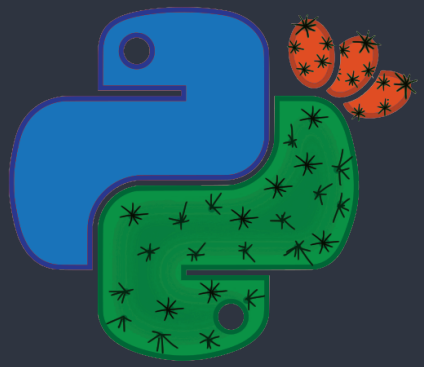

Liberate your API

Building a task manager inside Sanic

Adam Hopkins

```
start = datetime(2021, 5, 3, 10, 0, 0, tzinfo=ZoneInfo(key="Asia/Jerusalem"))
end = start + timedelta(minutes=25)
```

```python
class Adam:

    def __init__(self):
        self.work = PacketFabric("Sr. Software Engineer")
        self.oss = Sanic("Core Maintainer")
        self.home = Israel("Negev")

    async def run(self, inputs: Union[Pretzels, Coffee]) -> None:
        while True:
            await self.work.do(inputs)
            await self.oss.do(inputs)

    def sleep(self):
        raise NotImplemented
```

- PacketFabric - Network-as-a-Service platform; private access to the cloud; secure connectivity between data centers
- Sanic Framework - Python 3.7+ `asyncio` enabled framework and server. Build fast. Run fast.
- GitHub - /ahopkins
- Twitter - @admhpkns

We've built our new web API ...

We've built our new web API ...

Our problem…

… framework

… server

… hosting

… code

… slow operations

Our problem…

… framework

… server

… hosting

… code

… slow operations

```
async def slow_stuff():
    data = await go_fetch_a_ton_of_data()
    computed = now_run_a_ton_of_computations(data)
```

Potential solutions:

1. Third-party package: `celery.send_task("execute_slow_stuff")`

Potential solutions:

1. Third-party package: `celery.send_task("execute_slow_stuff")`
2. Background tasks: `app.add_task(slow_stuff())`

Potential solutions:

1. Third-party package: `celery.send_task("execute_slow_stuff")`
2. Background tasks: `app.add_task(slow_stuff())`
3. In process task queue

Potential solutions:

1. Third-party package: `celery.send_task("execute_slow_stuff")`
2. Background tasks: `app.add_task(slow_stuff())`
3. In process task queue
4. Subprocess task queue

Option 1: Third-party package

```python
from celery import Celery

@app.before_server_start
def setup_celery(app, _):
    request.app.ctx.celery = Celery(...)

@app.post("/start_task")
async def start_task(request):
    task = request.app.ctx.celery.send_task(
        "execute_slow_stuff",
        kwargs=request.json
    )
    return text(f"Started task with {task.id=}", status=202)
```

Option 1: Third-party package

```
@app.post("/check/<task_id:uuid>")
async def check_status(request, task_id: UUID):
    result = app.AsyncResult(task_id)
    result.get()
    serialized = my_serializer(result)
    return json(serialized)
```

Option 2: Background tasks

```
async def send_email(user: User):
    ...

@app.post("/registration")
async def user_registration(request):
    user = await do_user_registration(request.json)
    request.app.add_task(send_email(user))
    return text("Welcome! 👏")
```

Option 3: In process task queue

```
@app.after_server_start
async def setup_task_executor(app, ):
    app.ctx.queue = asyncio.Queue(maxsize=64)
    for x in range(app.config.NUM_TASK_WORKERS)):
        app.add_task(worker(f"Worker-{x}", app.ctx.queue))
```

Option 3: In process task queue

```python
@app.after_server_start
async def setup_task_executor(app, ):
    app.ctx.queue = asyncio.Queue(maxsize=64)
    for x in range(app.config.NUM_TASK_WORKERS)):
        app.add_task(worker(f"Worker-{x}", app.ctx.queue))
```

```python
async def worker(name, queue):
    while True:
        job = await queue.get()
        if not job:
            break
        size = queue.qsize()
        logger.info(f"[{name}] Running {job}. {size} in queue.")
        await Job.create(job)
        await asyncio.sleep(0.1)
```

Option 3: In process task queue

```
@app.post("/start_task")
async def start_task(request):
    await request.app.ctx.queue.put("execute_slow_stuff")
    return text("Started task", status=202)
```

Option 3: In process task queue

```
@app.post("/start_task")
async def start_task(request):
    uid = uuid.uuid4()
    await request.app.ctx.queue.put(
        {
            "name": "execute_slow_stuff",
            "uid": uid,
        }
    )
    return text(f"Started task with {uid=}", status=202)
```

Option 4: Subprocess task queue

Requirements:

- Generate unique IDs at execution
- State of job needs to be stored
- Query job state with unique ID
- Cannot block HTTP request/response cycle

Meet SAJE

Sanic Asynchronous Job Executor

Questions?

GitHub - /ahopkins

Twitter - @admhpkns

PacketFabric - packetfabric.com

Sanic homepage - sanicframework.org

Sanic repo - /sanic-org/sanic