BEL Commons Documentation

Release 0.3.2-dev

Charles Tapley Hoyt

May 15, 2020

CONTENTS

CHAPTER

ONE

INSTALLATION

This application runs on Python 3.7+.

1.1 Database

For production, it is preferred to use a multi-threading relational database management system. PyBEL has been best tested on PostgreSQL, so this is preferred for now.

1.2 Message Broker

This application uses Celery as a task management system to support asynchronous parsing of BEL documents, running of analyses, and other slow operations.

RabbitMQ, or any other message queue supported by Celery are appropriate.

1.3 Server

Because this application is built with Flask, it can be run with the WSGI protocol. Running on a single machine is possible with either the built-in `werkzeug` test server or something easy to install like `gunicorn`.

For production, `uwsgi` seems to work pretty well.

CHAPTER

TWO

RUNNING WITH DOCKER

Docker is very powerful as a general way to specify how things should be installed, but has a steep learning curve. After installing and running `docker-machine` and `docker-compose`, BEL Commons can be run with a few simple commands.

2.1 Dockerfile

A simple Dockerfile is included at the root-level of the repository. This Dockerfile is inspired by the tutorials from Container Tutorials and Digital Ocean.

Warning:

- The virtual machine needs at least 2GB memory for the worker container
- The database needs a packet size big enough to accommodate large BEL files (>10 mb)

CHAPTER

THREE

PAGES

This module contains the user interface blueprint for the application.

CHAPTER

FOUR

CONFIGURATION

Default configuration can be found in the module `bel_commons.config`.

By default, PyBEL searches for a configuration file called `config.json` in `~/.config/pybel/`. This directory can be modified with the environment variable `PYBEL_CONFIG_DIRECTORY`. Additioanlly, the location of another custom configuration can be specified by the environment variable `BEL_COMMONS_CONFIG_JSON`.

In `config.json` add an entry `PYBEL_MERGE_SERVER_PREFIX` for the address of the server. Example: `http://lisa:5000` with no trailing backslash. This is necessary since celery has a problem with flask's url builder function `flask.url_for`.

Add an entry `PYBEL_CONNECTION` with the database connection string to either a local SQLite database or a proper relational database management system. It's suggested to `pip install psycopg2-binary` in combination with MySQL since it enables multi-threading.

For a deployment with a local instance of RabbitMQ, the default configuration already contains a setting for `amqp://localhost`. Otherwise, an entry `CELERY_BROKER_URL` can be set.

class bel_commons.config.**BELCommonsConfig**(*SECRET_KEY: str*, *BUTLER_PASSWORD: str*, *BUTLER_EMAIL: str = 'butler'*, *BUTLER_NAME: str = 'BEL Commons Butler'*, *DEBUG: bool = False*, *TESTING: bool = False*, *JSONIFY_PRETTYPRINT_REGULAR: bool = True*, *SQLALCHEMY_TRACK_MODIFICATIONS: bool = False*, *SQLALCHEMY_DATABASE_URI: str = <factory>*, *DISALLOW_PRIVATE: bool = True*, *USE_CELERY: bool = True*, *CELERY_BROKER_URL: str = 'amqp://localhost'*, *CELERY_BACKEND_URL: str = 'redis://localhost'*, *SECURITY_REGISTERABLE: bool = True*, *SECURITY_CONFIRMABLE: bool = False*, *SECURITY_SEND_REGISTER_EMAIL: bool = False*, *SECURITY_RECOVERABLE: bool = False*, *SECURITY_PASSWORD_HASH: str = 'pbkdf2_sha512'*, *SECURITY_PASSWORD_SALT: str = 'default_please_override'*, *MAIL_SERVER: Optional[str] = None*, *MAIL_DEFAULT_SENDER_NAME: str = 'BEL Commons'*, *MAIL_DEFAULT_SENDER_EMAIL: Optional[str] = None*, *REGISTER_EXAMPLES: bool = False*, *REGISTER_USERS: Optional[str] = None*, *REGISTER_ADMIN: bool = True*, *REGISTER_TRANSFORMATIONS: bool = True*, *WRITE_REPORTS: bool = False*, *ENABLE_UPLOADER: bool = False*, *ENABLE_PARSER: bool = False*, *ENABLE_ANALYSIS: bool = False*, *ENABLE_CURATION: bool = False*, *SWAGGER_CONFIG: Mapping[str, Any] = <factory>*)

Configuration for BEL Commons.

It assumes you have:

- SQLite for the PyBEL Cache on localhost
- RabbitMQ or another message broker supporting the AMQP protocol running on localhost

SECRET_KEY: str = None
The Flask app secret key.

BUTLER_PASSWORD: str = None
Password for the butler account

DEBUG: bool = False
Flask app debug mode

TESTING: bool = False
Flask app testing mode

JSONIFY_PRETTYPRINT_REGULAR: bool = True

Database and SQLAlchemy settings

DISALLOW_PRIVATE: bool = True
Should private network uploads be allowed?

USE_CELERY: bool = True
Should celery be used?

CELERY_BACKEND_URL: str = 'redis://localhost'
Celery backend url

SECURITY_PASSWORD_HASH: str = 'pbkdf2_sha512'
What hash algorithm should we use for passwords

SECURITY_PASSWORD_SALT: str = 'default_please_override'
What salt should to use to hash passwords

REGISTER_EXAMPLES: bool = False
Should example graphs be automatically included?

REGISTER_USERS: Optional[str] = None
Path to user manifest file

REGISTER_ADMIN: bool = True
Register the Flask-Admin interface

WRITE_REPORTS: bool = False
Which parts of BEL Commons should run?

classmethod load_dict() → Mapping[str, Any]
Get configuration as a dictionary.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX