MSAApp

Example Usage of MSAApp

MSAApp Service Definition Settings

```
# MSAServiceDefinition

name: str = "msaSDK Service"
"""Service Name, also used as Title."""
version: str = "0.0.0"
"""Version of the Service."""
host: str = "127.0.0.1"
"""Host/IP which the service runs on."""
port: int = 8090
"""Port which the service binds to."""
tags: List[str] = []
"""Optional Metadata: Use this to carry some variables through the service
instance."""
allow_origins: List[str] = ["*"]
"""CORSMiddleware. List[str]. List of allowed origins (as strings) or all of them
with the wildcard ``*`` ."""
allow_credentials: bool = True
"""CORSMiddleware. Bool. Allow (True) Credentials (Authorization headers, Cookies,
etc)."""
allow_methods: List[str] = ["*"]
"""CORSMiddleware. List[str]. Specific HTTP methods (POST, PUT) or all of them with
the wildcard ``*`` ."""
allow_headers: List[str] = ["*"]
"""CORSMiddleware. List[str]. Specific HTTP headers or all of them with the wildcard
``*`` ."""
healthdefinition: MSAHealthDefinition = MSAHealthDefinition()
"""Healthdefinition Instance."""
uvloop: bool = True
"""Use UVLoop instead of asyncio loop."""
sysrouter: bool = True
"""Enable the System Routes defined by router.system module (/sysinfo, /sysgpuinfo,
/syserror, ...)."""
servicerouter: bool = True
"""Enable the Service Routes defined by the MSAApp (/scheduler, /status, /defintion,
/settings, /schema, /info, ...)."""
starception: bool = True
"""Enable Starception Middleware."""
validationception: bool = True
"""Enable Validation Exception Handler."""
httpception: bool = True
"""Enable the HTTP Exception Handler, which provides HTML Error Pages instead of
JSONResponse."""
httpception_exclude: List[int] = [307, ]
"""List of HTTP Exception Codes which are excluded and just forwarded by the HTTP
Exception Handler."""
cors: bool = True
"""Enable CORS Middleware."""
httpsredirect: bool = False
"""Enable HTTPS Redirect Middleware."""
gzip: bool = False
"""Enable GZIP Middleware."""
session: bool = False
"""Enable Session Middleware."""
csrf: bool = True
"""Enable CSRF Forms Protection Middleware."""
msgpack: bool = False
"""Enable Messagepack Negotiation Middleware."""
instrument: bool = True
```

```
    """Enable Prometheus Instrumentation for the instance."""
    static: bool = True
    """Enable the internal Static Folder (``msastatic``) and Mount to instance."""
    pages: bool = True
    """Enable the Pages Routes (/profiler, /testpage), if site is Off also (/, /monitor,
    /monitor_inline)."""
    graphql: bool = False
    """Enable initiation of Strawberry GraphQLRouter (/graphql)."""
    context: bool = False
    """Enable Context Middleware."""
    pagination: bool = False
    """Enable FastAPI Pagination."""
    profiler: bool = False
    """Enable Profiler Middleware."""
    profiler_output_type: str = "html"  # text or html
    """Set the Profiler Output Type, should be html or text, html is needed if you want
    to use the profiler on the Admin Site."""
    profiler_single_calls: bool = True
    """Enable to Track each Request by the Profiler."""
    profiler_url: str = "/profiler"
    """Set the URL to reach the profiler result html, /profiler."""
    templates: bool = True
    """Enable the internal Templates and mount the directory."""
    templates_dir: List[str] = ["msatemplates", "msatemplates/errors"]
    """Set the List of Directories for the MSAUITemplate Engine to look for the
    requested template."""
    timing: bool = False
    """Enables Timing Middleware, reports timing data at the granularity of individual
    endpoint calls."""
    limiter: bool = False
    """Enables Rate Limiter (slowapi)."""
    scheduler: bool = True
    "Enables MSA Scheduler Engine."
    scheduler_debug: bool = False
    "Enables MSA Scheduler debug messages."
    abstract_fs: bool = True
    """Enables internal Abstract Filesystem."""
    abstract_fs_url: str = "."
    """Set's Filesystem URL"""
    json_db: bool = True
    """Enables internal NoSQl/TinyDB DB."""
    json_db_memory_only: bool = False
    """JSON DB only in memory, don't store to file/db url"""
    json_db_url: str = "./msa_sdk.json"
    """Set's DB URL, compatibility with async and SQLModel/SQLAlchemy is required."""
    sqlite_db: bool = True
    """Enables internal Asynchron SQLite DB."""
    sqlite_db_debug: bool = False
    """Enables internal DB Debug output."""
    sqlite_db_crud: bool = True
    """Enables CRUD API creation of the provided SQLModels."""
    sqlite_db_meta_drop: bool = False
    """If True, all existing Data and Schemas in internal DB get's deleted at
    Startup."""
    sqlite_db_meta_create: bool = True
    """Enables internal DB Metadata creation from defined SQLModels at Startup."""
    sqlite_db_url: str = "sqlite+aiosqlite:///msa_sdk.sqlite_db?check_same_thread=True"
    """Set's DB URL, compatibility with async and SQLModel/SQLAlchemy is required."""
    ui_justpy: bool = True
    """Enables internal justpy mounting."""
    ui_justpy_demos: bool = True
    """Enables justpy demos"""
```

```
    site: bool = True
    """Enables internal Admin Site Dashboard."""
    site_auth: bool = False
    """Extends internal Admin Dashboard with Auth."""
    site_title: str = "Admin"
    """Set's internal Admin Dashboard Titel."""
    site_copyright: str = "Copyright © 2022 by u2d.ai"
    """Set's internal Admin Dashboard copyright information."""
    site_icon: str = "/msastatic/img/favicon.png"
    """Set's internal Admin Dashboard Favicon."""
    site_url: str = ""
    """Set's internal Admin Dashboard Site URL, normally empty."""
    root_path: str = "/admin"
    """Set's internal Admin Dashboard Root Path, normally ``/admin``."""
    language: str = ""  # 'zh_CN','en_US'
    """Set's internal Admin Dashboard language (``zh_CN`` or ``en_US``=default if
    empty)."""
```

