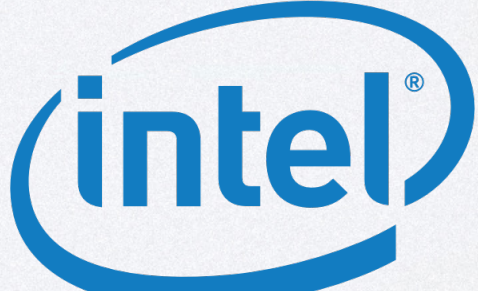


BUILDING BEAUTIFUL **REST** APIs

with Flask, Swagger UI and Flask-RESTPlus

Michał Karzyński • EuroPython 2016

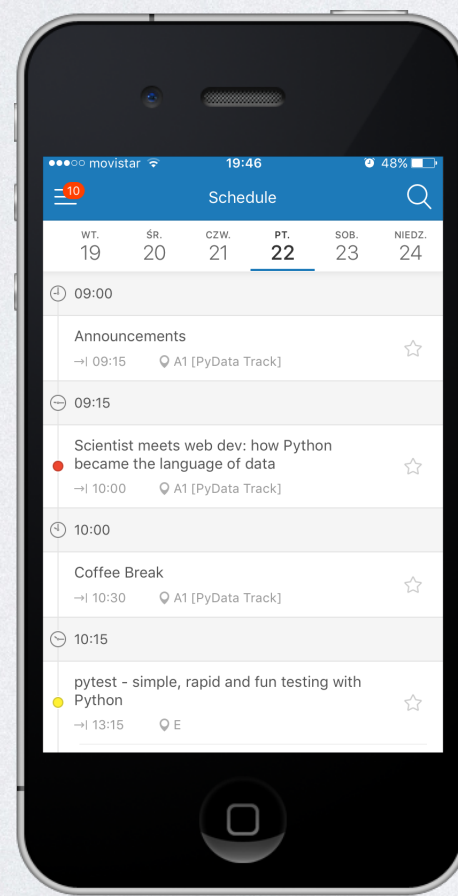
ABOUT ME

- My name is Michał Karzyński (that's Polish for Mike)
- I blog at <http://michal.karzynski.pl>
Short URL: karzyn.com
- I wrote a book for Linux admins, I write code in Python and JavaScript
- I'm the tech lead of a Web UI team at 

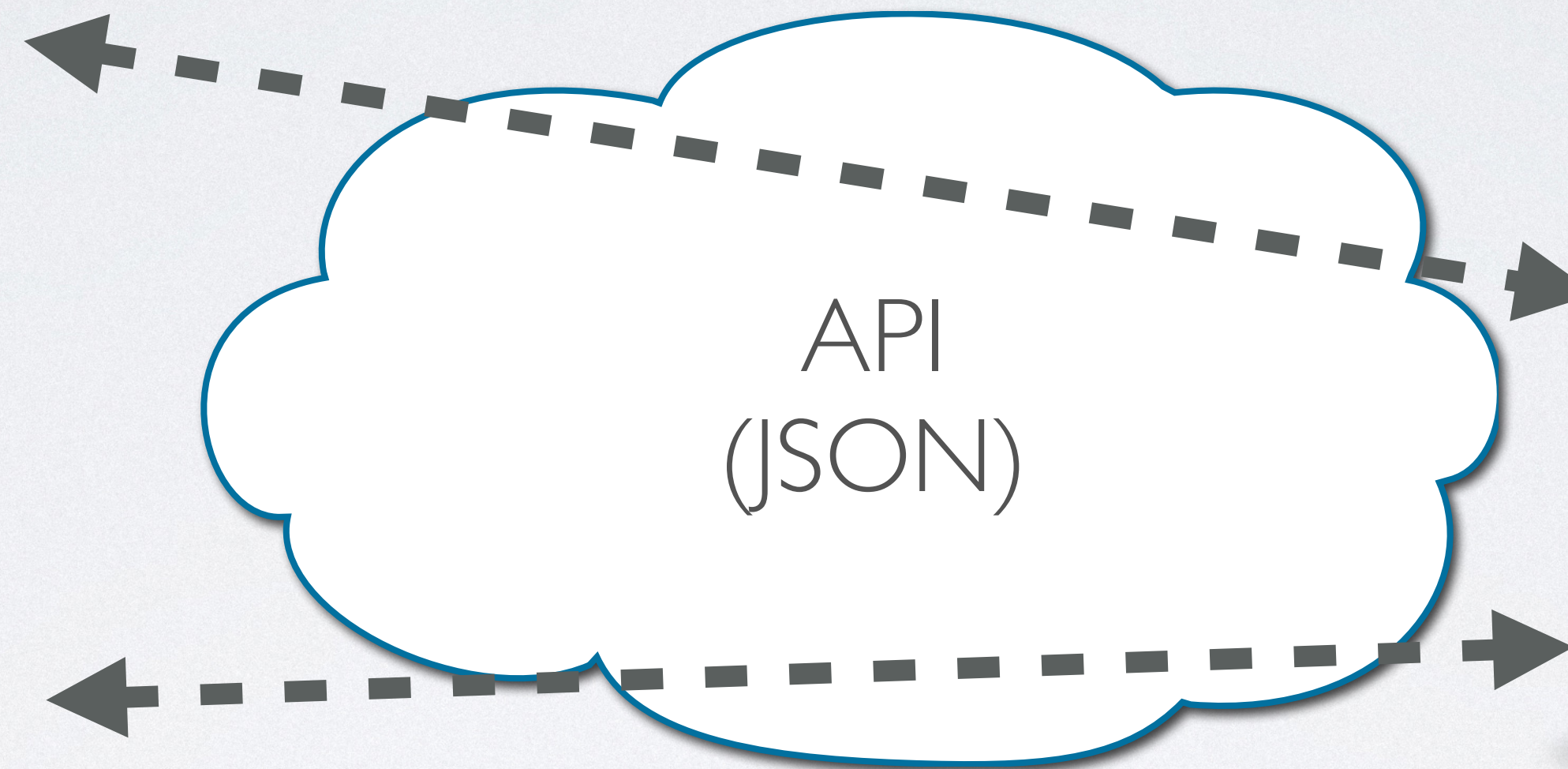
WHAT IS A WEB API?



