Next-Gen Django Background Workers

— Django Meetup Vol. 56 —

Andreas Schmitz

2024-07-16

$ whoami

Andreas Schmitz

- Lead Software Development Engineer at …
    - ✉️ me@andreas.earth
    - 🌍 www.andreas.earth
- … wirbauen.digital
    - 🏗 Construction-Tech Startup
    - 📍 Cologne, Germany
    - 🚀 Trying to digitize the Construction Industry

wirbauen .digital

A Common Problem

Run long running tasks outside the request-response lifecycle.

Current Landscape

Background Workers

Solution: Install a third-party solution

- Celery
- Django Q or Django Q2
- Huey
- …

Third-Party Solutions

Background Workers

The Problem

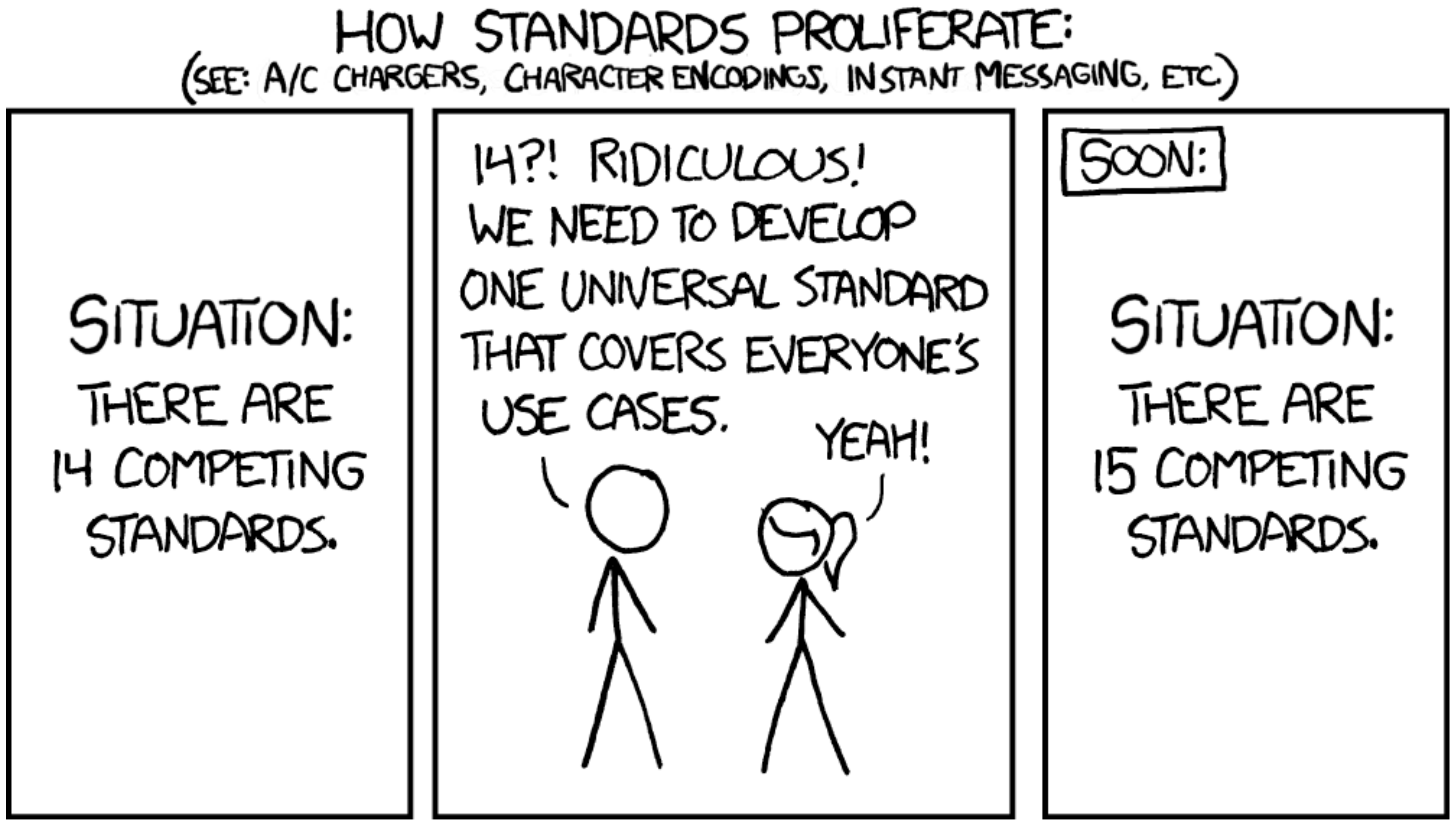
- Yet another dependency
- No common interface
- Hard to migrate from one to another

