Chassis.py

Oct 15, 2019

Contents

Chassis.py is a framework/library that helps developing services by providing a solution for a some cross-cutting concerns.

CHAPTER 1

Usage

1.1 Configuration

Using the `configuration` module the chassis can be configured, but it can also be used for application-specific configuration.

When you make a new instance of the `ConfigParser` it will contain a property `config` that contains all the configuration.

The dictionary is filled based on following priority:

1. command line arguments (*not implemented*)
2. environment variables (*not implemented*)
3. config.yml file in the project root folder
4. default values

```
from viaa.configuration import ConfigParser

config = ConfigParser()
```

Example of a `config.yml` file.

```
viaa:
  logging:
    level: 40
application:
  throttle_time: 10
```

1.2 Logging

The logging interface is made to resemble the Python standard logging as close as possible. The biggest difference is that you can pass a config object when getting a logger instance.

ALL logging will go to *stdout* as a JSON string, this means that logging by external packages will be formatted aswell. Formatting all logs as JSON makes it easier to parse the logs in other applications.

Running following example:

```
import logging as basic_logging
from viaa.observability import logging
from viaa.configuration import ConfigParser

example_dictionary = {"test_key": "test_value"}

config = ConfigParser()
logger = logging.get_logger(__name__, config)

basic_logger = basic_logging.getLogger()

logger.warning("Hello world!", string="extra info", dictionary=example_dictionary)
basic_logger.warning("basic log")
```

Will print:

```
{"message": "Hello world!", "string": "extra info", "dictionary": {"test_key": "test_
↪value"}, "logger": "__main__", "level": "warning", "timestamp": "2019-10-
↪03T12:52:21.857624Z", "source": ".\\hello.py:<module>:12"}
{"message": "basic log"}
```

1.2.1 API

`viaa.observability.logging.`**`get_logger`**(*name=''*, *config: viaa.configuration.ConfigParser = None*)

Return a logger with the specified name and configuration, creating it if necessary. If no name is specified, return the root logger. If a config is specified it will override the current config for a logger.

CHAPTER 2

Code examples

2.1 Logging

2.2 Tracing

2.3 API Wrappers

2.3.1 Mediahaven API

Python Module Index

V

Index

G

V