Web (JavaScript)



Phone (Swift, Java)



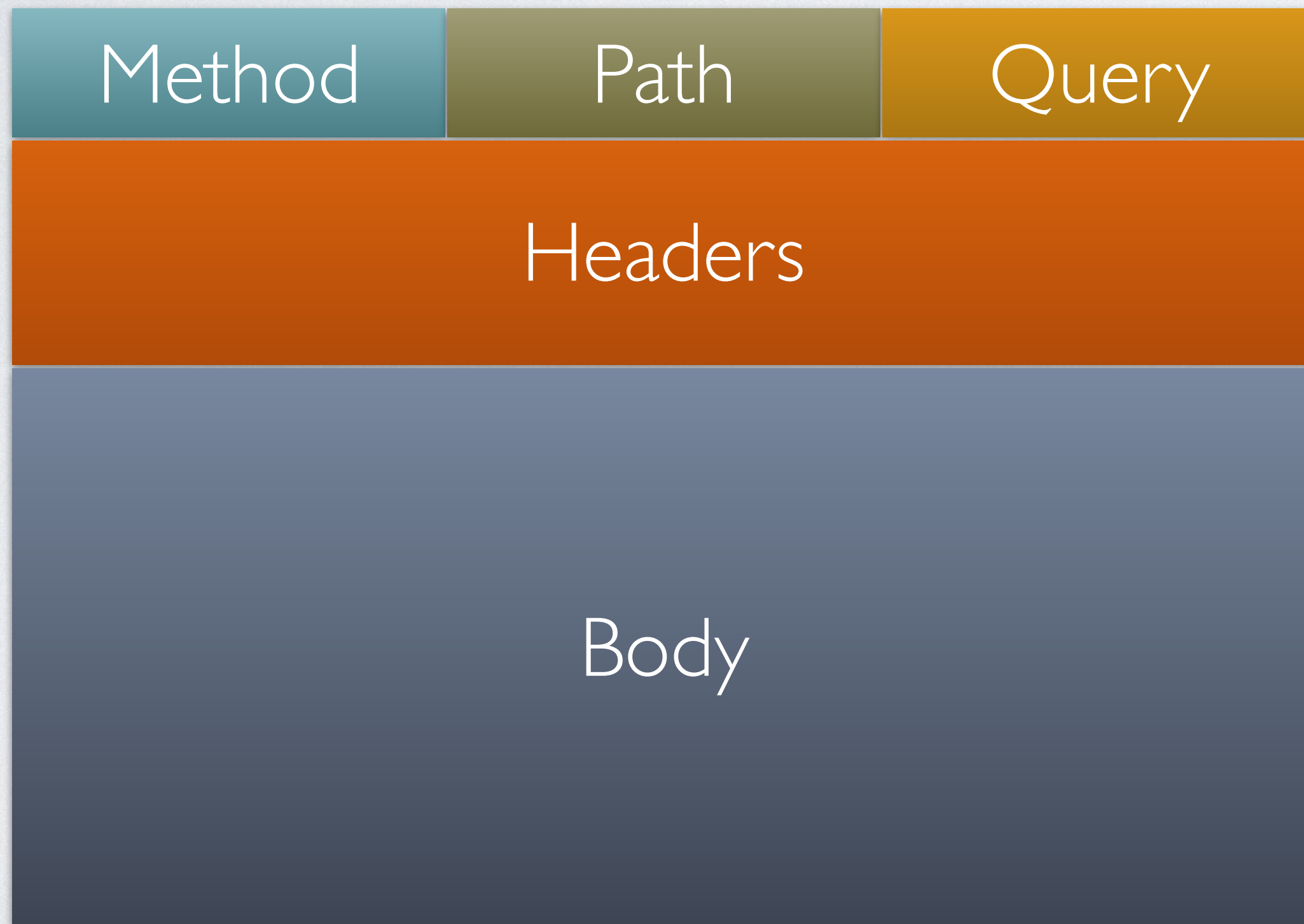
Server (Python)

WHAT IS A **REST** API?

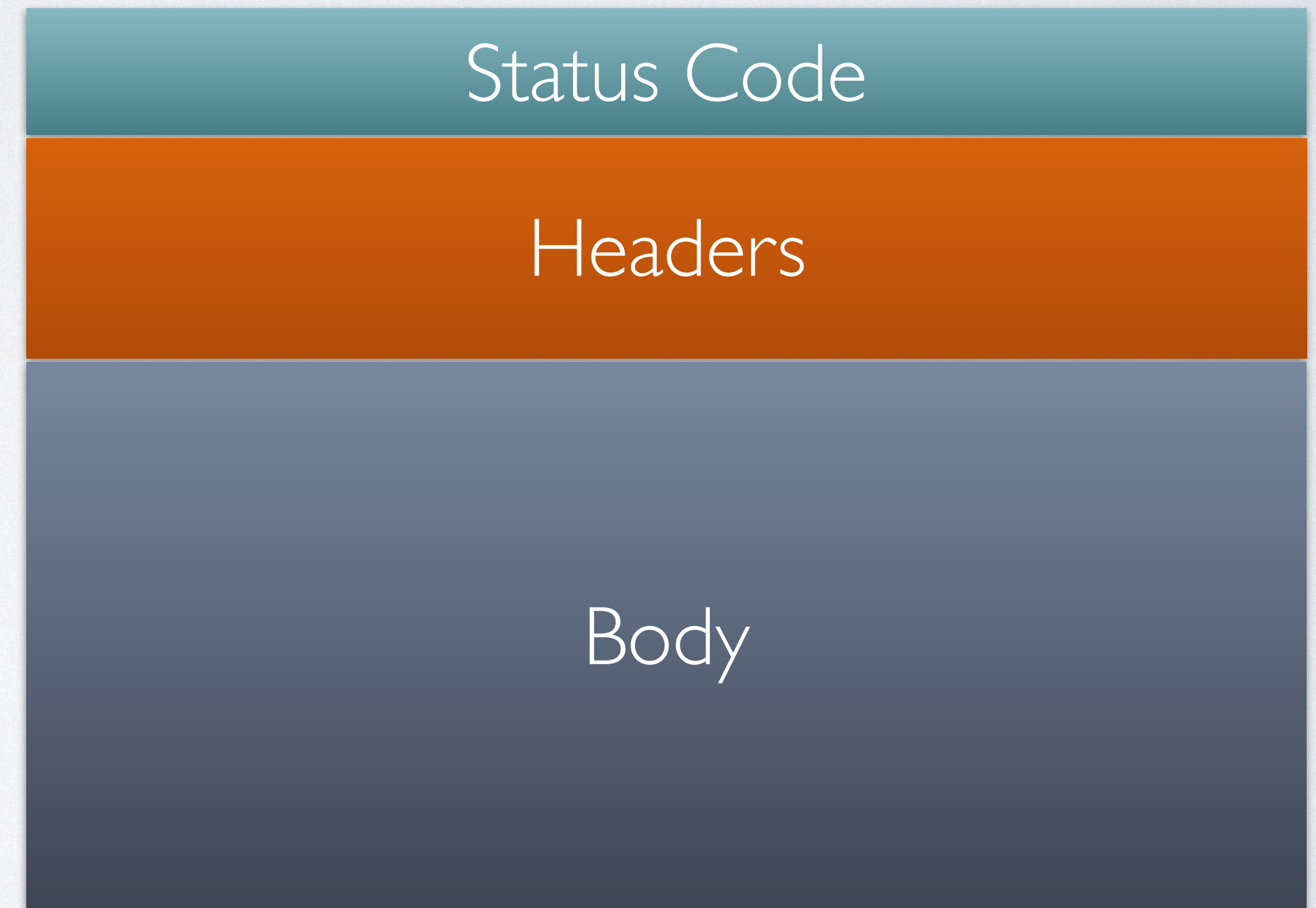
REPRESENTATIONAL STATE TRANSFER

A clever way to use **HTTP** to build APIs.

