The Falcon Web Framework - Documentation

What is the Falcon framework?

Falcon is a WSGI-compliant web framework designed to build RESTful APIs without requiring external code library dependencies.

Falcon is a reliable, high-performance Python web framework for building large-scale app backends and microservices. Falcon apps work with any WSGI server.

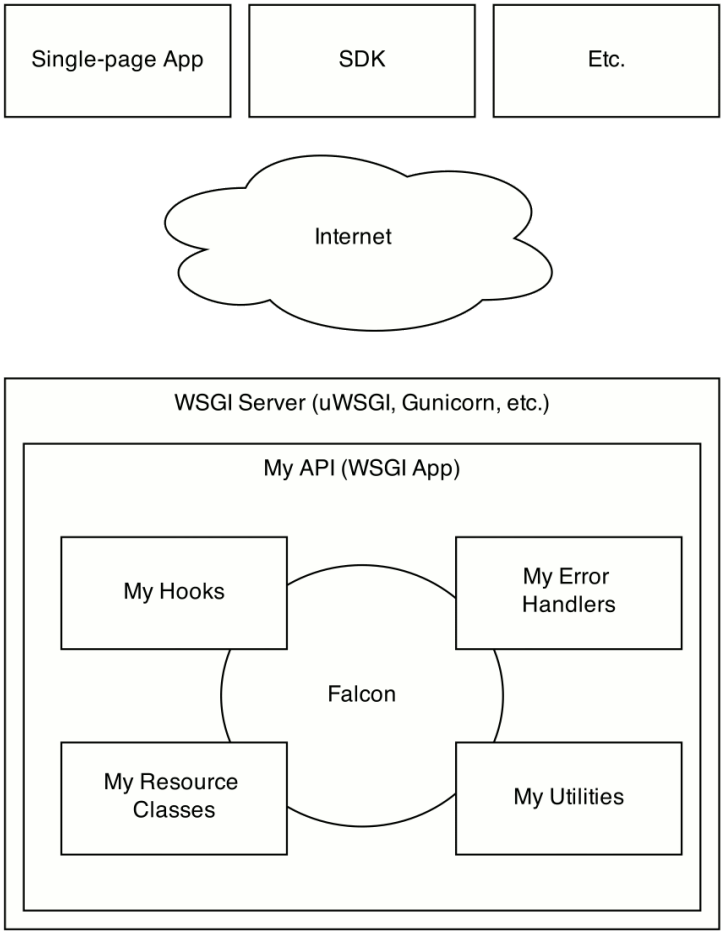
Why should I use Falcon?

Falcon is as powerful as other python's frameworks like flask, Django, web.py and it is fast, reliable, extensible. Falcon is 21 times faster on pypy than flask and Django.

How is Falcon different?

Fast : Falcon turns around requests several times faster than most other Python frameworks.
Reliable : The code is rigorously tested with numerous inputs. Falcon does not depend on any external Python packages.
Flexible : Falcon leaves a lot of decisions and implementation details to the API developer. This gives a lot of freedom to customize and tune implementation.
Debuggable : Easy to tell which inputs lead to which outputs. Unhandled exceptions are never encapsulated or masked.Well documented. Framework keeps logic paths simple and understandable.

Falcon Architecture



WSGI Server

A Web Server Gateway Interface (WSGI) server implements the web server side of the WSGI interface for running Python web applications.

Hooks

Falcon supports before and after hooks. You can install a hook simply by applying decorators, either to an individual responder or to an entire resource. Hooks in Falcon is its a trigger you can handle before and after your single function.

Error Handler

Falcon tries to make things a bit easier and more consistent by providing a set of error classes you can raise from within your app. Falcon catches any exception that inherits from `falcon.HTTPError`, and automatically converts it to an appropriate HTTP response.

Utilities

URI utilities module provides utility functions to parse, encode, decode, and otherwise manipulate a URI. These functions are not available directly in the falcon module, and so must be explicitly imported like:

```
From falcon import uri
```

Resource classes

A resource in Falcon is just a regular Python class that includes one or more methods representing the standard HTTP verbs supported by that resource. Each requested URL is mapped to a specific resource.

Features

- Clean and extensible code base
- Simple and quick access to headers and bodies
- Simple and effective exception handling
- Support for various versions of og Cpython, PyPy and Jython

Installation

The installation of Falcon is simple and quick. It can be installed using Pip, as below:

```
$  pip install falcon
```

Web Server Gateway Interface (WSGI)

Falcon communicates through WSGI, and so in order to serve a Falcon app, you will need a WSGI server. Gunicorn, uWSGI and Waitress are some WSGI servers.

```
$  pip install [ gunicorn | uwsgi | waitress ]
```

Example

```
import falcon, json

class ObjRequestClass(object):
    def on_get(self, req, res):
        res.status = falcon.HTTP_200

        data = json.loads(req.stream.read())

        content = {
            'name': 'Pooja',
            'age': '23',
            'country': 'India'
        }

        output = {}

        if('method' not in data):
            res.status = falcon.HTTP_501
            output['value'] = 'Error: none method found - sorry'
        else:
            if(data['method'] == 'get_name'):
                output['value'] = content['name']
            else:
                res.status = falcon.HTTP_404
                output['value'] = None

        res.body = json.dumps(output)

api = falcon.API()

obj = ObjRequestClass()

api.add_route('/test',obj)
```

You can run code using any WSGI server, such as Gunicorn or uWSGI or Waitress server
For example :

```
$  waitress-serve --port=8000 app:api
```

You can use GET request as below:

```
http://localhost:8000/test
```

Advantages

- Falcon is OS agnostic and focusing on running efficiently on any given hardware, with framework flexibility.
- REST HTTP handlers provide request resolutions and easy state transition.
- Falcon uses only two third-party dependencies.
- Falcon can make up to 19x more requests per second than Django under identical conditions.

Disadvantages

- Falcon is not suitable for serving HTML pages.
- Lacks a built-in web server.

Comparison

A rough performance comparison between the given frameworks, and to demonstrate the baseline per-request overhead each framework incurs on an application. The frameworks were tested on several different versions of Python.

CPython 2.2.16

	speedup	**req/sec**	**µs/req**
Falcon Cythonized (2.0.0)	27x	78,432	12.75
Falcon (2.0.0)	17x	48,930	20.44
Bottle (0.12.16)	9x	25,946	38.54
Werkzeug (0.15.4)	3x	7,535	132.72
Flask (1.0.2)	1x	4,382	228.22
Django (1.11.20)	1x	2,945	339.55

CPython 3.7.3

	speedup	req/sec	µs/req
Falcon Cythonized (2.0.0)	41x	72,679	13.76
Falcon (2.0.0)	28x	50,030	19.99
Bottle (0.12.16)	15x	27,281	36.66
Werkzeug (0.15.4)	5x	9,698	103.11
Flask (1.0.2)	3x	5,404	185.06
Django (2.2.1)	1x	1,790	558.60

PyPy2 7.1.1

	speedup	req/sec	µs/req
Falcon (2.0.0)	20x	461,399	2.17
Bottle (0.12.16)	12x	290,709	3.44
Werkzeug (0.15.4)	4x	100,040	10.00
Django (1.11.20)	1x	31,213	32.04

Flask (1.0.2)	1x	23,492	42.57

PyPy3 7.0.0

	speedup	**req/sec**	**µs/req**
Falcon (2.0.0)	75x	330,676	3.02
Bottle (0.12.16)	37x	162,274	6.16
Werkzeug (0.15.4)	11x	47,395	21.10
Flask (1.0.2)	8x	36,746	27.21
Django (2.2.1)	1x	4,436	225.41