🙄

Background Workers

DEP 0014

Yet Another Background Worker?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Source XKCD: https://xkcd.com/927

Background Workers

Why another one?

A competing standard or a solution?

Project Goal

Interface and **base** implementation for long-running background tasks **shipped with** Django.

Background Workers

Project Goal

- Common Interface
  - "task backend" interface specification
  - For Third-Party Libraries & Your Code
- Base Implementation
  - ImmediateBackend — Run tasks immediately
  - DatabaseBackend — Django ORM to store tasks
  - DummyBackend — Don't execute, store in memory

Background Workers

Backend Interface

- Task backends must inherit from `BaseTaskBackend`
- Common interface between Django and the task runner
- Functions to determine if a backend can run a task
  - e.g. if async is not supported
  - Throws InvalidTaskError exception
- Project can have multiple backends and queues

Background Workers

Task Interface

- Generic `Task` class
- Actions the task runners execute
- No need to subclass
- Tasks are immutable
  - Create a (reconfigured) copy of a task with "using"

Background Workers

Task Interface

```
from django.tasks import task

@task()
def calculate_meaning_of_life():
    return 42
```

- Can be function or coroutine
- Decorator parameters can overwrite defaults
- Task arguments must be JSON serializable

Background Workers

Tasks are normal functions

```
from django.tasks import task

@task()
def do_a_task(*args, **kwargs):
    pass

# Calls `do_a_task` as if it weren't a task
do_a_task()
```

Background Workers

Queueing Tasks

```python
from django.tasks import task

@task()
def add(a: int, b: int):
    return a + b

result = add.enqueue(31, b=11)
```

Background Workers

Queueing Tasks

```python
from django.tasks import task

@task(priority=100, backend='other-backend', queue_name='high-load')
def calculate_meaning_of_life():
    return 42
```

- Manage multiple queues
- Select different backend
- Set priorities

Background Workers

Overwrite Defaults

```
from django.tasks import task

@task()
def calculate_meaning_of_life():
    return 42

calculate_meaning_of_life.using(
    priority=100,
    backend='other-backend',
    queue_name='high-load'
).enqueue()
```

Background Workers

Deferring Tasks

```
from django.tasks import task

@task()
def do_a_task():
    pass

do_a_task.using(run_after=timedelta(minutes=5)).enqueue()
```

Background Workers

Task Results

- Queueing a task returns a `TaskResult` object
- Caches the task's result (once available)
- Stores status of task

NEW	The task has been created, but hasn't started running yet
RUNNING	The task is currently running
FAILED	The task failed
COMPLETE	The task is complete, and the result is accessible

Background Workers

Database Background Worker

- Start worker with: `python manage.py db_worker`
- Executes queued tasks
- Can be configured to only execute specific queues

Background Workers

Out of Scope (for now)

- Completion hooks
- Bulk queuing
- Automated task retrying
- Generic task runner execution
  - Custom backends need to implement their own runners
- Task Queue monitoring and reporting
- Cron-based scheduling
- Task timeouts

Background Workers

Disclaimer

- Work in Progress
- Not yet part of Django
  - Reference implementation django-tasks is an early demo
- Breaking changes likely
- A lot of missing features
  - See DEP for more infos

Demo

— Django Tasks in Action —

Django Tasks

Packages & Resources

- **Django Enhancement Proposal 14: Background Workers**
- **Django Tasks**
  - Reference Implementation by Jake Howard

Slides

andreas.earth/s/djmc-56