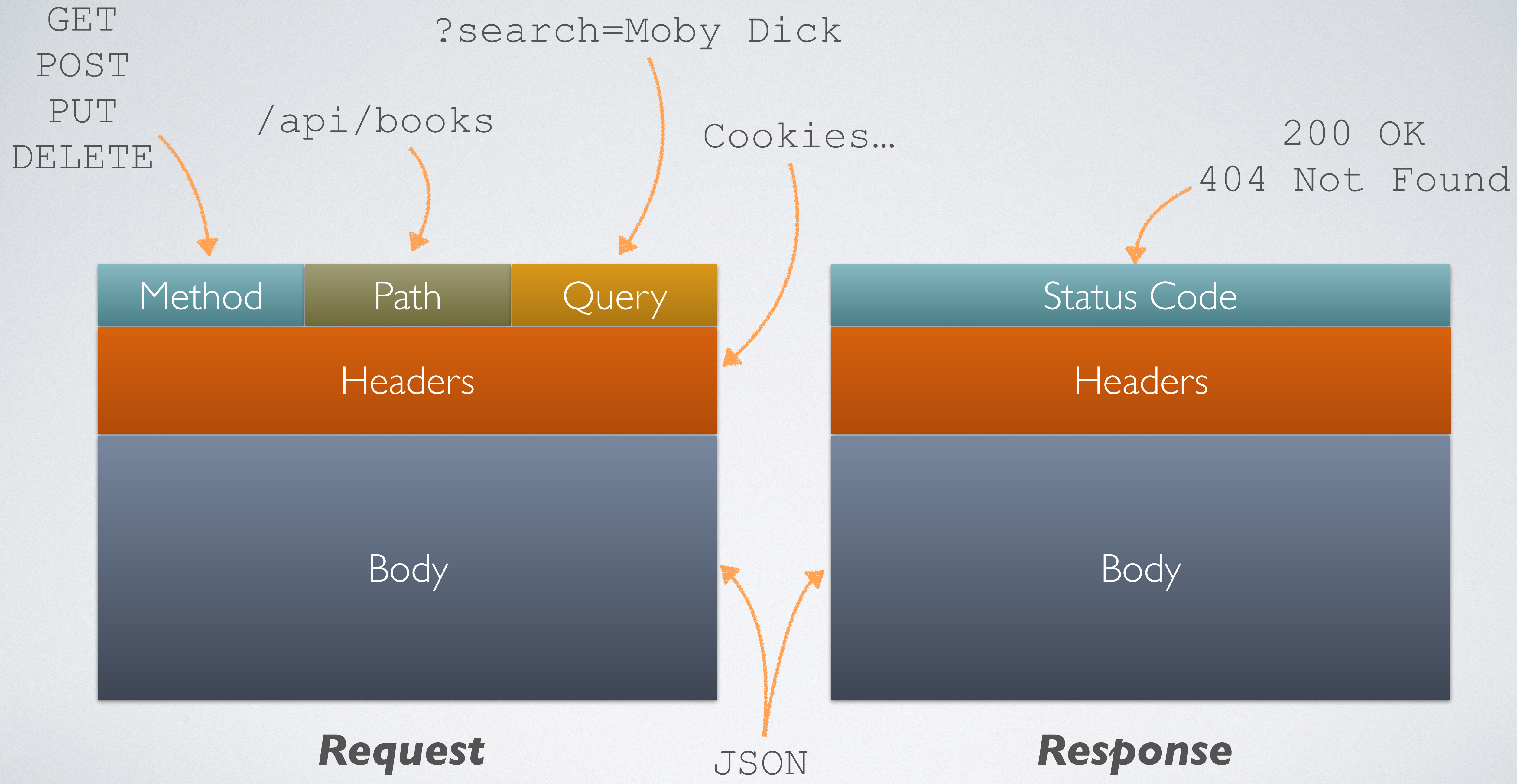
ANATOMY OF **HTTP**



Request



Response



REST CONVENTIONS

Method	Path	Query
Headers		
Body		

	GET	PUT	POST	DELETE
Collection /books	List books		New book	
Item /books/123	Display book	Update book		Delete book
Controller /books/123/borrow			Borrow book	

