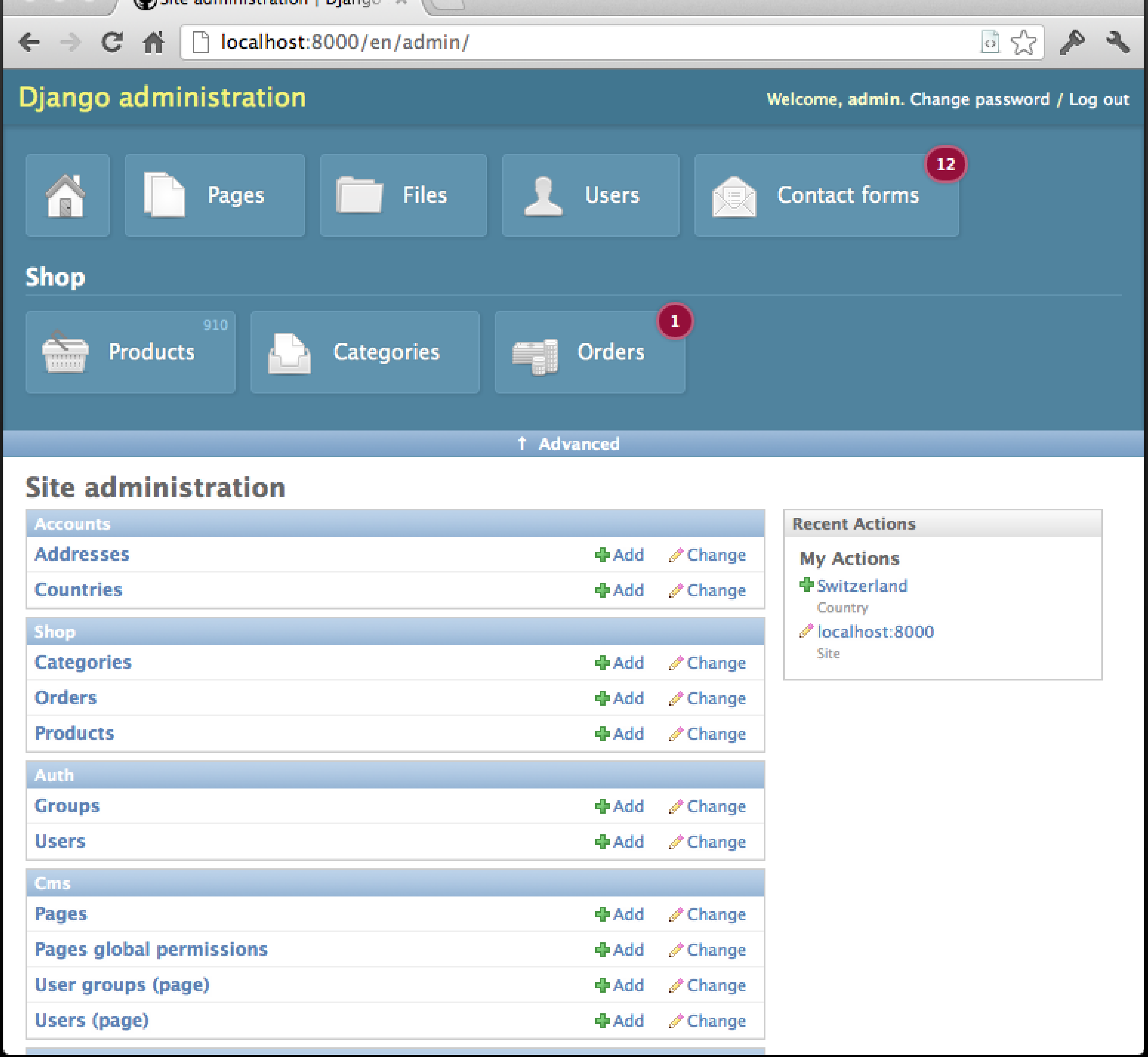
mysite/news/models.py

```
from django.db import models

class Article(models.Model):
    pub_date = models.DateField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter)
```

mysite/news/admin.py

```
from django.contrib import admin

from . import models

admin.site.register(models.Article)
```

… is … not



localhost:8000/en/admin/

Django administration

Welcome, admin. Change password / Log out

Pages | Files | Users | Contact forms 12

Shop

Products 910 | Categories | Orders 1

Advanced

Site administration

Accounts		
Addresses	Add	Change
Countries	Add	Change

Shop		
Categories	Add	Change
Orders	Add	Change
Products	Add	Change

Auth		
Groups	Add	Change
Users	Add	Change

Cms		
Pages	Add	Change
Pages global permissions	Add	Change
User groups (page)	Add	Change
Users (page)	Add	Change

Recent Actions

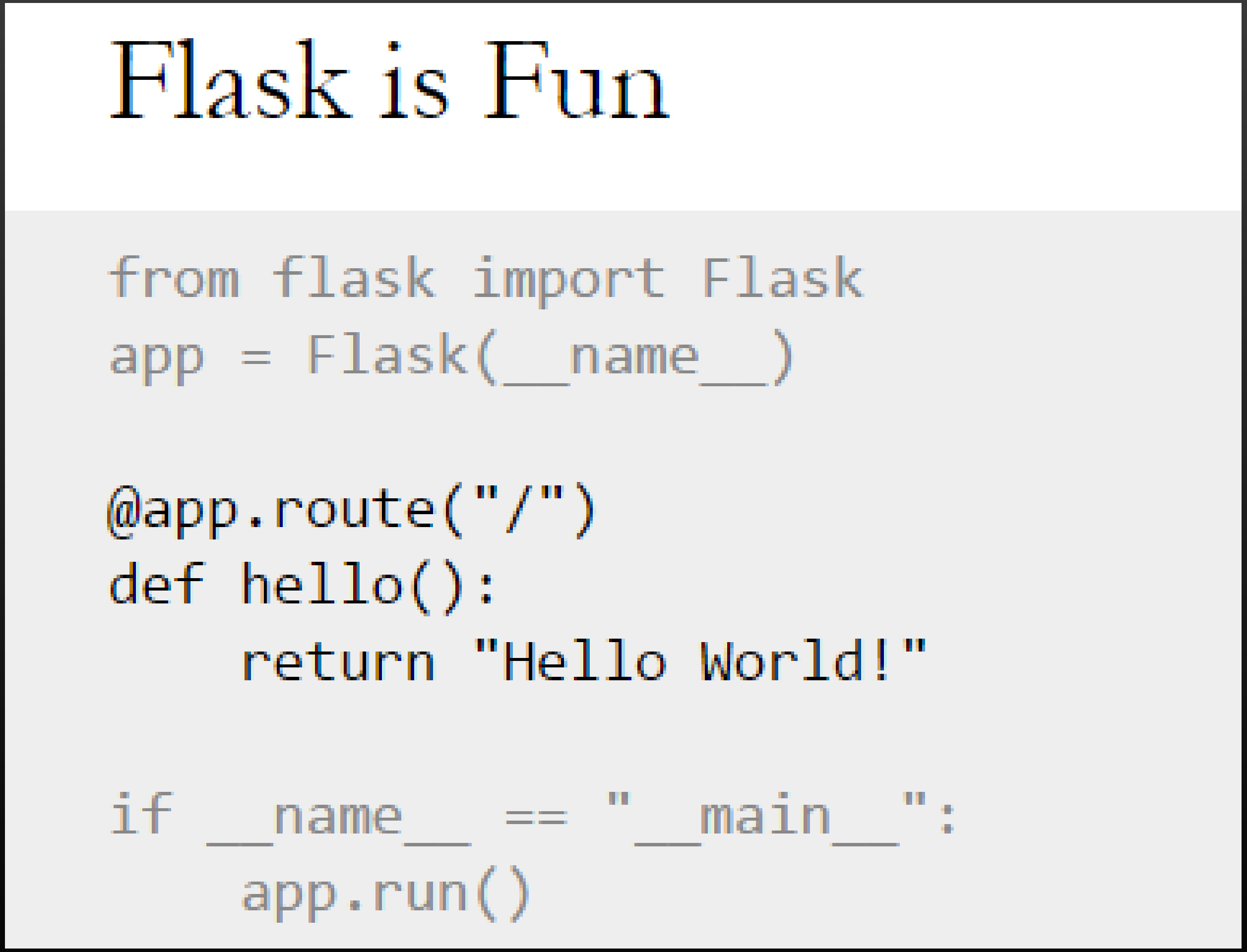
My Actions

Switzerland
Country

localhost:8000
Site

Flask
"A Python Microframework"

