FLASK'S FLEXIBILITY FOR THE WIN

When Building Applications That Don't Follow The Normal Patterns
— Brett Alistair Kromkamp for **FlaskCon 2020**

WHO AM I?

BRIEF OVERVIEW

- **Brett** Alistair Kromkamp – @brettkromkamp (Twitter)
- Dutch, born in Africa (Zambia), living in Northern **Norway**
- Primarily a **software developer**, but also a CTO for 4 years and currently, CMO
- **Semantic technologies** enthusiast
- Worked in the **tourism industry** in Spain as a software developer for 17 years
- Currently, working in the **EdTech** sector – and have been, for the last 8 years

TALKING POINTS

01 THE TOPIC MAPS PARADIGM

02 SHOW, DON'T TELL: CONTEXTUALISE

03 WHY FLASK?

04 THE REPOSITORY PATTERN

05 WIRING UP AND USING THE DATA STORE

06 OTHER ESSENTIAL FLASK FEATURES

01

THE TOPIC MAPS PARADIGM

Topic maps provide a way to describe complex relationships between abstract concepts and the accompanying real-world (information) resources

THE TOPIC MAPS PARADIGM

DOMAIN MODEL — AN ASSOCIATIVE GRAPH

- **Topic**: represents an abstract concept
- **Association**: expresses a semantically meaningful relationship between two or more topics
- **Occurrence**: connects an information resource to a topic
- Scopes and scope filtering
- Metadata

02

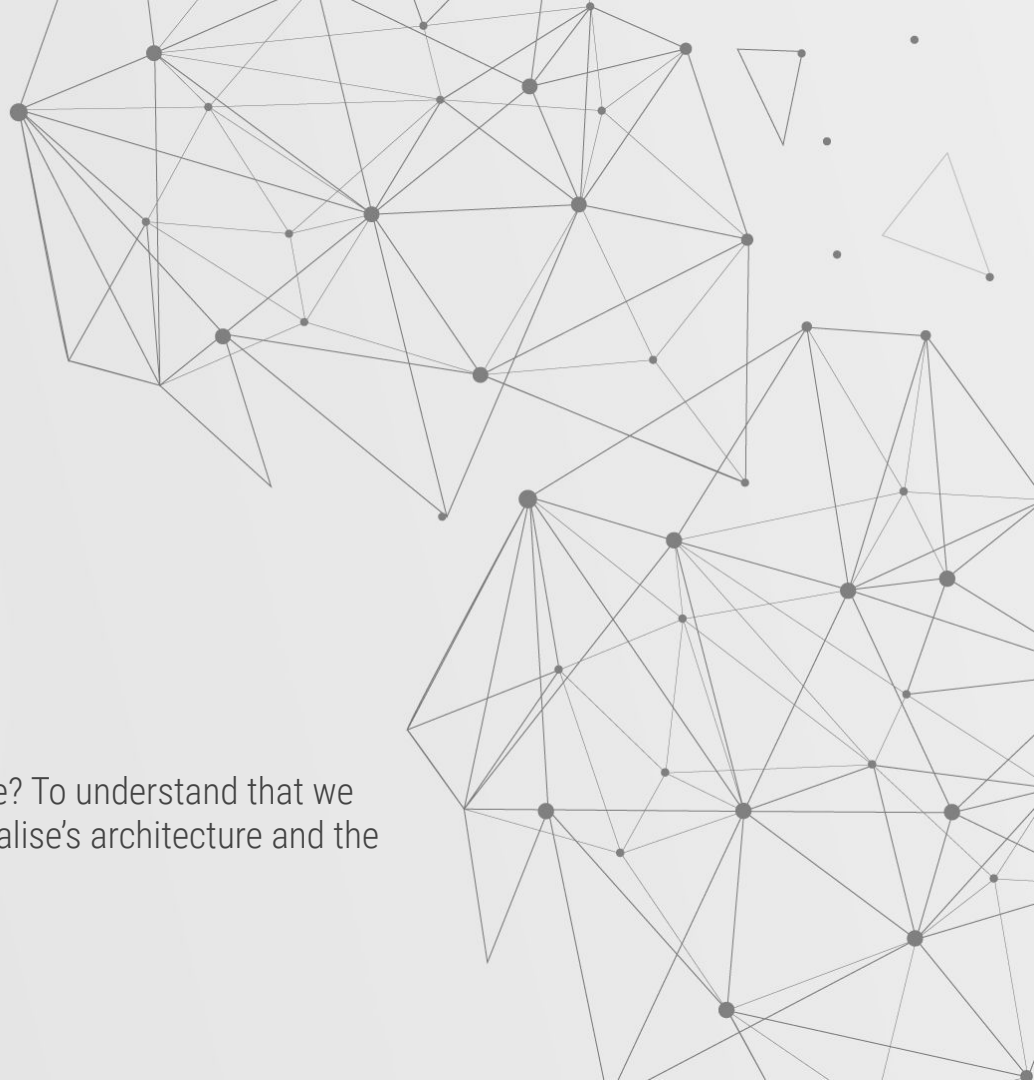
SHOW, DON'T TELL: CONTEXTUALISE

Contextualise is a simple and flexible open source tool particularly suited for organising information-heavy projects and activities consisting of unstructured and widely diverse data and information resources

