RETRY BEST PRACTICES

Fabio Fleitas & Peter Hadlaw @ Tesorio

August 11, 2017

https://www.tesorio.com/careers

FETCH GITHUB USERS

```
import requests

def fetch_github_users():
    url = 'https://api.github.com/users'
    while url:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()
        for user in data:
            yield user

        url = next_page_url(response)
```

```
$ pip install tenacity
```

RETRY ... EVERYTHING

```python
from tenacity import retry

@retry
def fetch_github_users():
    url = 'https://api.github.com/users'
    while url:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()
        for user in data:
            yield user

        url = next_page_url(response)
```

RETRY INDIVIDUAL REQUESTS

```python
@retry
def get_data(url):
    response = requests.get(url)
    response.raise_for_status()
    return {
        'results': response.json(),
        'next_url': next_page_url(response),
    }

def fetch_github_users():
    url = 'https://api.github.com/users'
    while url:
        data = get_data(url)
        for user in data['results']:
            yield user

        url = data['next_url']
```

RATE LIMITING

```python
class RateLimitExceededError(Exception):
    pass

def validate_response(response):
    is_rate_limited = (
       response.headers['X-RateLimit-Remaining'] == '0')

    if is_rate_limited:
        raise RateLimitExceededError()

    response.raise_for_status()
```

```python
from tenacity import retry_if_exception_type, wait_fixed

@retry(
    retry=retry_if_exception_type(RateLimitExceededError),
    wait=wait_fixed(10))
def get_data(url):
    response = requests.get(url)
    validate_response(response)
    # ...
```

BETTER RATE LIMITING

```python
import time

def validate_response(response):
    # ...
    if is_rate_limited:
        rate_limit_reset_time = (
            int(response.headers['X-RateLimit-Reset']))

        time_to_sleep = rate_limit_reset_time - time.time()
        time.sleep(time_to_sleep)

        raise RateLimitExceededError()
    # ...
```

GitHub is down...

```python
class GitHubIsDownError(Exception):
    pass

def validate_response(response):
    # ...

    github_is_down = response.status_code == 500
    if github_is_down:
        raise GitHubIsDownError()

    # ...
```

```python
from tenacity import wait_exponential

@retry(retry=retry_if_exception_type(RateLimitExceededError))
@retry(
    retry=retry_if_exception_type(GitHubIsDownError),
    # Wait 2^x * 1 second between each retry
    wait=wait_exponential(multiplier=1))
def get_data(url):
    # ...
```

RETRYING POST REQUESTS

The goal is to have idempotent POST requests.

IDEMPOTENCY

In computer science, the term idempotent is used more comprehensively to describe an operation that will produce the same results if executed once or multiple times.

More details: https://stripe.com/blog/idempotency

```python
import uuid
import stripe
stripe.api_key = "APIKEY"

@retry(retry=retry_if_exception_type(RateLimitExceededError))
@retry(
    retry=retry_if_exception_type(StripeIsDownError),
    wait=wait_exponential(multiplier=1))
def send_request(idempotency_key, **kwargs):
    return stripe.Charge.create(
        idempotency_key=idempotency_key, **kwargs)
```

```python
def charge_customer():
    idempotency_key = uuid.uuid4()
    return send_request(
        idempotency_key,
        amount=9001,
        currency="usd",
        description="Charge for goku@dbz.com",
        source="tok_mastercard",  # obtained with Stripe.js
    )
```

WE'RE HIRING

https://www.tesorio.com/careers