FLASK



Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

flask.pocoo.org

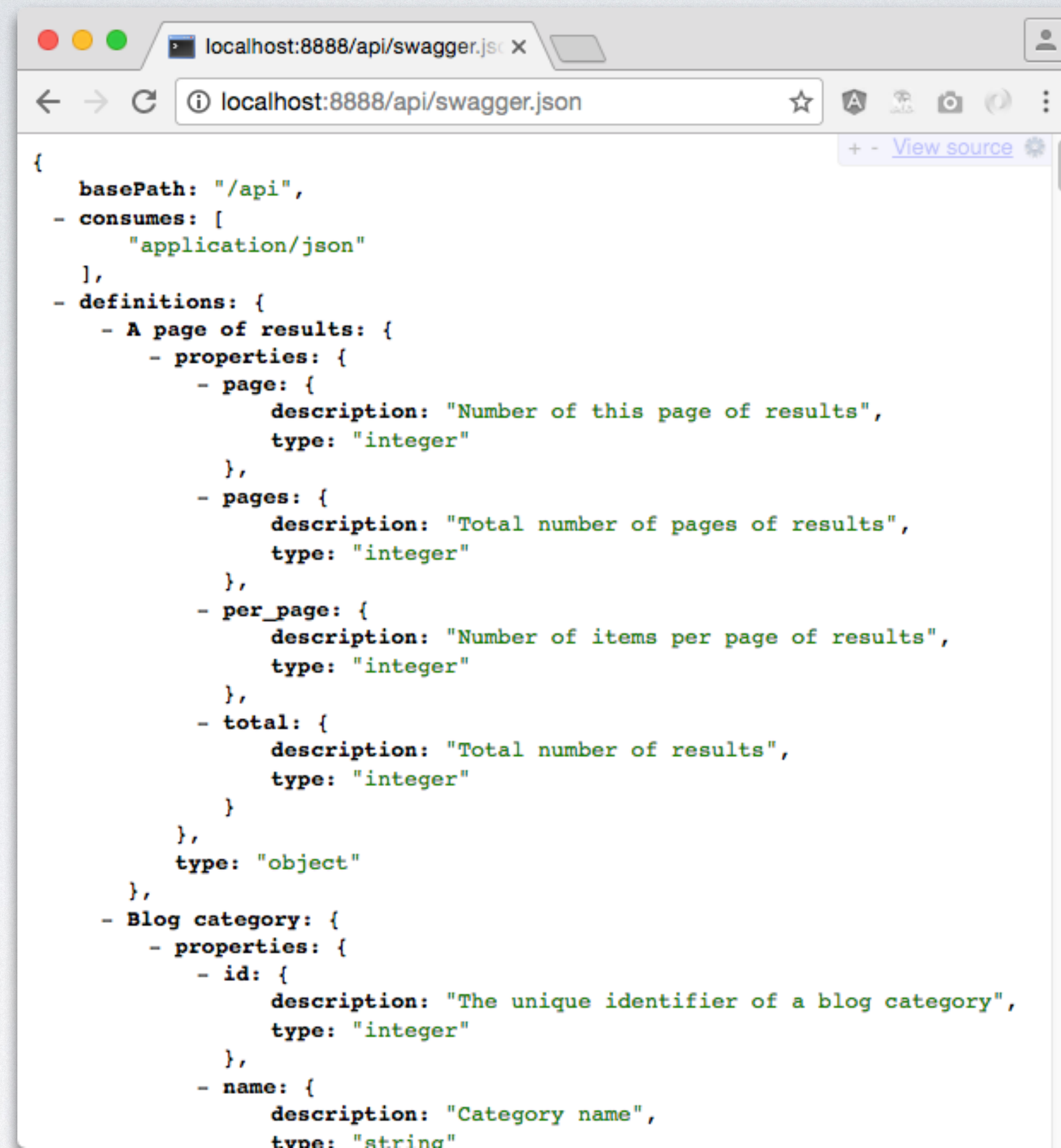
FLASK-RESTPlus



flask-restplus.rtfld.io

- define and document endpoints
- validate input
- format output (as JSON)
- turn Python exceptions into HTTP responses
- minimise boilerplate code
- generate interactive documentation (**Swagger UI**)

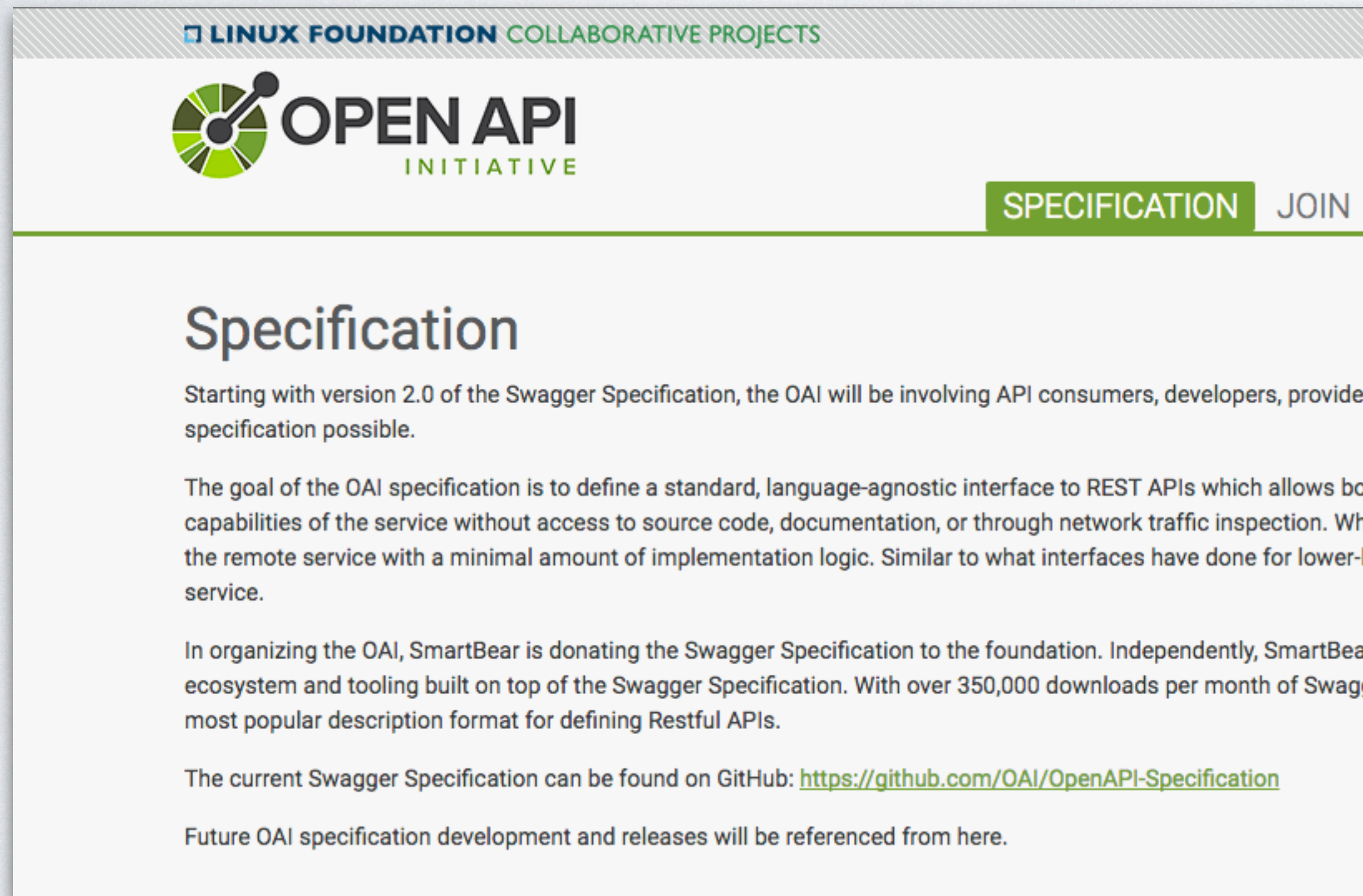
Demo



```
{
  basePath: "/api",
  - consumes: [
    "application/json"
  ],
  - definitions: {
    - A page of results: {
      - properties: {
        - page: {
          description: "Number of this page of results",
          type: "integer"
        },
        - pages: {
          description: "Total number of pages of results",
          type: "integer"
        },
        - per_page: {
          description: "Number of items per page of results",
          type: "integer"
        },
        - total: {
          description: "Total number of results",
          type: "integer"
        }
      }
    },
    type: "object"
  },
  - Blog category: {
    - properties: {
      - id: {
        description: "The unique identifier of a blog category",
        type: "integer"
      },
      - name: {
        description: "Category name",
        type: "string"
      }
    }
  }
}
```