SHOW, DON'T TELL: CONTEXTUALISE

TOPICS, ASSOCIATIONS AND OCCURRENCES

- Multiple topic maps
- Topics
- Associations
  - Navigable network graph
  - Associative tags
  - Knowledge paths — for easy hierarchical navigation through a topic map
- Occurrences and information resources
  - Images, Files, Links and Videos
  - glTF-based 3D scenes — with AR and VR support in September 2020



03

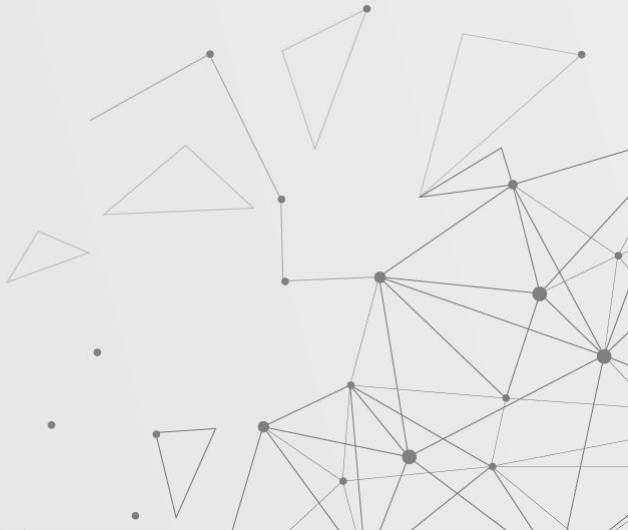
WHY FLASK?

Why did Flask make sense for Contextualise? To understand that we need to look at the intersection of Contextualise's architecture and the nature of Flask — hint: it's unopinionated

WHY FLASK?

CONTEXTUALISE'S ARCHITECTURE AND FLASK CHARACTERISTICS

- Flask
  - Small core
  - Extendable
  - Large and active community
  - Good documentation
  - **Unopinionated**
- Contextualise architecture
  - Broadly speaking, Contextualise is divided into a web "frontend" on one hand, and a **graph-based backend**, on the other
  - TopicDB, a so-called **topic maps engine** — a variation of the **repository pattern**



04

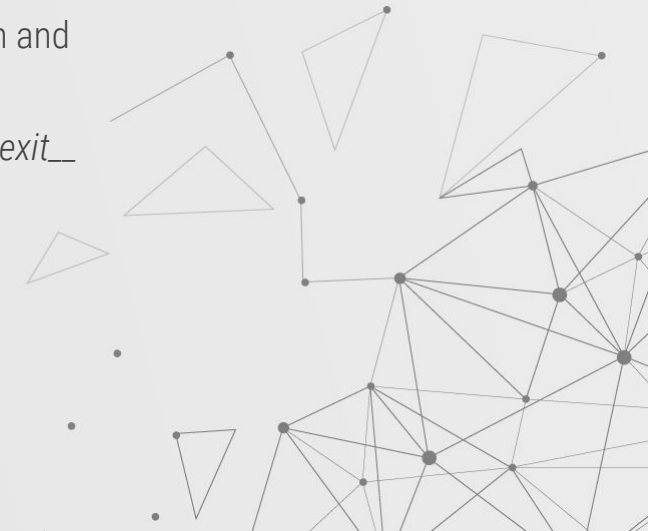
THE REPOSITORY PATTERN

Conceptually, a repository encapsulates the set of domain models persisted in a data store and the operations performed over them — *Martin Fowler*

THE REPOSITORY PATTERN

ONE OF THE SO-CALLED “ENTERPRISE” PATTERNS

- ➔ Mediates between the domain and data mapping layers
- ➔ Beneficial for a system with a complex domain model
- ➔ Achieves a clean separation and one-way dependency between the domain and data mapping layers
- ➔ In Python terms, the repository is a **context manager** with *__enter__* and *__exit__* methods for *open* and *close* (**connection**) semantics



05

WIRING UP AND USING THE DATA STORE

Use the source, Luke! Well, in this case… read the documentation:
Define and Access the Database

WIRING UP AND USING THE DATA STORE

TOPIC_STORE.PY

```
def get_topic_store():
    if "topicstore" not in g:
        g.topic_store = TopicStore(
            current_app.config["TOPIC_STORE_USER"],
            current_app.config["TOPIC_STORE_PASSWORD"],
            host=current_app.config["TOPIC_STORE_HOST"],
            port=current_app.config["TOPIC_STORE_PORT"],
            dbname=current_app.config["TOPIC_STORE_DBNAME"]
        )
        g.topic_store.open()
    return g.topic_store

def close_topic_store(e=None):
    topic_store = g.pop("topicstore", None)
    if topic_store is not None:
        topic_store.close()
```

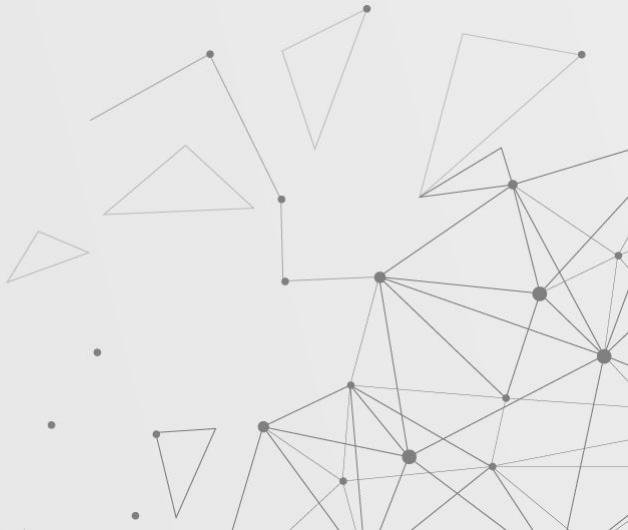
WIRING UP AND USING THE DATA STORE

TOPIC_STORE.PY (CONTINUED)

```
def init_app(app):
    app.teardown_appcontext(close_topic_store)
```

__INIT__.PY

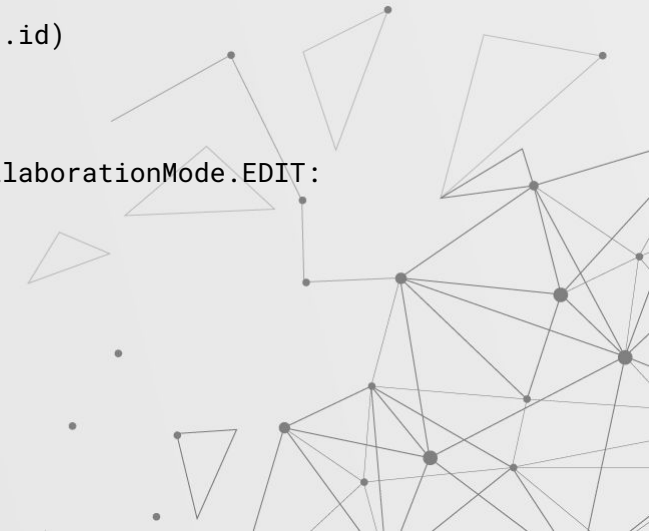
```
from contextualise import topic_store

topic_store.init_app(app)
```



WIRING UP AND USING THE DATA STORE

VIDEO.PY (BLUEPRINT)

```
@bp.route("/videos/<map_identifier>/<topic_identifier>")
@login_required
def index(map_identifier, topic_identifier):
    topic_store = get_topic_store()
    topic_map = topic_store.get_topic_map(map_identifier, current_user.id)
    if topic_map is None:
        abort(404)
    if not topic_map.owner and topic_map.collaboration_mode is not CollaborationMode.EDIT:
        abort(403)
    topic = topic_store.get_topic(map_identifier, topic_identifier)
    if topic is None:
        abort(404)
```



06

OTHER ESSENTIAL FLASK FEATURES

Blueprints, filters and extensions

OTHER ESSENTIAL FLASK FEATURES

FLASK FEATURES THAT MADE MY LIFE EASY

- Blueprints
- Filters
- Extensions
  - Flask-Security-Too
  - Flask-WTF
  - Flask-SeaSurf — extension for preventing cross-site request forgery, CSRF

RESOURCES

LINKS

- ➔ Contextualise application: https://contextualise.dev/
- ➔ Contextualise on GitHub: https://github.com/brettkromkamp/contextualise
- ➔ TopicDB topic maps engine on GitHub: https://github.com/brettkromkamp/topic-db
- ➔ The TAO of Topic Maps: https://ontopia.net/topicmaps/materials/tao.html

THANKS

Does anyone have any questions?

info@contextualise.dev
https://contextualise.dev