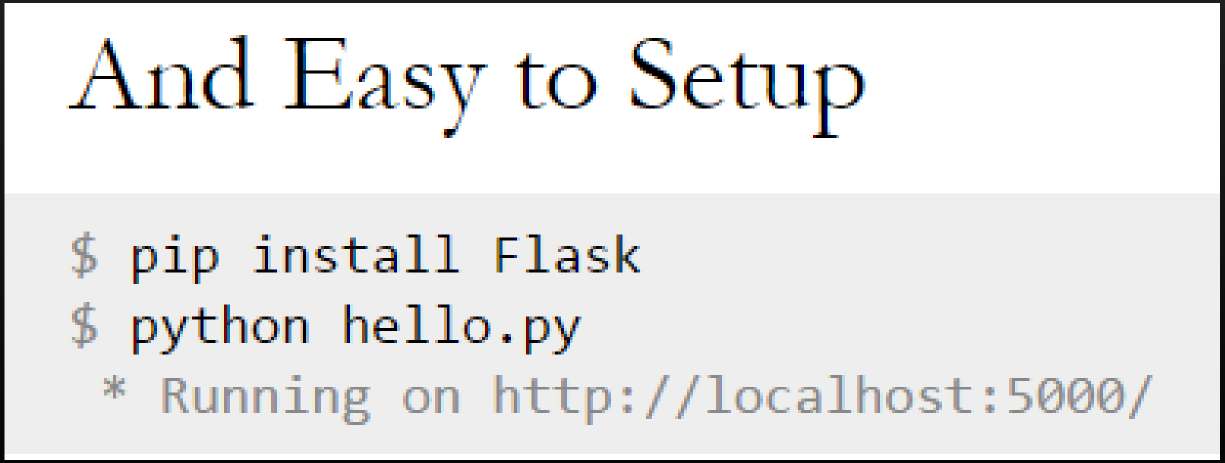
Flask
web development, one drop at a time



Flask is Fun

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

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

```
pip install virtualenv
virtualenv venv
source venv/bin/activate
```



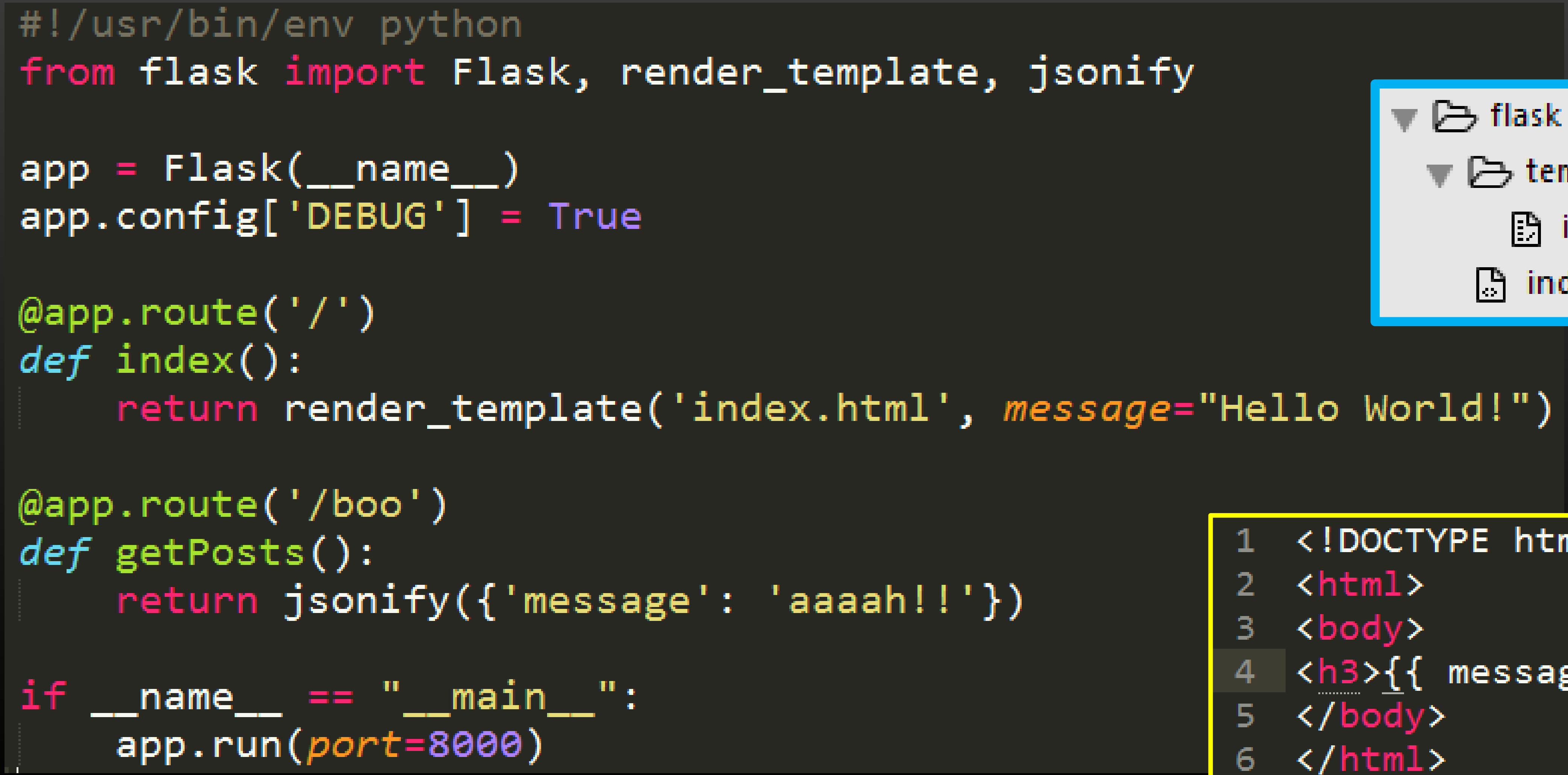
And Easy to Setup

```
$ pip install Flask
$ python hello.py
 * Running on http://localhost:5000/
```

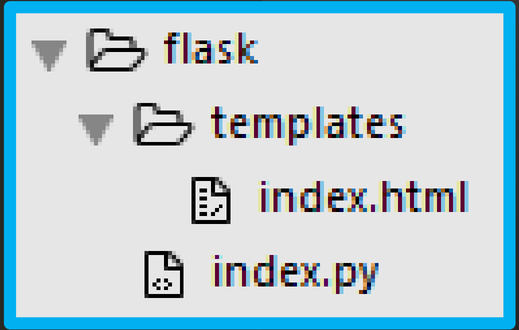
Flask in Action

Flask depends on

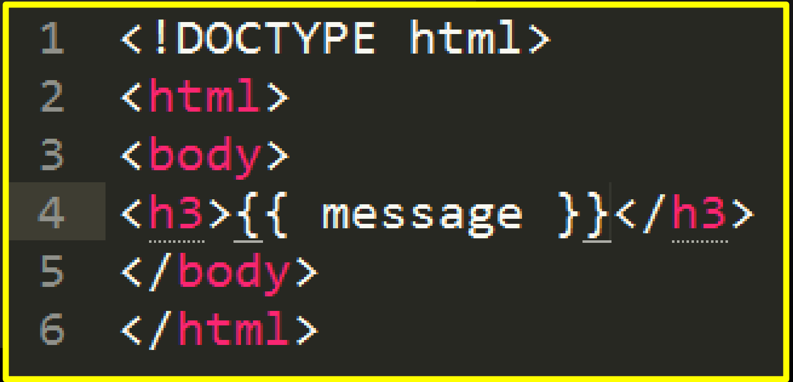
- Werkzeug WSGI toolkit
- Jinja2 template engine