OPEN API FORMAT

OPEN API SPECIFICATION



The screenshot shows the top section of the Open API Specification page. At the top, it says "LINUX FOUNDATION COLLABORATIVE PROJECTS". Below that is the "OPEN API INITIATIVE" logo, which consists of a green circular icon with a stylized 'O' and the text "OPEN API INITIATIVE". To the right of the logo is a green button labeled "SPECIFICATION" and a link labeled "JOIN". Below the navigation bar is a section titled "Specification" with a sub-heading "Starting with version 2.0 of the Swagger Specification, the OAI will be involving API consumers, developers, providers, and tooling to make the specification possible." The main text describes the goal of the OAI specification: "The goal of the OAI specification is to define a standard, language-agnostic interface to REST APIs which allows both clients and servers to understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When the client interacts with the remote service with a minimal amount of implementation logic. Similar to what interfaces have done for lower-level services." It also mentions that SmartBear is donating the Swagger Specification to the foundation and that the current Swagger Specification can be found on GitHub at <https://github.com/OAI/OpenAPI-Specification>. Finally, it states that future OAI specification development and releases will be referenced from here.

- Standard language to describe REST APIs
- Open source (Linux Foundation)
- Tools:
 - Swagger UI
 - Swagger Editor
 - Code generators
- Initiative with many powerful members

openapis.org

swagger.io

OPEN API SPECIFICATION

- Standard language to describe REST APIs
- Open source (Linux Foundation)
- Tools:

The screenshot shows the Open API Initiative website. At the top, it says "LINUX FOUNDATION COLLABORATIVE PROJECTS". Below that is the "OPEN API INITIATIVE" logo. There are two buttons: "SPECIFICATION" and "JOIN". The "Specification" page is selected. Below the header, there is a "Members" section with a grid of logos for various companies.

Members:

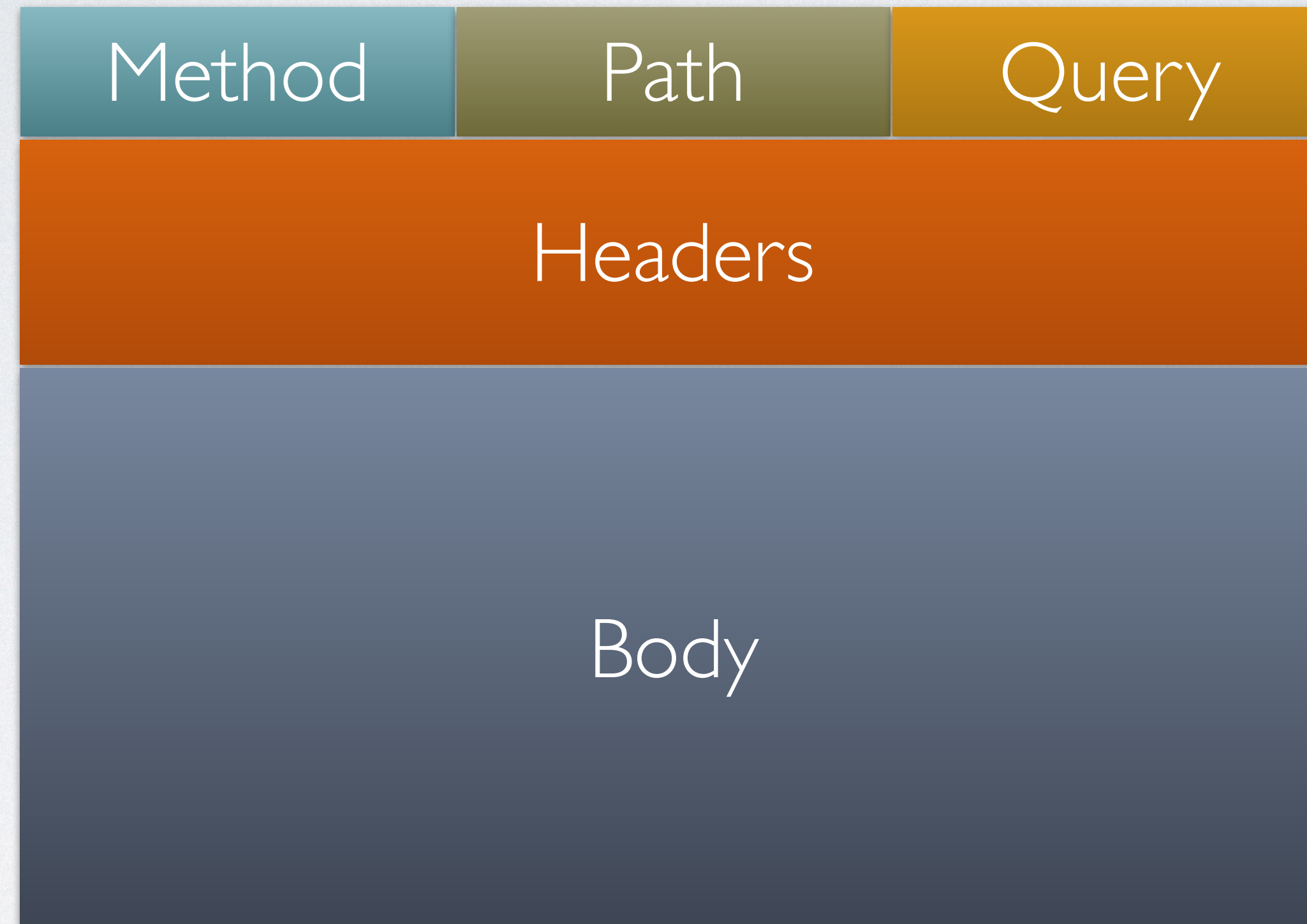
- 3scale
- apiary
- apigee
- apinf
- Atlassian
- Capital One
- Google
- Heart
- IBM
- intuit
- mashape
- Microsoft
- PayPal
- Restlet
- ARTIK Cloud
- SMARTBEAR

or

ators

many powerful members

swagger.io



Request

REQUEST METHOD

POST /api/books/123/borrow?when=today

Method	Path	Query
	Headers	
	Body	

```
from flask_restplus import Resource
```

```
@api.route('/<int:id>/borrow')
```

```
class BorrowBookController(Resource):
```

```
    def post(self, id):
```

```
        """ Borrow book from library.
```

```
        Allows the current user to borrow  
        the book out of the library.
```

```
        """
```

```
        ...
```

```
        return {'message': 'OK'}
```


