RQ w praktyce

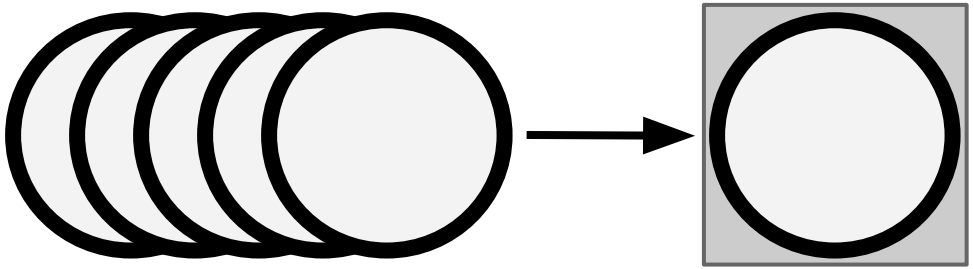
Co jest fajne? Z czy mogą być problemy...

Andrzej Jankowski

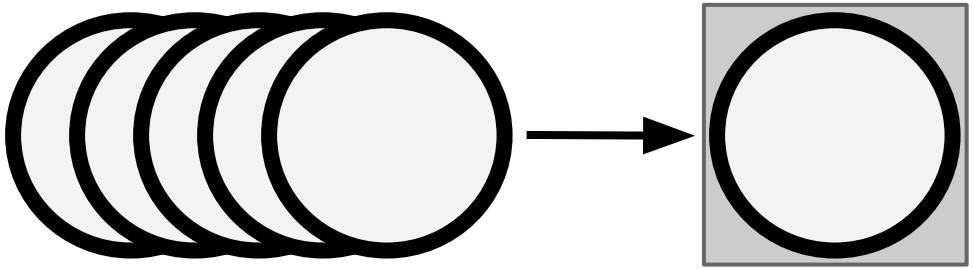
Czym jest RQ?

Czym jest RQ?

Czym jest RQ?



Czym jest RQ?



Co chcemy uzyskać?

Co chcemy uzyskać?

- Zlecić zadanie do wykonania

Co chcemy uzyskać?

- Zlecić zadanie do wykonania
- Wykonać zadanie - WORKER

Co chcemy uzyskać?

- Zlecić zadanie do wykonania
- Wykonać zadanie - WORKER
- Obsłużyć błędy

Zabawa z zadaniami

```
from redis import Redis
from rq import Queue

def foo(a, b):
    return "%s - %s" % (a, b)

q = Queue(connection=Redis())

job = q.enqueue(foo, "ala", "ola")
job.result
```

Zabawa z zadaniami

```
from rq.decorators import job

@job("queue name", connection=my_conn)
def foo(a, b):
    return "%s - %s" % (a, b)

foo.delay("ola", "ala")
```

Zabawa z zadaniami

```
# v0.4.0
queue = Queue()

calculate_job = queue.enqueue(
    calculate_something,
)
queue.enqueue(
    send_results,
    depends_on=calculate_job,
)
```

Timeout, TTL

- timeout
  - JobTimeoutException
  - timeout a oczekiwanie na wykonanie...
- result_ttl

Timeout, TTL

```
queue.encueue_call(
    func=func_to_call,
    args=tuple(),
    kwargs=dict(),
    timeout=60,
    result_ttl=60,
)
```

Worker

```
from redis import Redis
from rq import Queue, Connection, Worker

connection = Redis()
with Connection(connection):
    worker = Worker(Queue(name="abc"))
    worker.work()
```

META

```
def foo():
    job = get_current_job()
    job.meta.setdefault(
        "messages", [],
    ).append("Message 1")
    job.save()
```

META

```
q = Queue(Redis())
job = q.enqueue(foo)
# wait...
job.meta  # => {"messages": ["Message 1"]}

job.meta.setdefault("messages", []).append(
    "Message 2",
)
job.save()
```

Co z błędami?

Co z błędami?

- Kolejka failed

Co z błędami?