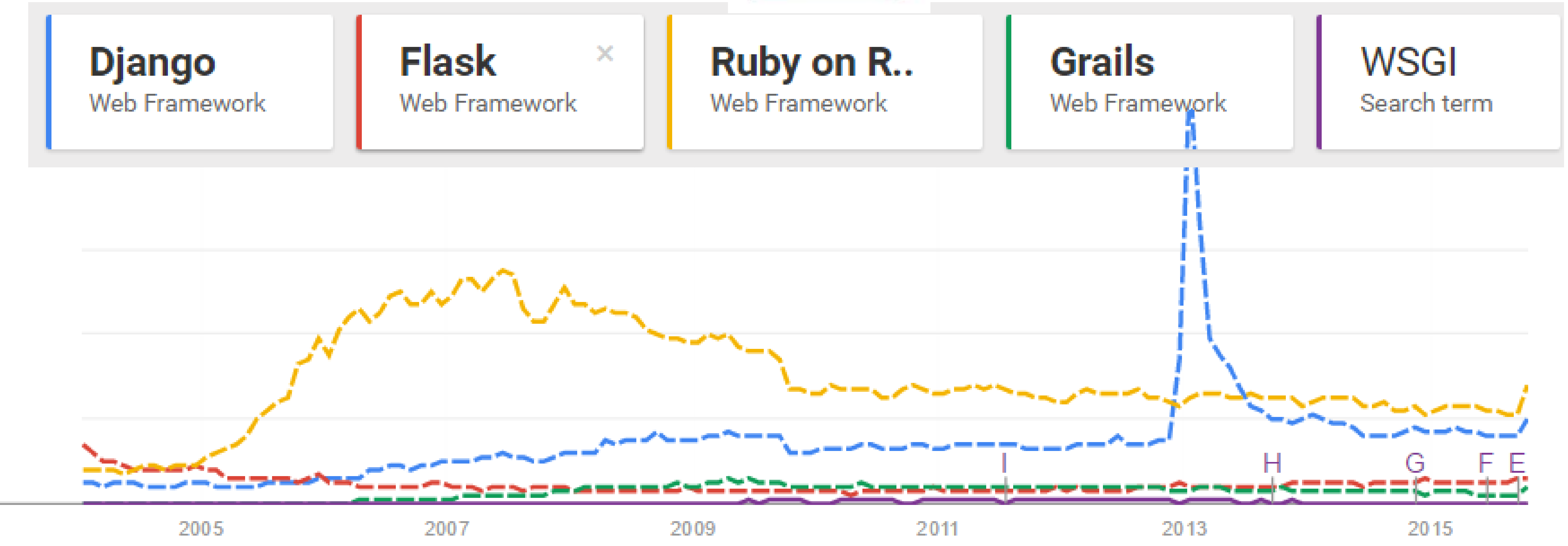
```python
#!/usr/bin/env python
from flask import Flask, render_template, jsonify

app = Flask(__name__)
app.config['DEBUG'] = True

@app.route('/')
def index():
    return render_template('index.html', message="Hello World!")

@app.route('/boo')
def getPosts():
    return jsonify({'message': 'aaaah!!'})

if __name__ == "__main__":
    app.run(port=8000)
```



```
flask
    templates
        index.html
    index.py
```

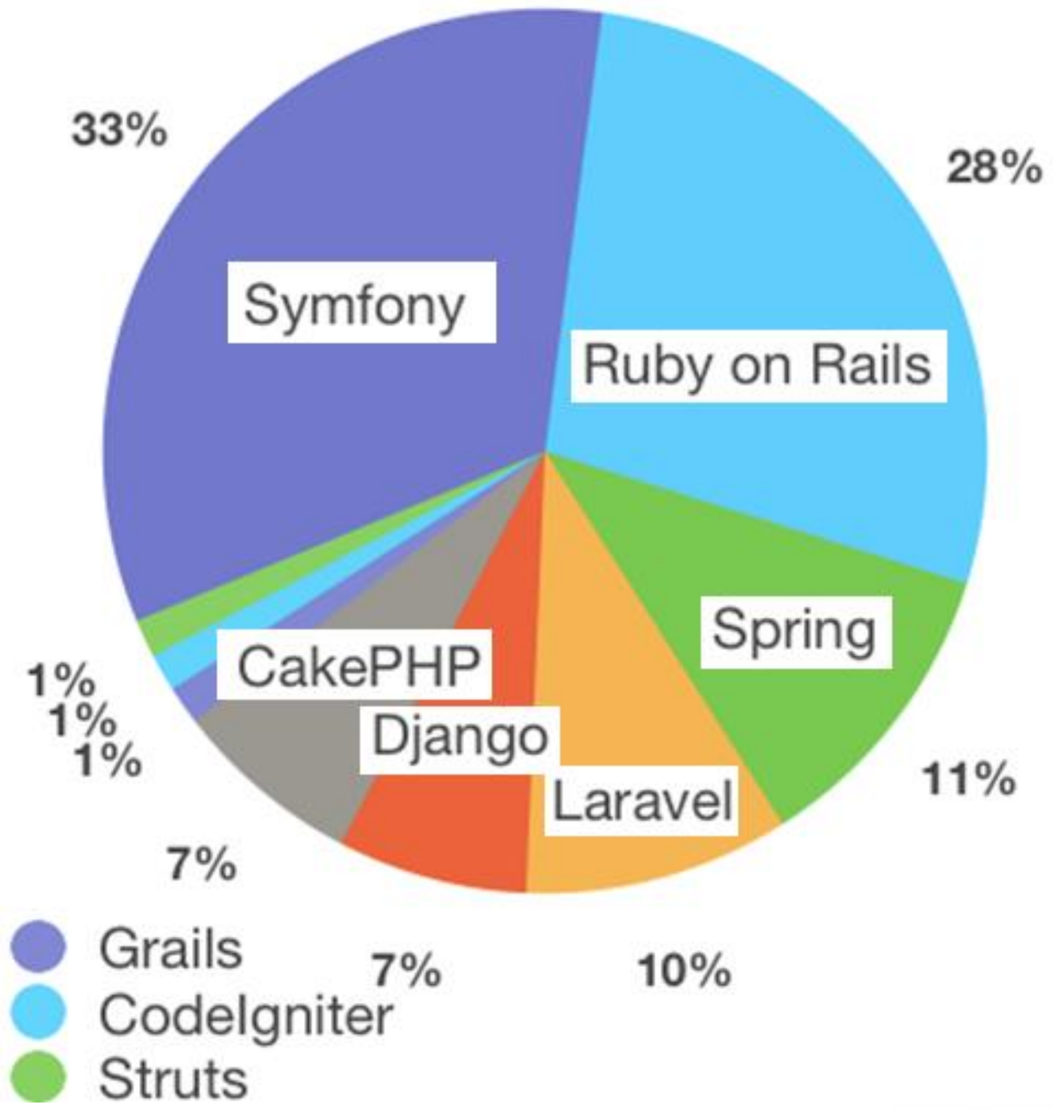


```html
<!DOCTYPE html>
<html>
<body>
<h3>{{ message }}</h3>
</body>
</html>
```