REQUEST METHOD

POST /api/books/123/borrow?when=today

Method	Path	Query
	Headers	
	Body	

```
from flask_restplus import Resource
```

```
@api.route('/<int:id>/borrow')
```

```
class BorrowBookController(Resource):
```

```
def post(self, id):
```

```
    """ Borrow book from library.
```

```
    Allows the current user to borrow  
    the book out of the library.
```

```
    """
```

```
    ...
```

```
    return {'message': 'OK'}
```

```
class Resource:
```

```
def get(self)...
```

```
def post(self)...
```

```
def put(self)...
```

```
def delete(self)...
```

```
def patch(self)...
```

```
def options(self)...
```

```
def head(self)...
```


REQUEST METHOD

POST /api/books/123/borrow?when=today

Method	Path	Query
Headers		
Body		

POST /books/{id}/borrow Borrow book from library

Implementation Notes

Allows the current user to borrow the book out of the library.

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	<input type="text" value="(required)"/>		path	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Success		

[Try it out!](#)

REQUEST METHOD

POST /api/books/123/borrow?when=today

Method	Path	Query
Headers		
Body		

GET	/blog/categories/	Returns list of blog categories
POST	/blog/categories/	Creates a new blog category
DELETE	/blog/categories/{id}	Deletes blog category
GET	/blog/categories/{id}	Returns a category with a list of posts
PUT	/blog/categories/{id}	Updates a blog category

REQUEST PATH

POST **/api/books/123/borrow**?when=today

Method	Path	Query
Headers		
Body		

```
@api.route('/books/<int:id>/borrow')
```

```
@api.route('/articles/<title>')
```

```
@api.route('/wiki/<path:wikipage>')
```

```
@api.route('/values/<float:value>')
```

```
@api.route('/object/<uuid:identifier>')
```


REQUEST PATH

GET **/api/book/123/borrow**?when=today

Method	Path	Query
Headers		
Body		

Parameters

Parameter	Value	Description	Parameter Type	Data Type
wikipage	<input type="text" value="(required)"/>		path	string
title	<input type="text" value="(required)"/>		path	string
identifier	<input type="text" value="(required)"/>		path	string
id	<input type="text" value="(required)"/>		path	integer
value	<input type="text" value="(required)"/>		path	double

QUERY ARGUMENTS

GET /api/books?page=1&per_page=10

Method	Path	Query
	Headers	
	Body	

```
from flask_restplus import reqparse

pagination = reqparse.RequestParser()
pagination.add_argument('page', type=int, required=False,
                        default=1, help='Page number')
pagination.add_argument('per_page', type=int, required=False,
                        choices=[10, 20, 30, 40, 50])
```



QUERY ARGUMENTS

GET /api/books?page=1&per_page=10

Method	Path	Query
	Headers	
	Body	

```
from flask import request
from flask_restplus import Resource

@api.route('/')
class PostsCollection(Resource):
    @api.expect(parsers.pagination)
    def get(self):
        args = pagination_arguments.parse_args(request)
        page = args.get('page', 1)
        per_page = args.get('per_page', 10)
        ...
```



QUERY ARGUMENTS

GET /api/books?page=1&per_page=10

Method	Path	Query
Headers		
Body		

Parameters

Parameter	Value	Description	Parameter Type	Data Type
per_page	<input type="text" value="10 (default)"/>	Results per page	query	integer
page	<input type="text" value="1"/>	Page number	query	integer


Try it out!

REQUEST BODY (JSON)

API MODELS

Method	Path	Query
Headers		
Body		

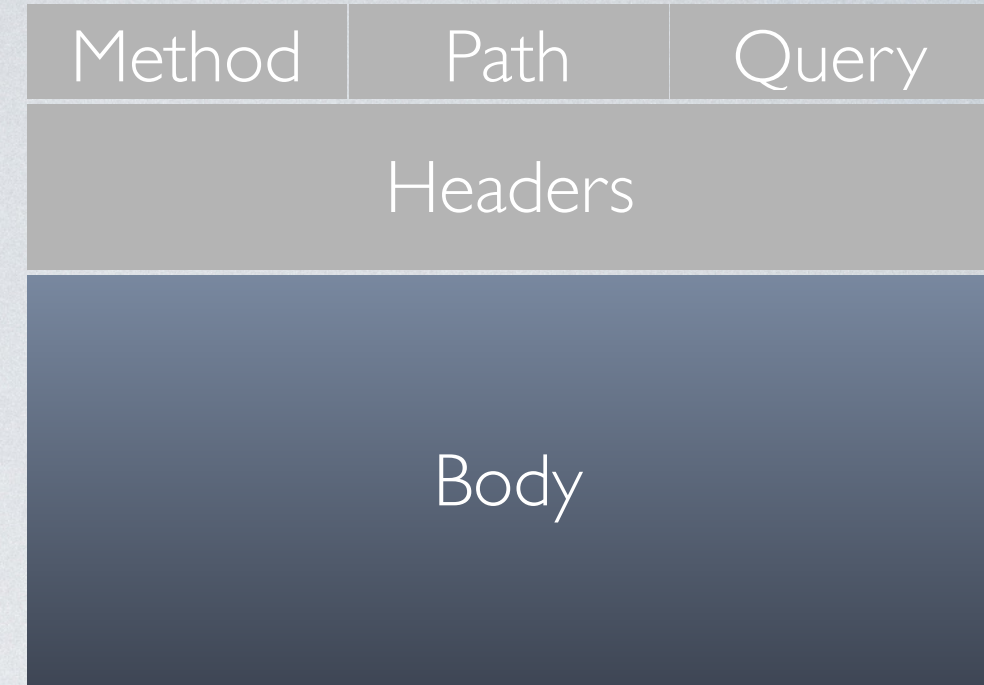
```
blog_post = api.model('Blog post', {  
    'title': fields.String(description='Article title'),  
    'body': fields.String(description='Article content'),  
    'pub_date': fields.DateTime,  
    'category_id': fields.Integer(min=1),  
})
```



```
@api.expect(blog_post)  
def post(self):  
    ...
```


REQUEST BODY (JSON)

