quorum documentation

Release 0.2.1

Hive Solutions Lda.

April 24, 2014

Contents

A small extension framework for Flask to easy a series of simple tasks.

CHAPTER 1

Introduction

1.1 Goals and Focus

The focus of this project was...

CHAPTER 2

Changelog

List the complete set of changes to the quorum project since it's creation.

2.1 Current Versions

2.1.1 0.2.0

- Initial support for `Python 3.0+`
- More stability in the infra-structure

2.1.2 0.1.8

- New support for `quorum.exists_amazon_key()` and `quorum.clear_amazon_bucket()` calls
- Better unit testing for `amazon.py`
- Support for the SERVER_* environment variables

2.1.3 0.1.7

- Better signature for `quorum.send_mail()`
- Improved asynchronous mode under `quorum.send_mail_a()`
- New support for `quorum.delete_amazon_key()` calls

2.1.4 0.1.6

- Support for Amazon S3 using boto
- Experimental documentation

2.2 Older Versions

2.2.1 0.1.5

- Initial support for `mongodb`

2.2.2 0.1.4

- Legacy support values

2.2.3 0.1.3

- Legacy support values

2.2.4 0.1.1

- Legacy support values

2.2.5 0.1.0

- Initial release
- First specification of the framework

CHAPTER 3

Configuration Variables

3.1 Global

`.. NAME:: string (default = None)`
String that is going to be used to identify the app for the internal structures a fallback value should be defined by the developer of the app.

`.. INSTANCE:: string (default = None)`
The descriptive value of the instance that is going to be executed. This value should provide the required mechanisms to correctly separate to virtually different running instance (eg: databases, log files, etc.).

`.. LEVEL:: string (default = ”INFO”)`
Defines the verbosity level that is going to be used while running the application’s logger. It also provides a way of defining if the application is running for production where production is *INFO* or more verbose levels.

`.. DEBUG:: boolean (default = False)`
If the application should be ran under the run mode.

Note: This variable is considered to be deprecated and the `LEVEL` variable should be used instead with the DEBUG level set.

`.. PORT:: integer (default = 5000)`
The port to be used by the default http server when binding to the socket.

Note: The `PORT` value is not used when running in contained **WSGI**.

`.. HOST:: string (default = ”127.0.0.1”)`
The port to be used by the default http server when binding to the socket. This value may be *0.0.0.0* in case the user want to bind to every interface on the system (may be insecure).

Note: The `HOST` value is not used when running in contained **WSGI**.

3.2 Mail / SMTP

`.. SMTP_HOST:: string`
Hostname or IP address of the server to be used as gateway for sending smtp messages (e-mails) under quorum.

This value may contain an optional port value separated by a **‘:’** character.

.. **SMTP_USER::** `string`
Username to be used in the authentication process on the SMTP connections used for sending email messages.

Note: Most of the times the username is an email address and as such it's also used as the default fallback value for the sender value for outgoing emails.

.. **SMTP_PASSWORD::** `string`
Password to be used in the authentication process on the SMTP connections used for sending email messages.

3.3 MongoDB

.. **MONGOHQ_URL::** `string`
The url to be used for the establishment of connection to the MongoDB server. It must contain authentication information, host, port and optionally the default database to be used.

Note: An example url for mongo would be something like **mongodb://root:root@db.hive:27017**.

3.4 Redis

.. **REDISTOGO_URL::** `string`
The URL that described the connection to be used with the REDIS key value database, this URL is going to be used under the redis-py infra-structure.

Note: An example url for rabbit would be something like **redis://root:root@db.hive**.

3.5 RabbitMQ / AMQP

.. **CLOUDAMQP_URL::** `string`
URL used by the RabbitMQ library (pika) to create the connection with the server that is going to be used in the session. It should contain both authentication and location information.

Note: An example url for rabbit would be something like **amqp://root:root@db.hive**.

3.6 Amazon Web Services

.. **AMAZON_ID::** `string`
Identifier of the Amazon S3 account to be sued for the connection, this should comply with the expected string values.

.. **AMAZON_SECRET::** `string`
The secret value of the account to be used, this value should be kept secret from any external person to avoid security problems.

.. **AMAZON_BUCKET::** `string`
The name of the bucket where file of the current application are going to be stored. Currently there's no support for multiple buckets per one application scope.