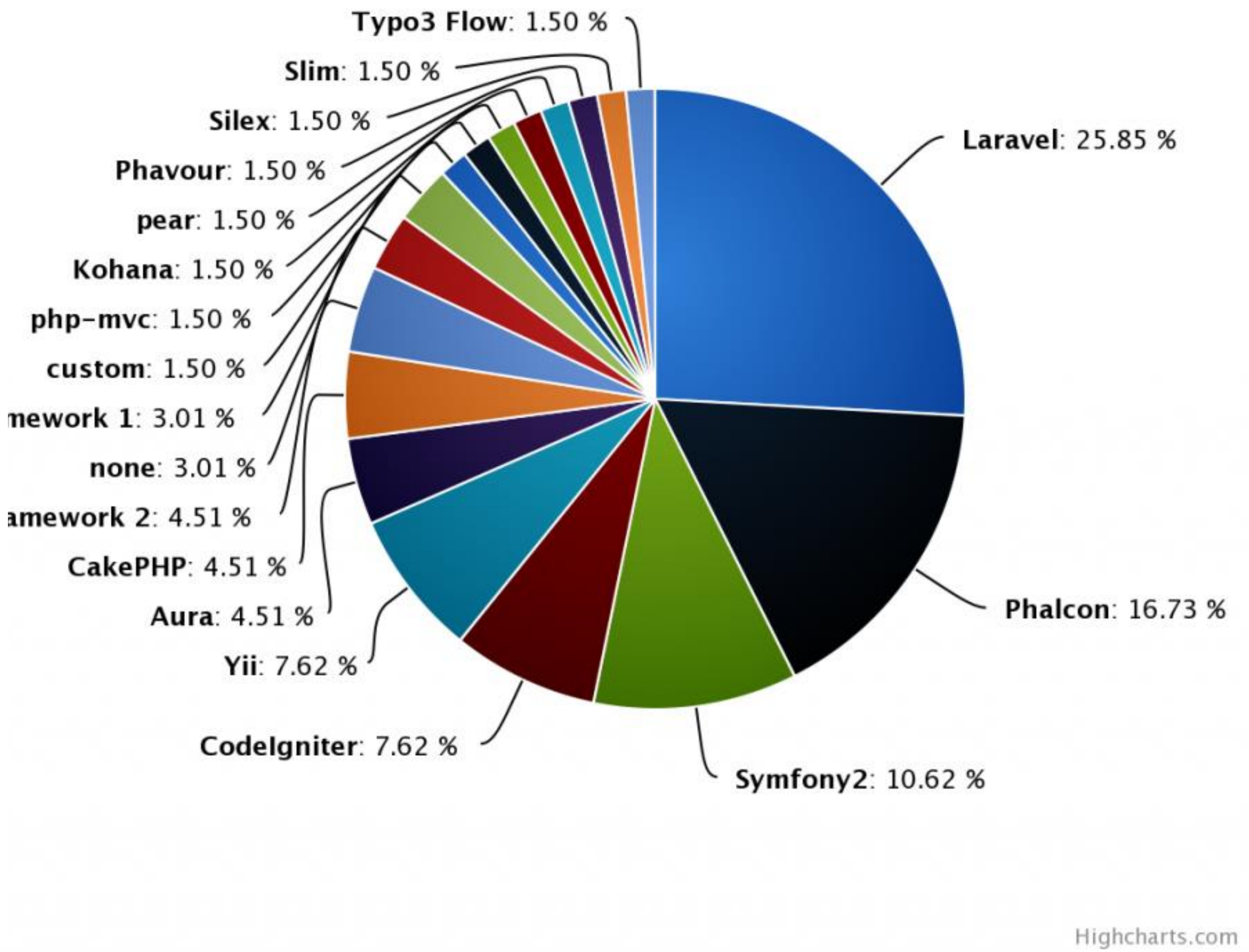

Django
Web Framework
Flask ×
Web Framework
Ruby on R..
Web Framework
Grails
Web Framework
WSGI
Search term

Trends

Trends

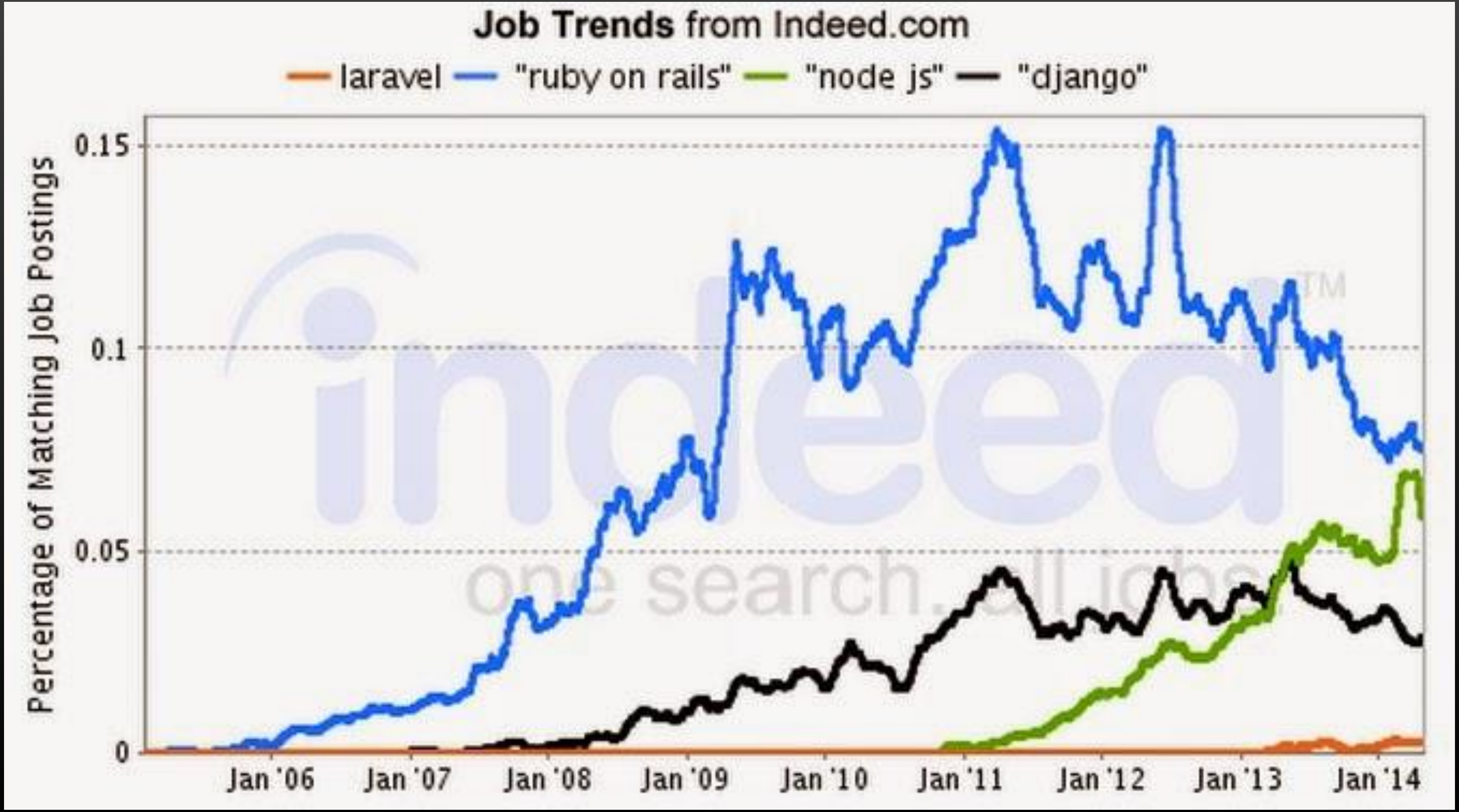


33%
28%
Symfony
Ruby on Rails
Spring
CakePHP
Django
Laravel
11%
1%
1%
1%
7%
7%
10%
● Grails
● CodeIgniter
● Struts



