Index

msaUtils - General utils for Microservices based on FastAPI like Profiler, Scheduler, Sysinfo, Healtcheck, Error Handling etc.

- **MSASendEmail**: Helper class to send email via smtp server.
- **MSABaseExceptionHandler**: Central exception handler which sends formatted exception to logger
- **Filehandler utilities**: Classes for FileDelete, FileUpload async with chunking, Archive pack/unpack formats, helper functions
- **Find system fonts**: Find available system installed fonts or search in specific directory
- **mkdocs code reference helper**: Create virtual mkdocs navs for code reference and used libraries from requirements.txt
- **MSAHealthCheck**: Healthcheck class with internal own thread, which checks url for health
- **HTML Sanitizer**: Helper function to sanitize 'dirty html' for text processing
- **logger intercept handler**: allows to change handler from all logger and define specific output format with loguru
- **Models for files and health classes**: reusable pydantic models for file handling and dealing with healthcheck status
- **MSAProfilerMiddleware**: PyInstrument Profiler as Middleware to create a html for an admin Dashboard

- **MSAScheduler**: Scheduler class to define scheduled tasks in natural language `time of day between 10:00 and 16:00`
- **MSAAppSettings**: API oriented settings class with environment vars and `.env` file support
- **Service oriented System Info**: Classes and functions to get pydantic model response about system and gpu information
- **MSAToken**: API token handler classes and functions based on oauth2, jwt etc.

Usage example

```
from fastapi import FastAPI

from msaUtils.profiler import MSAProfilerMiddleware
from msaUtils.scheduler import MSAScheduler
from msaUtils.sysinfo import MSASystemInfo, get_sysinfo

app = FastAPI()

...

# sysinfo
sysinfo: MSASystemInfo = get_sysinfo()
# return app.templates.TemplateResponse(
#     "monitor.html", {"request": request, "outputSystemInfo": sysinfo}
# )

# profiler middleware
app.add_middleware(
    MSAProfilerMiddleware
)


# scheduler
async def test_timer_min():
    print("msaSDK Test Timer Async Every Minute")


def test_timer_five_sec():
    print("msaSDK Test Timer Sync 5 Second")


myScheduler = MSAScheduler()
myScheduler.task("every 1 min", func=test_timer_min)
myScheduler.task("every 5 sec", func=test_timer_five_sec)
```

License Agreement

- `msaUtils` Based on `MIT` open source and free to use, it is free for commercial use, but please show/list the copyright information about msaUtils somewhere.

How to create the documentation

We use mkdocs and mkdocsstring. The code reference and nav entry get's created virtually by the triggered python script /docs/gen_ref_pages.py while `mkdocs` `serve` or `build` is executed.

Requirements Install for the PDF creation option:

PDF Export is using mainly weasyprint, if you get some errors here pls. check there documentation. Installation is part of the msaUtils, so this should be fine.

We can now test and view our documentation using:

```
mkdocs serve
```

Build static Site:

```
mkdocs build
```

Build and Publish

Build:

```
python setup.py sdist
```

Publish to pypi:

```
twine upload dist/*
```