3.7 Pusher

.. **PUSHER_APP_ID**:: string
    TODO

.. **PUSHER_KEY**:: string
    TODO

.. **PUSHER_SECRET**:: string
    TODO

CHAPTER 4

API Reference

4.1 quorum

4.1.1 Base

`quorum.`**`APP`** **= None**

The reference to the top level application object that is being handled by quorum. This value is used across the quorum infra-structure to access flask data and capabilities.

Note: Changing this value directly should be done with care as it may create undesired results. To set/start this value use the `quorum.load()` function instead.

`quorum.`**`load`**(*app=None*, *name=None*, *locales=('en_us'*, *)*, *secret_key=None*, *execution=True*, *redis_session=False*, *mongo_database=None*, *logger=None*, *models=None*, *safe=False*, ***kwargs*)

Initial loader function responsible for the overriding of the flask loading system and for the loading of configuration.

Note: This function should be called inside you main app file failure to do so may result in unexpected behavior.

Parameters

- **app** (*Application*) – The optional flask application object to be used in the loading of quorum (useful for self managed apps).
- **name** (*String*) – The name to be used to describe the application for the management of internal values.
- **locales** (*List*) – The list containing the various locale strings for the locales available for the application to be loaded.
- **secret_key** (*String*) – The secret seed value to be used for cryptographic operations under flask (eg: client side sessions) this value should be shared among all the pre-fork instances.
- **execution** (*bool*) – Flag indicating if the (background) execution thread should be started providing the required support for background tasks.
- **redis_session** (*bool*) – If the session management for the flask infra-structure should be managed using a server side session with support from redis.
- **mongo_database** (*String*) – The default name of the database to be used when using the mongo infra-structure.

- **logger** (*String*) – The name to be used as identification for possible logger files created by the logging sub-module.
- **models** (*Module*) – The module containing the complete set of model classes to be used by the data infra-structure (eg: `mongo`).
- **safe** (*bool*) – If the application should be run in a safe mode meaning that extra validations will be done to ensure proper execution, typically these kind of validations have a performance impacts (not recommended).

Return type Application

Returns The application that is used by the loaded quorum environment in case one was provided that is retrieved, otherwise the newly created one is returned instead.

`quorum.`**`unload`**`()`

Unloads the current quorum instance, after this call no more access to the quorum facilities is allowed.

A normal setup of the application would never require this function to be called directly.

Warning: Use this function with care as it may result in unexpected behavior from a developer point of view.

Note: After the call to this method most of the functionally of quorum will become unavailable until further call to `quorum.load()`.

`quorum.`**`run_back`**(*callable*, *args=*[ ], *kwargs={}*, *target_time=None*, *callback=None*)

Runs the provided callable (function, method, etc) in a separated thread context under submission of a queue system. It's possible to control the runtime for the execution with the `target_time` argument and it's also possible to be notified of the end of the execution providing a callable to the `callback` parameter.

Warning: The execution is not guaranteed as the system process may be interrupted and resuming of the execution would not be possible.

Parameters

- **callable** (*Function*) – The callable object to be called in a separated execution environment.
- **args** (*Dictionary*) – The list of unnamed argument values to be send to the callable upon execution.
- **args** – The dictionary of named argument values to be send to the callable upon execution.
- **target_time** (*float*) – The target timestamp value for execution, in case it's not provided the current time is used as the target one.
- **callback** (*Function*) – The callback function to be called upon finishing the execution of the callable, in case an error (exception) on executing the callback the error is passed as error argument.

`quorum.`**`run_background`**(*callable*, *args=*[ ], *kwargs={}*, *target_time=None*, *callback=None*)

Runs the provided callable (function, method, etc) in a separated thread context under submission of a queue system. It's possible to control the runtime for the execution with the `target_time` argument and it's also possible to be notified of the end of the execution providing a callable to the `callback` parameter.

Warning: The execution is not guaranteed as the system process may be interrupted and resuming of the execution would not be possible.