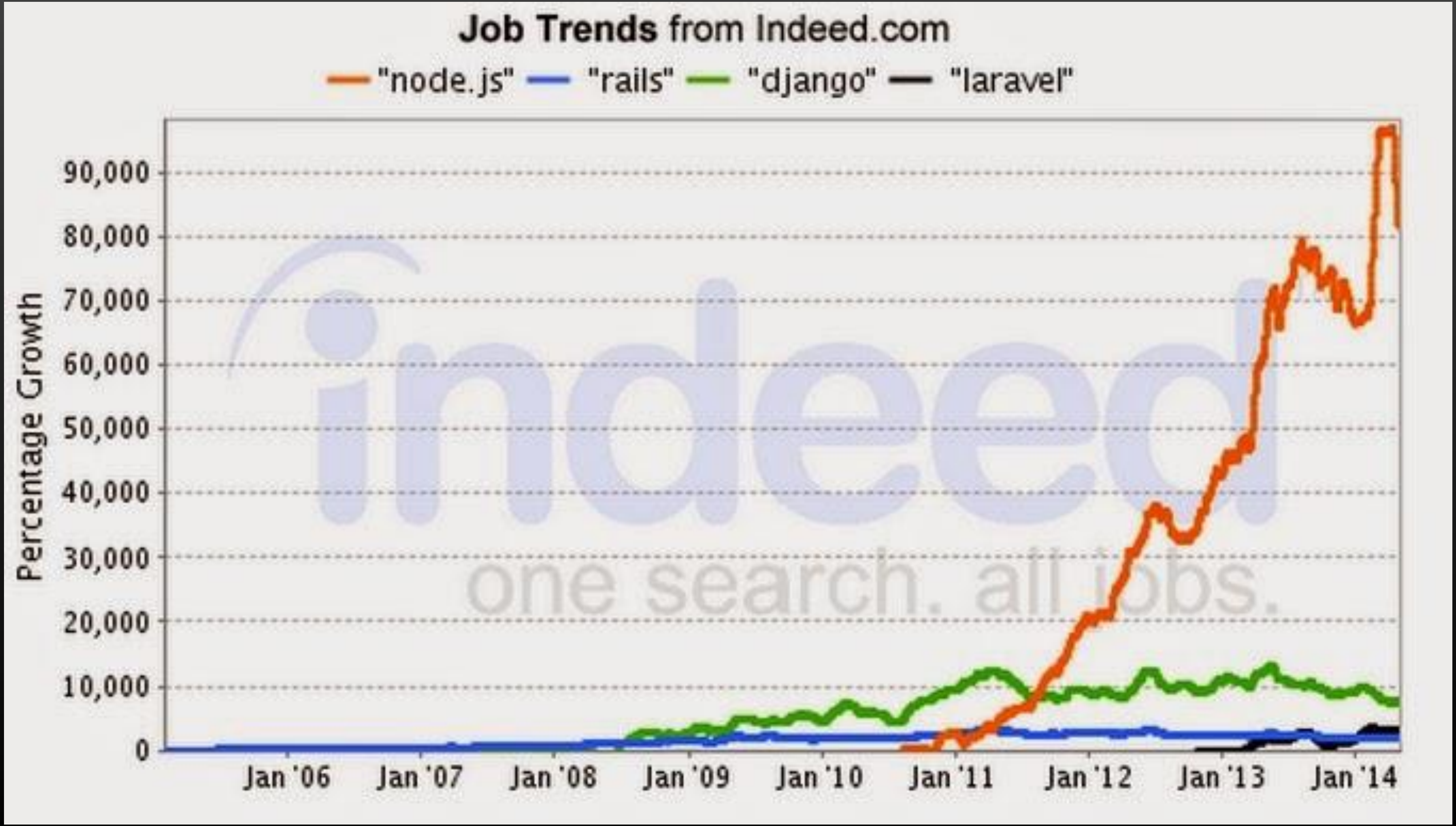
Framework popularity, end of 2013; SitePoint
Typo3 Flow: 1.50 %
Slim: 1.50 %
Silex: 1.50 %
Phavour: 1.50 %
pear: 1.50 %
Kohana: 1.50 %
php-mvc: 1.50 %
custom: 1.50 %
nework 1: 3.01 %
none: 3.01 %
amework 2: 4.51 %
CakePHP: 4.51 %
Aura: 4.51 %
Yii: 7.62 %
CodeIgniter: 7.62 %
Symfony2: 10.62 %
Phalcon: 16.73 %
Laravel: 25.85 %
Highcharts.com

What’s Hot



Job Trends from Indeed.com
laravel
"ruby on rails"
"node js"
"django"
Percentage of Matching Job Postings
0.15
0.1
0.05
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
indeed one search. all jobs.

What’s Hot



Job Trends from Indeed.com
"node.js"
"rails"
"django"
"laravel"
Percentage Growth
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
indeed
one search. all jobs.

Beginning to get MEAN

Mongo-Express-Angular-Node

- Express adds to Node a number of helpful libraries
- Minimalist philosophy
- Middleware is key

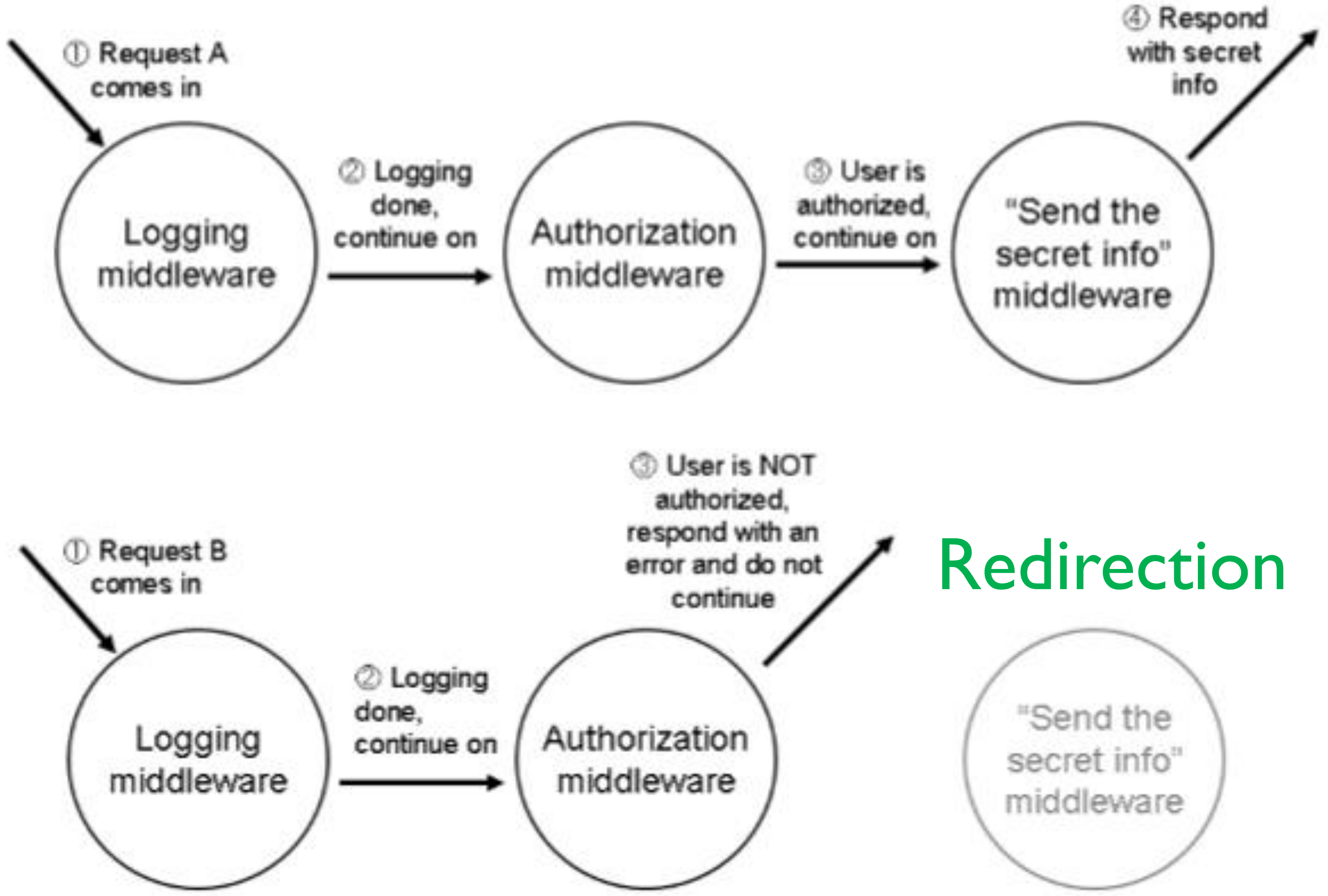
```
> mkdir backend; cd backend
> npm init -y
> npm install express --save
```