- Kolejka failed
- Własna obsługa błędów

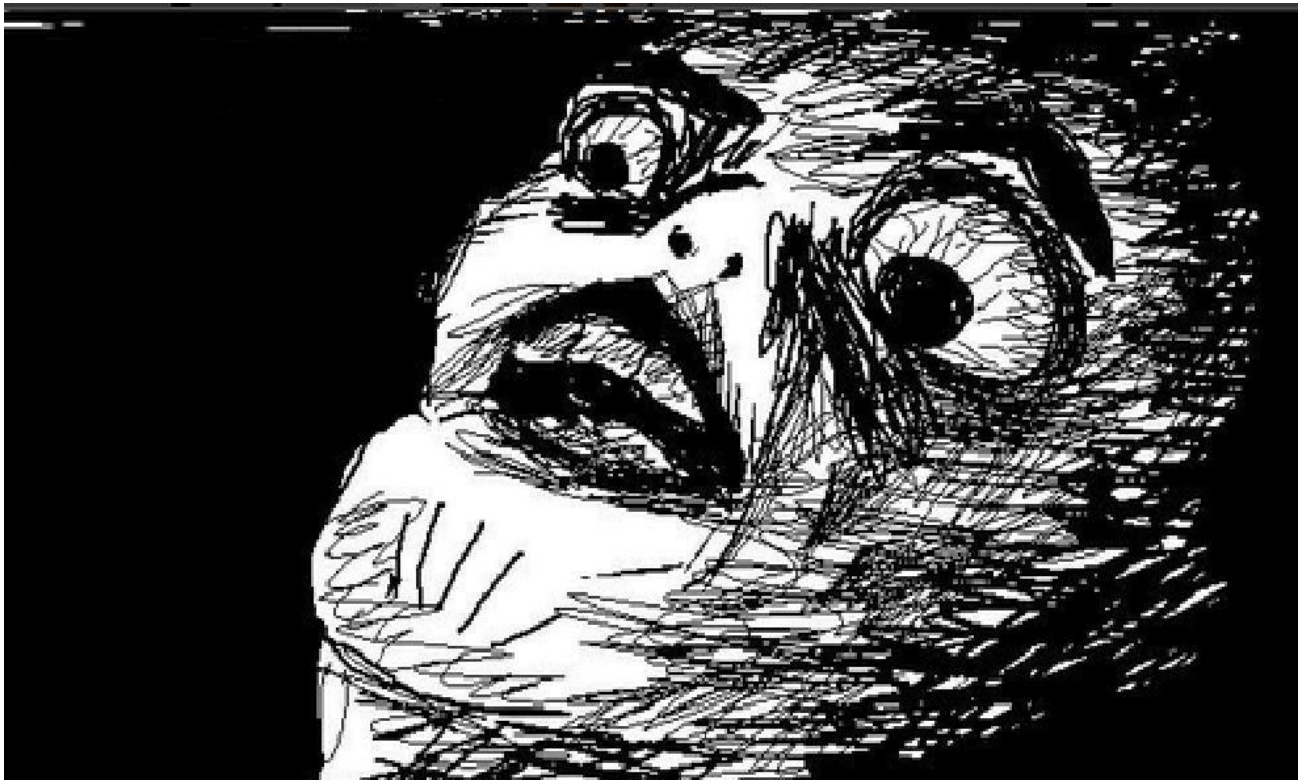
Co z błędami?

```
def custom_handler(
    job,
    exc_type,
    exc_value,
    traceback
):
    # do something
    return False  # stop processing
```

Co z błędami?

```
with Connection():
    worker = Worker(Queue())
    worker.push_exc_handler(custom_handler)
    worker.work()
```

Workery umierają...



Workery umierają...

```
# worker core
while True:
    # ...
    job = self.dequeue_job_and_maintain_ttl(...)
    # ...
    self.heartbeat((job.timeout or 180) + 60)
    self.fork_and_perform_job(job)
    self.heartbeat()
    # ...
```

Workery umierają...

```
# worker core
def heartbeat(self, timeout=0):
    timeout = max(timeout, self.default_worker_ttl)
    self.connection.expire(self.key, timeout)
```

Workery umierają...

```
# worker core
def fork_and_perform_job(self, job):
    # ...
    # fork()
    # ...
    self.perform_job(job)
```

Workery umierają...

```
# worker core
def perform_job(self, job):
    # ...
    with death_penalty_after(job.timeout or 180):
        job_result = job.perform()
    # ...
```

Workery umierają...

```
# !!!
try:
    # do something
except:
    pass
```

Dziękuję za uwagę.

http://python-rq.org/
https://github.com/andrzej-jankowski/rq-presentation