Parameters

- **callable** (*Function*) – The callable object to be called in a separated execution environment.
- **args** (*Dictionary*) – The list of unnamed argument values to be send to the callable upon execution.
- **args** – The dictionary of named argument values to be send to the callable upon execution.
- **target_time** (*float*) – The target timestamp value for execution, in case it's not provided the current time is used as the target one.
- **callback** (*Function*) – The callback function to be called upon finishing the execution of the callable, in case an error (exception) on executing the callback the error is passed as error argument.

4.1.2 Mail / SMTP

`quorum.`**`send_mail`**(*app=None*, *subject=''*, *sender=None*, *receivers=[ ]*, *data=None*, *plain=None*, *rich=None*, *context={}*)

Sends an email message using the provided `SMTP_HOST`, `SMTP_USER` and `SMTP_PASSWORD` configurations.

The email message is sent under the `alternative` mime type so that both the plain text and the rich text (html) parts are sent in the same message.

The `plain` and `rich` arguments allow the user to process a template with the context provided by the `context` map.

Warning: This is a blocking call and as such the control flow may block for more that a second, if you want a non blocking (asynchronous) call please use `quorum.send_mail_a()`.

Parameters

- **app** (*Application*) – Optional application object to be used for the rendering operation as the main object for flask. In case this value is not provided the global `quorum.APP` value is used instead (fallback).
- **subject** (*String*) – The mime subject to be sent with this message, note that if this value is not set many spam filters will consider the message as spam.
- **sender** (*String*) – The email (and name) of the sender of the email, in case this value is not set the `SMTP_USER` variable is used instead.
- **receivers** (*List*) – The list of receivers (with email and name) for which the email will be sent.
- **data** (*String*) – The buffer containing the data to be used for both the plain and the rich text parts in case the template associated arguments are not set or in case the rendering is not successful.
- **plain** (*String*) – Relative path to the plain text template to be used for the rendering of the email, this path must be relative to the templates folder.
- **rich** (*String*) – Relative path to the rich text template to be used for the rendering of the email, this path must be relative to the templates folder.
- **context** (*Dictionary*) – The map containing the complete set of variables that are going to be *"exposed"* to the template rendering engine for both the `plain` and the `rich`.

`quorum.`**`send_mail_a`**(**args*, ***kwargs*)

Asynchronous call to the `quorum.send_mail()` function that is executed in a different thread from the current one. The currently loaded queue system is used for the sending of the email, for more information check on `quorum.run_background()` .

Note: The arguments to be send for this function are the same as the one present in the original `quorum.send_mail()` function.

4.1.3 MongoDB

`enumerate`(*sequence*[, *start=0*])

Return an iterator that yields tuples of an index and an item of the

4.2 quorum.mongodb

`quorum.mongodb.`**`get_connection`**`()`

CHAPTER 5

Glosary

BIOS A set of computer instructions in firmware that control input and output operations.

EPEL Common acronym for Extra Packages for Enterprise Linux. These are typically included in Fedora Linux, and provided for RedHat, CentOS, and other RPM-based distributions. The project's homepage is here: FedoraProject:EPEL.

MAC Pronounced as "mack" and often used as a noun referring to a network device's Media Access Controller (MAC) address. A MAC address is a globally unique number assigned to each interface in an Ethernet network and used to direct Ethernet frames between source and destination devices.

OSI Open Systems Interconnection (OSI) is an effort to develop standards-based computer networking protocols in order to allow networking equipment from different vendors to interoperate without relying on implementation of competing proprietary protocols. The OSI is best known for the development of the standard seven-layer OSI model for describing layers of abstraction into which the various networking protocols are categorized.

POSIX An acronym for "Portable Operating System Interface", is a family of standards specified by the IEEE for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems. Read more here: Wikipedia:POSIX.

RFC The RFC documents (Request for Comments) are a series of Internet standards, best common practices, and related documents describing how networked computers communicate. This document series provides the standards for how the Internet and many other related technologies interrelate and interoperate.

CHAPTER 6

Installation

Install using setuptools, e.g. (within a virtualenv):

```
$ easy_install quorum
```

Example

```python
import flask
import quorum

app = quorum.load(
    name = __name__
)

@app.route("/", methods = ("GET",))
def index():
    return flask.render_template("index.html.tpl")

if __name__ == "__main__":
    quorum.run()
```

CHAPTER 8

Indices and tables

- *genindex*
- *modindex*
- *search*