Express

Fast, unopinionated, minimalist web framework for Node.js

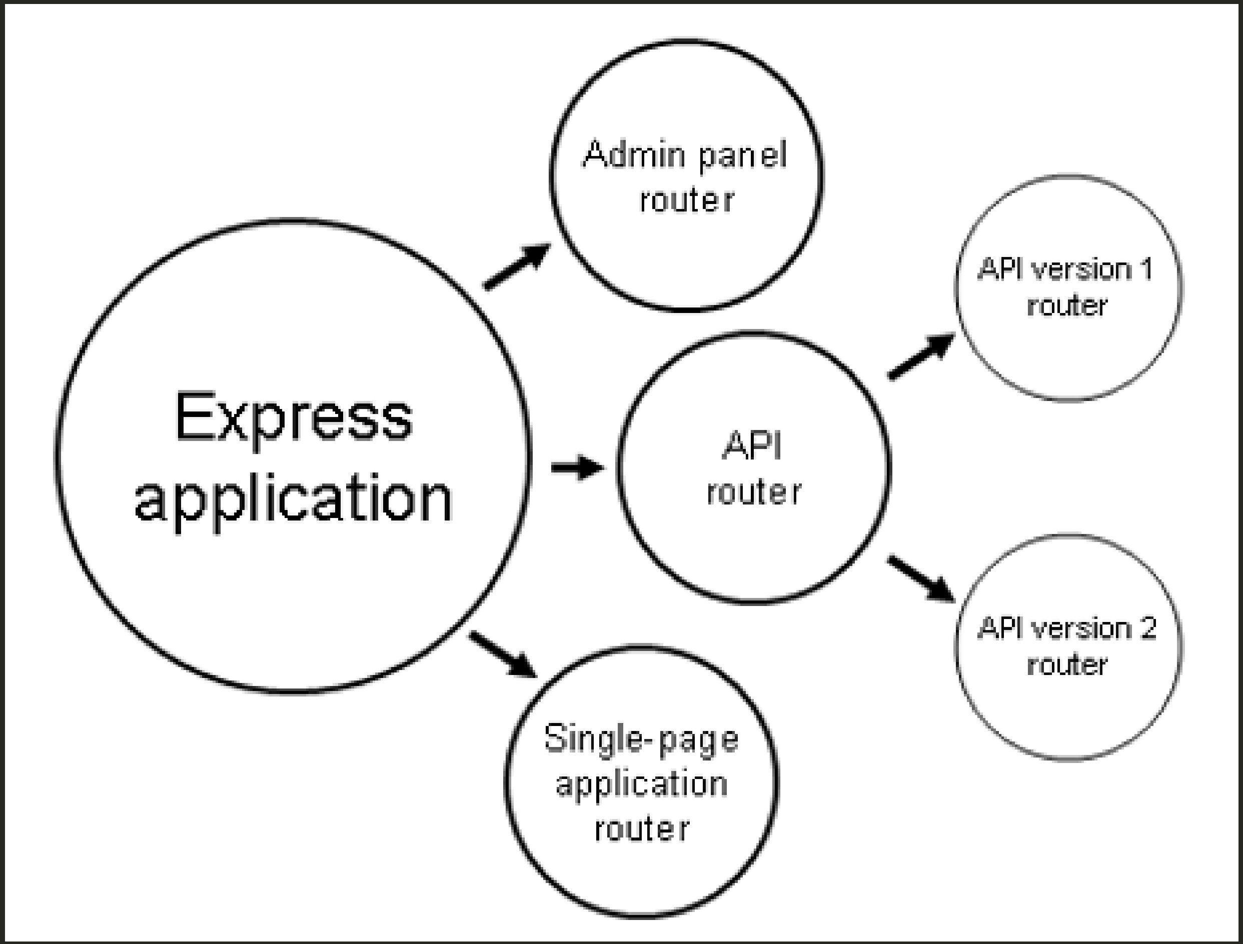
Middleware

```
app.put('/logout', isLoggedIn, logout)
function isLoggedIn(req, res, next)
```


① Request A comes in
Logging middleware
② Logging done, continue on
Authorization middleware
③ User is authorized, continue on
"Send the secret info" middleware
④ Respond with secret info
① Request B comes in
Logging middleware
② Logging done, continue on
Authorization middleware
③ User is NOT authorized, respond with an error and do not continue
Redirection
"Send the secret info" middleware

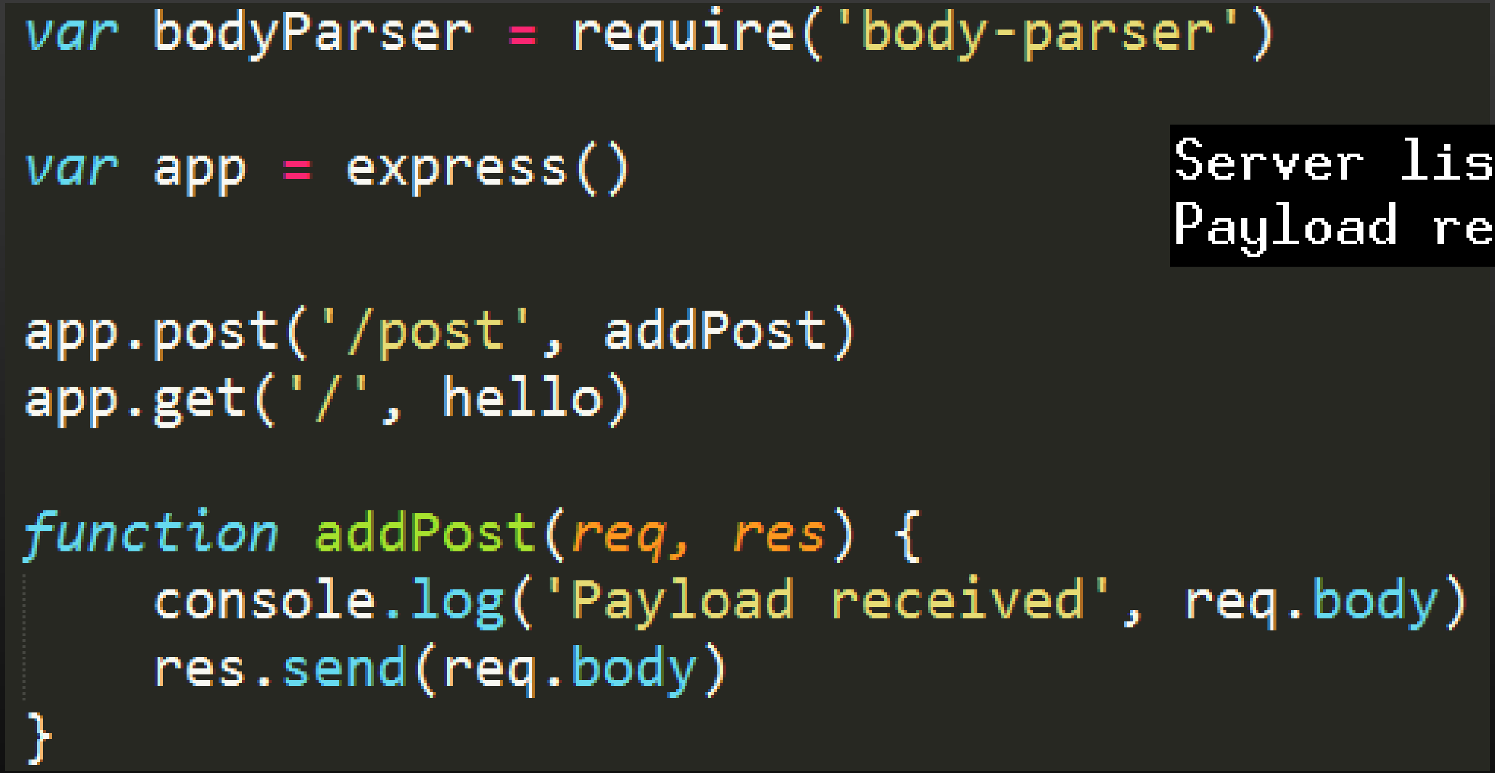
Routing with Express

```
var express = require('express')

var app = express()

app.get('/', getIndex)
app.post('/', postIndex)

function getIndex(req, res) {
    res.send('hello world!')
}

function postIndex(req, res) {
    res.send('You POSTed to the homepage')
}

var server = app.listen(8080, function() {
    console.log('Server listening at http://%s:%s',
                server.address().address,
                server.address().port)
})
```

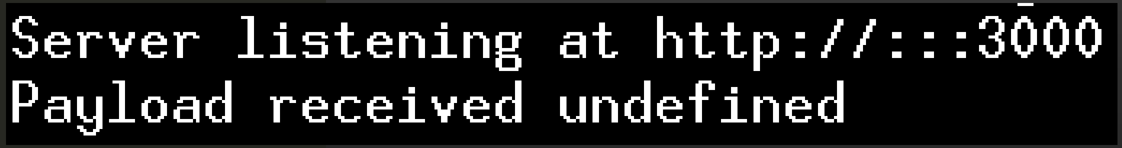


Install some Middleware

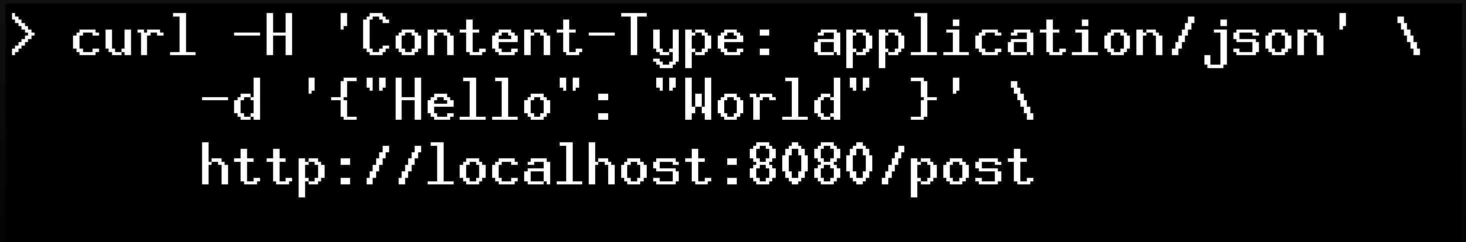
```
> npm install body-parser --save
```



```
var bodyParser = require('body-parser')

var app = express()

app.post('/post', addPost)
app.get('/', hello)

function addPost(req, res) {
    console.log('Payload received', req.body)
    res.send(req.body)
}
```