API MODELS



Parameters

Parameter	Value	Parameter Type	Data Type
payload	<div style="border: 1px solid #ccc; padding: 5px; min-height: 150px;">(required)</div>	body	<p>Model Model Schema</p> <p>Blog post { body (string): Article content, category_id (integer, optional), pub_date (string, optional), title (string): Article title }</p>

Parameter content type:

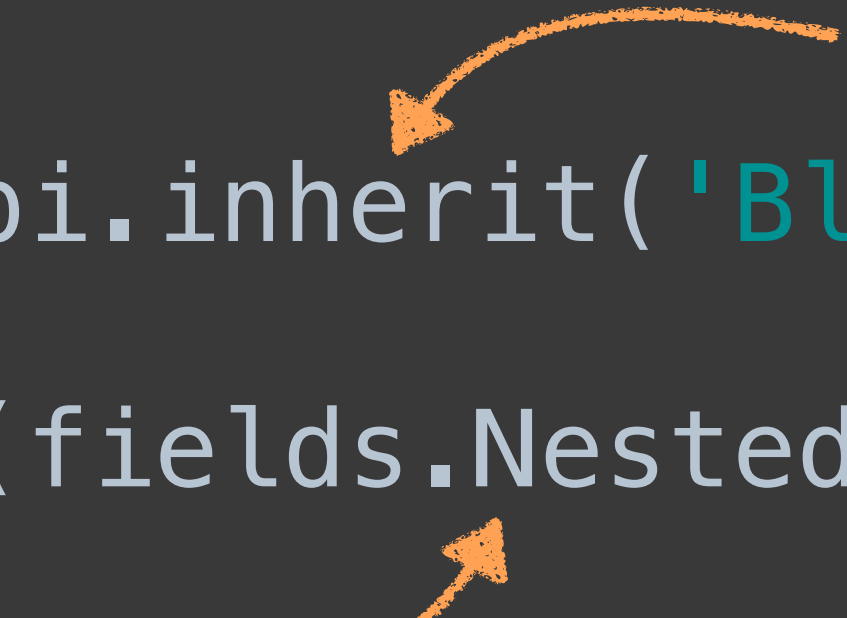
API MODELS

INHERITANCE AND NESTING

Method	Path	Query
Headers		
Body		

```
category = api.model('Blog category', {
    'id': fields.Integer(description='The unique id of category'),
    'name': fields.String(description='Category name'),
})

category_with_posts = api.inherit('Blog category with posts', category, {
    'posts': fields.List(fields.Nested(blog_post))
})
```