Minimal Example

```
from msaSDK.models.service import get_msa_app_settings
from msaSDK.service import MSAApp


settings = get_msa_app_settings()
settings.title = "Your Microservice Titel"
settings.version = "0.0.1"
settings.debug = True

app = MSAApp(settings=settings)

app.logger.info("Initialized " + settings.title + " " + settings.version)


@app.get("/my_service_url")
def my_service():
    return {"message": "hello world"}

if __name__ == '__main__':
    pass
```

With CRUD/SQLModels and Scheduler

```
from typing import Optional, List

from sqlmodel import SQLModel

from msaSDK.admin.utils.fields import Field
from msaSDK.models.service import get_msa_app_settings
from msaSDK.service import MSAApp
```

```
async def test_timer_min():
    app.logger.info("msaSDK Test Timer Async Every Minute")


def test_timer_five_sec():
    app.logger.info("msaSDK Test Timer Sync 5 Second")


class TestArticle(SQLModel, table=True):
    __table_args__ = {'extend_existing': True}
    id: Optional[int] = Field(default=None, primary_key=True, nullable=False)
    title: str = Field(title='ArticleTitle', max_length=200)
    description: Optional[str] = Field(default='', title='ArticleDescription',
max_length=400)
    status: bool = Field(None, title='status')
    content: str = Field(title='ArticleContent')


class TestCategory(SQLModel, table=True):
    __table_args__ = {'extend_existing': True}
    id: Optional[int] = Field(default=None, primary_key=True, nullable=False)
    title: str = Field(title='ArticleTitle', max_length=200)
    description: Optional[str] = Field(default='', title='ArticleDescription',
max_length=400)
    status: bool = Field(None, title='status')
    content: str = Field(title='ArticleContent')


get_msa_app_settings.cache_clear()
settings = get_msa_app_settings()
settings.title = "u2d.ai - MSA/SDK MVP"
settings.version = "0.0.1"
settings.debug = True

app = MSAApp(settings=settings, auto_mount_site=True,
             sql_models=[TestArticle, TestCategory],
             contact={"name": "msaSDK", "url": "http://u2d.ai", "email":
"stefan@u2d.ai"},
             license_info={"name": "MIT", "url": "https://opensource.org/licenses/MIT",
})

app.scheduler.task("every 1 min", func=test_timer_min )
app.scheduler.task("every 5 sec", func=test_timer_five_sec )

app.logger.info("Initialized " + settings.title + " " + settings.version)


@app.on_event("startup")
async def startup():
    app.logger.info("msaSDK Own Startup MSAUIEvent")
    #app.mount_site()


@app.on_event("shutdown")
async def shutdown():
    app.logger.info("msaSDK Own Shutdown MSAUIEvent")
```

```
if __name__ == '__main__':
    pass
```

System and Service API routes (Optional)

Turning on the MSAApp settings `sysrouter` and `servicerouter` then you also get the following routes:

Last update: September 17, 2022

Created: August 24, 2022