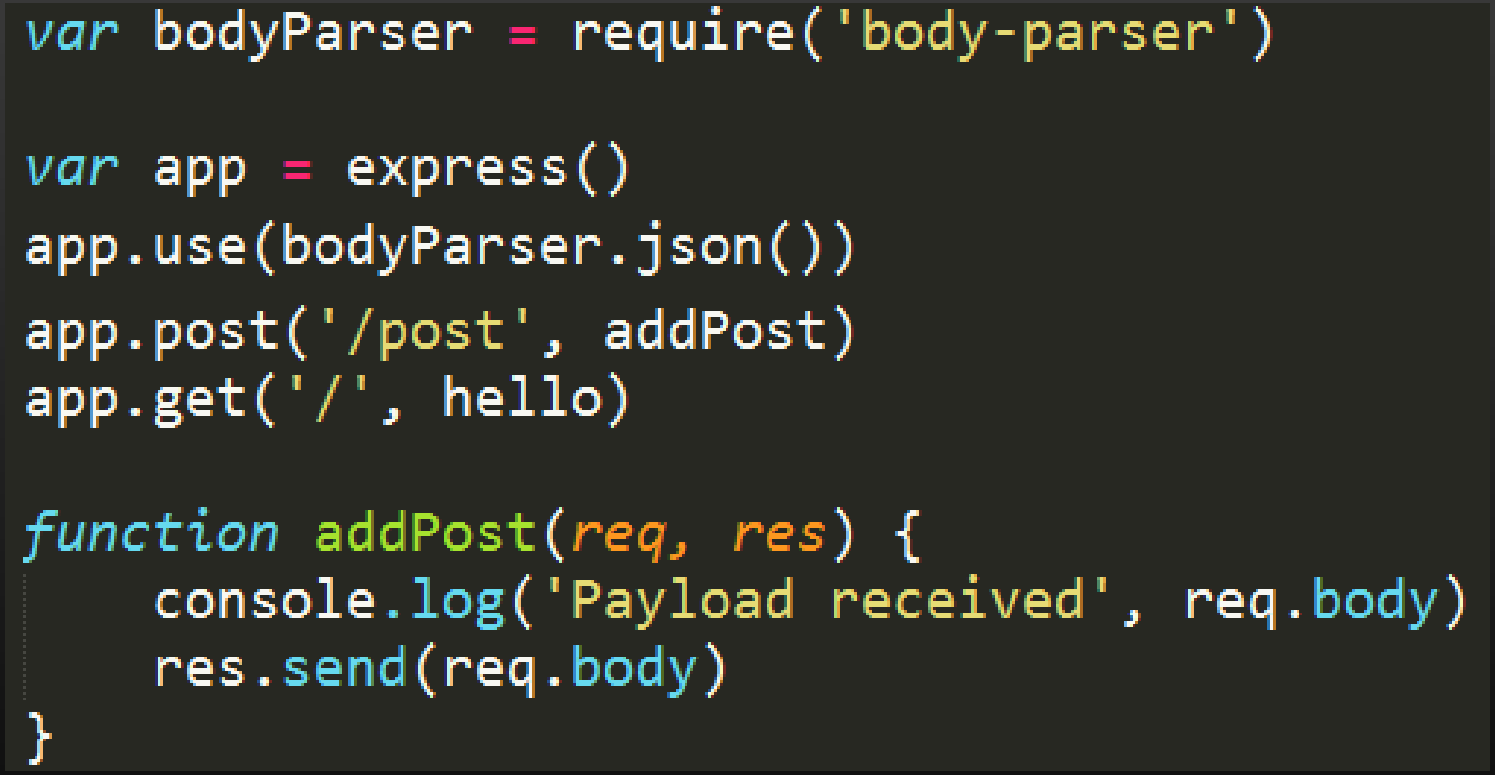
```
Server listening at http://::3000
Payload received undefined
```



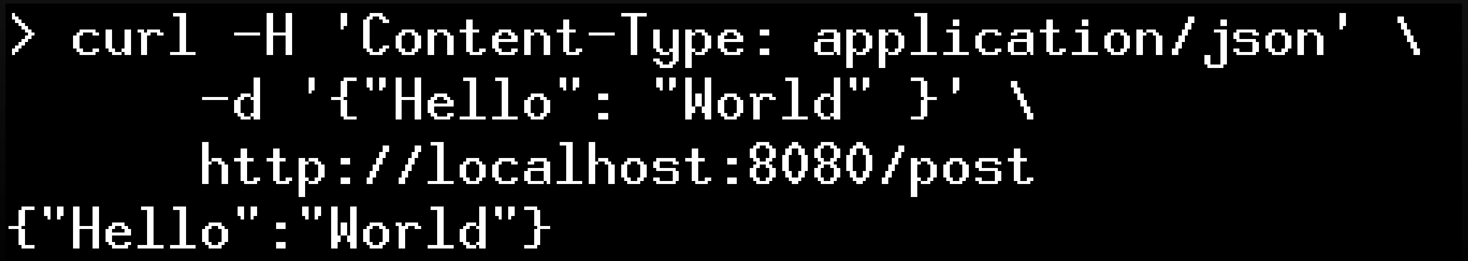
```
> curl -H 'Content-Type: application/json' \
      -d '{"Hello": "World" }' \
      http://localhost:8080/post
```

Accepting JSON Payloads

```
> npm install body-parser --save
```



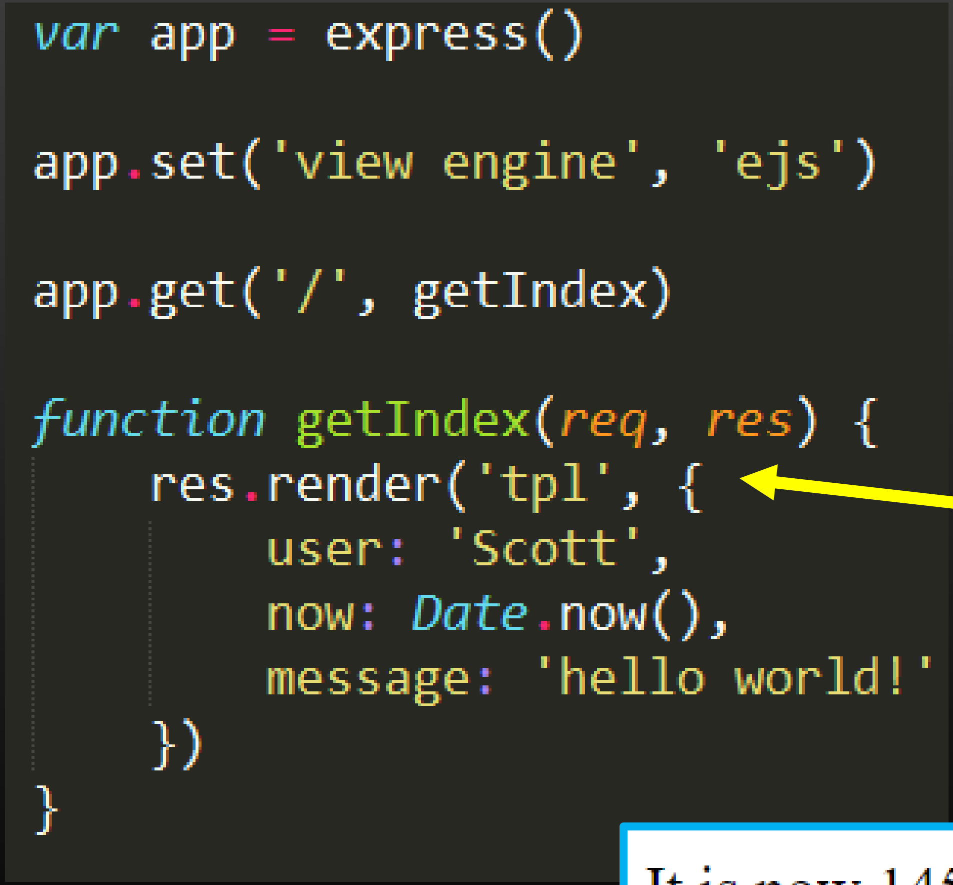
```
var bodyParser = require('body-parser')

var app = express()
app.use(bodyParser.json())
app.post('/post', addPost)
app.get('/', hello)

function addPost(req, res) {
    console.log('Payload received', req.body)
    res.send(req.body)
}
```