Status Code

Headers

Body

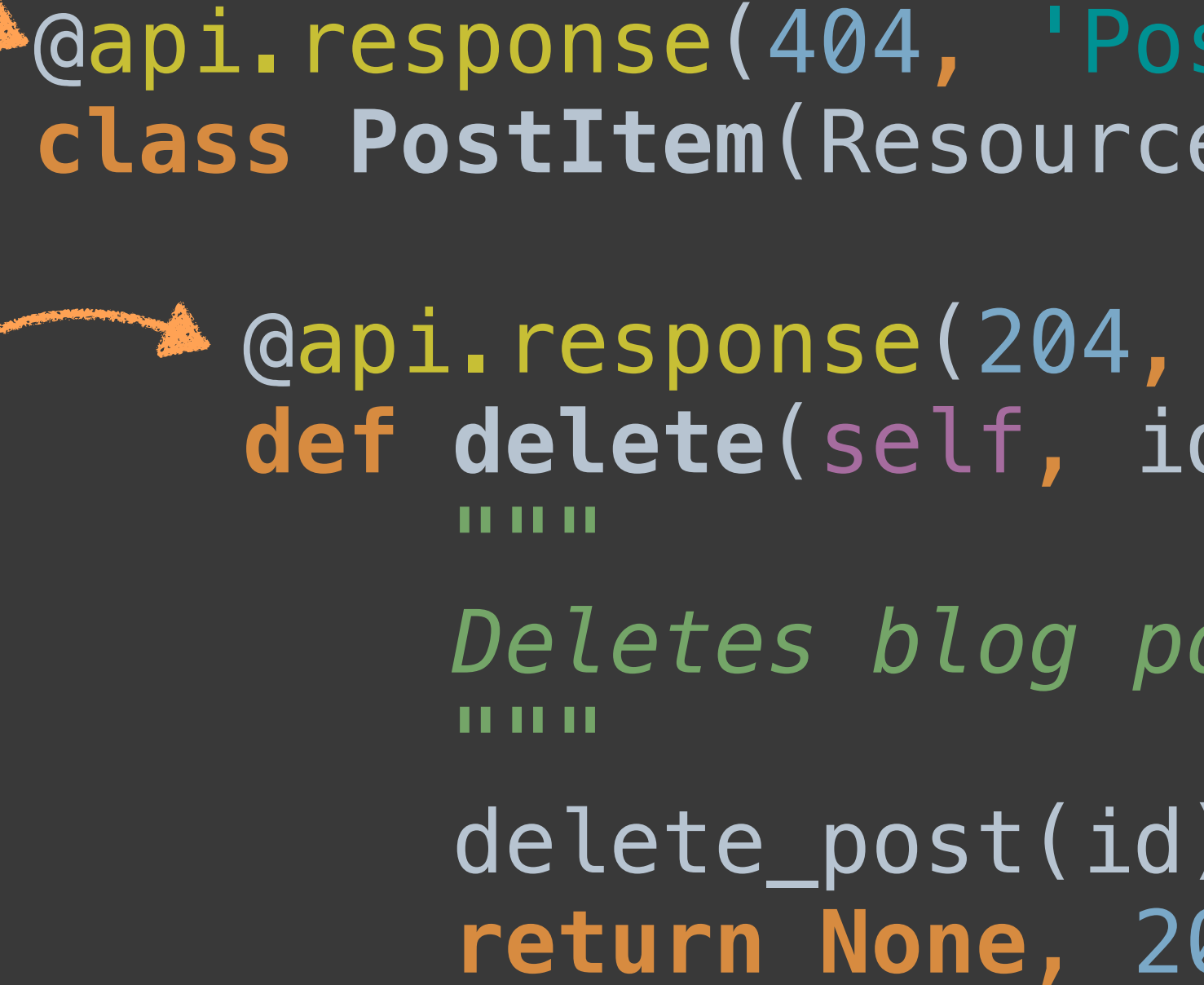
Response

RESPONSE STATUS CODE

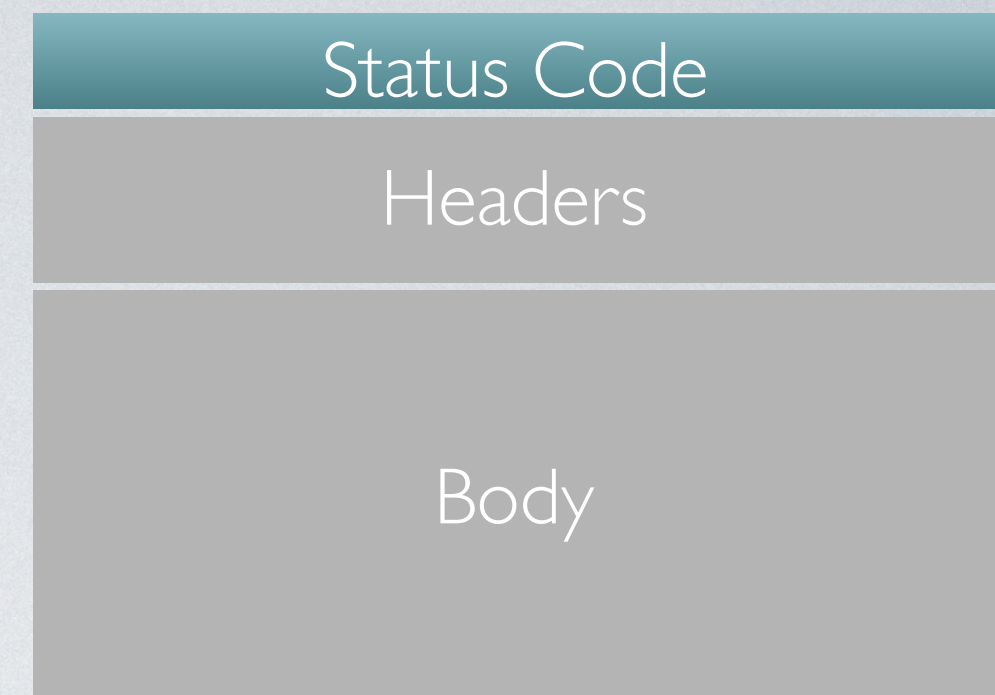
Status Code
Headers
Body

```
@api.route('/<int:id>')
@api.response(404, 'Post not found.')
class PostItem(Resource):

    @api.response(204, 'Post successfully deleted.')
    def delete(self, id):
        """
        Deletes blog post.
        """
        delete_post(id)
        return None, 204
```



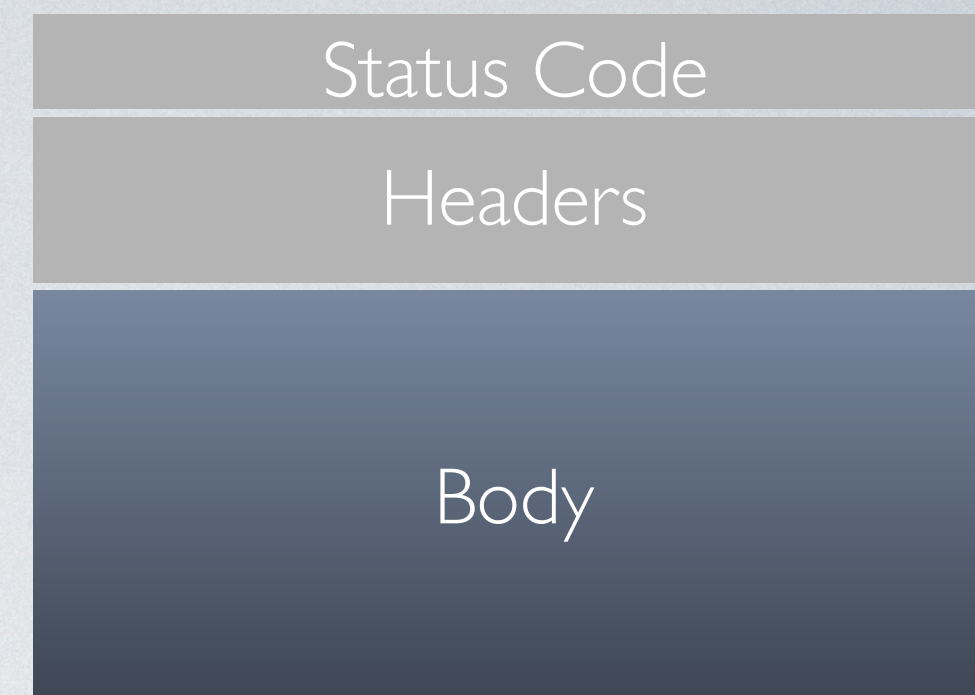
RESPONSE STATUS CODE



Response Messages

HTTP Status Code	Reason
204	Post successfully deleted.
404	Post not found.

RESPONSE BODY (JSON)



```
blog_post = api.model('Blog post', {  
    ...  
    'category_id': fields.Integer(attribute='category.id'),  
    'name': fields.String(attribute=lambda x: x._private_name),  
})
```

```
@api.marshal_with(blog_post)  
def get(self):  
    ...
```

```
@api.marshal_list_with(blog_post)  
def get(self):  
    ...
```


EXCEPTION HANDLING

```
from sqlalchemy.orm.exc import NoResultFound

@api.errorhandler(NoResultFound)
def database_not_found_error_handler(e):
    log.warning(traceback.format_exc())
    return {'message': 'A database result was not found.'}, 404
```


INTERACTIVE DEBUGGER

builtins.NameError

NameError: name 'username' is not defined

Traceback (most recent call last)

File "/path/to/rest_api_demo/venv/lib/python3.5/site-packages/flask/views.py", line 84, in view

```
return self.dispatch_request(*args, **kwargs)
```

File "/path/to/rest_api_demo/venv/lib/python3.5/site-packages/flask_restplus/resource.py", line 44, in dispatch_request

```
resp = meth(*args, **kwargs)
```

File ""/path/to/rest_api_demo/venv/lib/python3.5/site-packages/flask_restplus/marshalling.py", line 101, in wrapper

```
resp = f(*args, **kwargs)
```

File "/path/to/rest_api_demo/rest_api_demo/api/blog/endpoints/posts.py", line 25, in get

```
return("Hello" + username)
```

[console ready]

```
>>> username
```

Traceback (most recent call last):

File "<debugger>", line 1, in <module>

```
username
```

NameError: name 'username' is not defined

```
>>> |
```

NameError: name 'username' is not defined


```
from flask import Flask
from flask_restplus import Resource, Api

app = Flask(__name__)
api = Api(app)

@api.route('/hello')
class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

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


Demo code and article available on my blog:

karzyn.com

THANK YOU