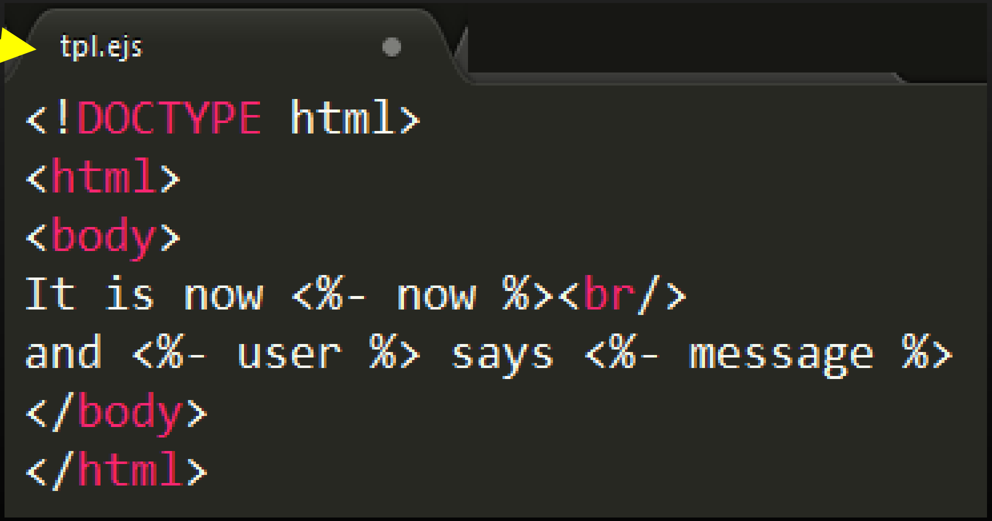
```
> curl -H 'Content-Type: application/json' \
      -d '{"Hello": "World" }' \
      http://localhost:8080/post
{"Hello":"World"}
```

Templating with Engines

Embedded JavaScript (ejs)
Jade
Swig
Nunjucks
Handlebars
Hogan
…

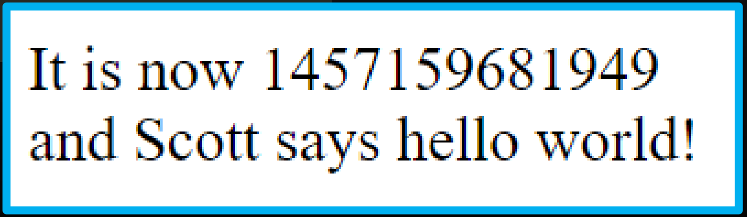


```
var app = express()

app.set('view engine', 'ejs')

app.get('/', getIndex)

function getIndex(req, res) {
    res.render('tpl', {
        user: 'Scott',
        now: Date.now(),
        message: 'hello world!'
    })
}
```



tpl.ejs

```
<!DOCTYPE html>
<html>
<body>
It is now <%- now %><br/>
and <%- user %> says <%- message %>
</body>
</html>
```



It is now 1457159681949
and Scott says hello world!

https://www.clear.rice.edu/comp431/sample/RiceBookServer/index.js

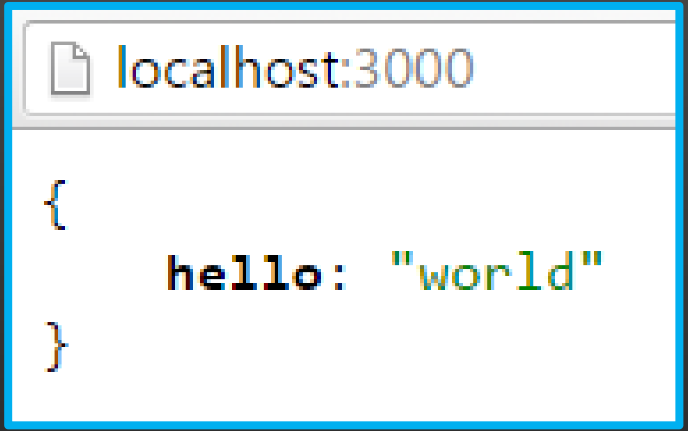
In-Class Exercise: Express Server

1. Below we respond with json payloads via bodyParser
2. Make the default "GET /" return { hello: 'world' }
3. Add "GET /post" that supplies JSON posts, start with 3 hard coded posts
4. Add "POST /post" that receives a JSON post, return the post with an id, and add the post to the list returned by GET
5. You should only have 3 endpoints

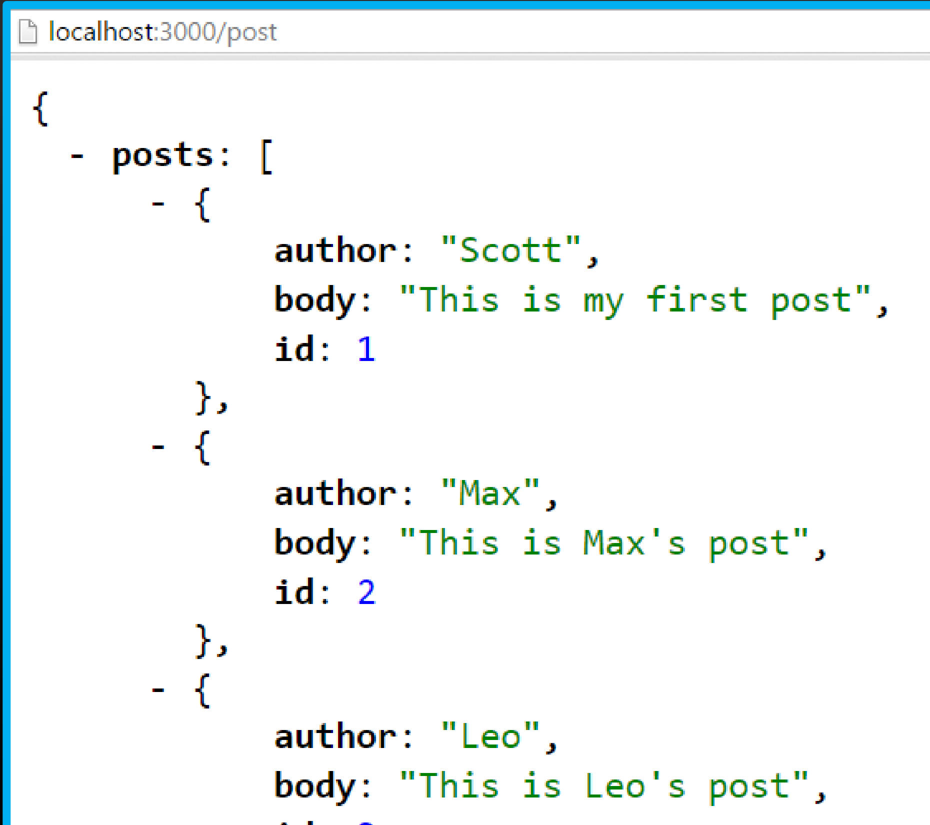
   app.get|post (…)

Turnin* index.js *to

COMP431-S16:inclass-16


localhost:3000

```
{
    hello: "world"
}
```


localhost:3000/post

```
{
  - posts: [
      - {
            author: "Scott",
            body: "This is my first post",
            id: 1
        },
      - {
            author: "Max",
            body: "This is Max's post",
            id: 2
        },
      - {
            author: "Leo",
            body: "This is Leo's post",
```