fastapi

- python restful API服务器fastAPI
- fastapi 路由管理
- fastapi项目结构
- 参数传递
- 使用中间件（拦截器）
- 链接数据库

python restful API服务器fastAPI

组件

- fastAPI : https://fastapi.tiangolo.com/
- uvicorn: https://www.uvicorn.org/
- https://iovhm.com/book/attachments/16

快速开始

```
# 安装fastAPI
pip install fastapi -i https://pypi.tuna.tsinghua.edu.cn/simple

# 安装web容器
pip install "uvicorn[standard]" -i https://pypi.tuna.tsinghua.edu.cn/simple

# 安装from参数表单
pip install python-multipart -i https://pypi.tuna.tsinghua.edu.cn/simple
```

编写代码

```
# main.py

from fastapi import FastAPI, APIRouter
from fastapi.staticfiles import StaticFiles

app = FastAPI()
router = APIRouter()

# 静态目录
app.mount("/static", StaticFiles(directory="./static"), name="static")
```

```
@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/")
async def read_root():
    html_content = """
    <html>
        <head>
            <title>FastAPI</title>
        </head>
        <body>
            <h1>Hello, FastAPI!</h1>
            <p>
                <a href="https://fastapi.tiangolo.com">
                    https://fastapi.tiangolo.com
                </a>
            </p>
        </body>
    </html>
    """
    return HTMLResponse(content=html_content, media_type="text/html")



# 运行
uvicorn main:app --reload

# 也可以用代码实现
if __name__ == "__main__":
    uvicorn.run(app, host="localhost", port=8000)

# 生成环境运行参数
# uvicorn main:app --reload  --host=localhost --port=8000
```

约定和规范

- 项目结构

```
├── main.py               # 主应用文件
├── config.py             # 配置文件
├── run.py                # 启动脚本
├── requirements.txt      # 项目依赖
├── .env                  # 环境变量
├── README.md             # 项目说明
├── routers/              # 路由
├──
├── models/               # 数据模型
├──
├── schemas/              # 数据架构
├──
├── repositories/         # 数据库操作
├──
├── services/             # 业务逻辑
├──
└── utils/                # 工具类
```

公共日志类

```
# common_logger.py

import logging
import sys

__all__ = ["get_logger"]

# 1. 只初始化一次
_INITIALIZED = False

# 2. 颜色表 + Windows 兼容
COLORS = {
    "DEBUG": "\033[36m",
    "INFO": "\033[32m",
    "WARNING": "\033[33m",
    "ERROR": "\033[31m",
    "CRITICAL": "\033[35m",
```

```
    "RESET": "\033[0m",
}


def _init_logger(level: int = logging.INFO):
    global _INITIALIZED
    # force=True □□□□
    if _INITIALIZED:
        return

    class ColorFormatter(logging.Formatter):
        def format(self, record):
            levelname = record.levelname
            record.levelname = (
                f"{COLORS.get(levelname, '')}{levelname}{COLORS['RESET']}"
            )
            return super().format(record)

    handler = logging.StreamHandler(sys.stdout)
    handler.setFormatter(
        ColorFormatter(
            fmt="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
            datefmt="%Y-%m-%d %H:%M:%S",
        )
    )
    logging.basicConfig(level=level, handlers=[handler], force=True)
    # force=True □□□□
    _INITIALIZED = True


def get_logger(name: str, level: int = logging.INFO) -> logging.Logger:
    _init_logger(level)
    return logging.getLogger(name)
```

uvicorn □□□□□

```
#! /usr/bin/env python3
"""
FastAPI □□□□□□
"""

import uvicorn
from config import settings
from common_logger import get_logger

log = get_logger(__name__)

LOGGING_CONFIG = {
    "version": 1,
    "disable_existing_loggers": False,
    "formatters": {
        "default": {
            "fmt": "%(asctime)s - %(name)s - %(levelname)s - %(message)s",
        },
        "access": {
            "fmt": "%(asctime)s - %(name)s - %(levelname)s - %(message)s",
        },
    },
}


if __name__ == "__main__":
    log.info(f"□□ {settings.app_name} v{settings.app_version}")
    log.info(f"□□□□□: http://{settings.host}:{settings.port}")
    # print(f"API □□: http://{settings.host}:{settings.port}/docs")
    log.info("-" * 50)

    uvicorn.run(
        "main:app",
        host=settings.host,
        port=settings.port,
        reload=settings.reload,
        log_level=settings.log_level.lower(),
        log_config=LOGGING_CONFIG,
```

```
)
```

fastapi 路由管理

定义包并编写路由

```
# routers\zlmedia_hook.py

from fastapi import APIRouter


router = APIRouter(prefix="/zlmedia/hook", tags=["zlmedia_hook"])


@router.get("/")
async def hello():
    return {"code": "0", "message": "success", "data": "hello world"}
```

自动加载路由

```

# main.py

import routers.zlmedia_hook as zlmedia_hook


# 附件路由
app.include_router(zlmedia_hook.router)
```

fastapi项目结构

```
.
├── main.py                # 主应用文件
├── config.py              # 配置文件
├── run.py                 # 启动脚本
├── requirements.txt       # 项目依赖
├── .env                   # 环境变量
├── README.md              # 项目说明
├── routers/               # 路由
├── models/                # 数据模型
├── schemas/               # 数据架构
├── repositories/          # 数据库操作
├── services/              # 业务逻辑
└── utils/                 # 工具类
```

公共日志类

```
# common_logger.py

import logging
import sys

__all__ = ["get_logger"]

# 1. 只初始化一次
_INITIALIZED = False

# 2. 颜色表 + Windows 兼容
COLORS = {
    "DEBUG": "\033[36m",
    "INFO": "\033[32m",
    "WARNING": "\033[33m",
    "ERROR": "\033[31m",
    "CRITICAL": "\033[35m",
```

```
    "RESET": "\033[0m",
}


def _init_logger(level: int = logging.INFO):
    global _INITIALIZED
    # force=True □□□□
    if _INITIALIZED:
        return

    class ColorFormatter(logging.Formatter):
        def format(self, record):
            levelname = record.levelname
            record.levelname = (
                f"{COLORS.get(levelname, '')}{levelname}{COLORS['RESET']}"
            )
            return super().format(record)

    handler = logging.StreamHandler(sys.stdout)
    handler.setFormatter(
        ColorFormatter(
            fmt="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
            datefmt="%Y-%m-%d %H:%M:%S",
        )
    )
    logging.basicConfig(level=level, handlers=[handler], force=True)
    # force=True □□□□
    _INITIALIZED = True


def get_logger(name: str, level: int = logging.INFO) -> logging.Logger:
    _init_logger(level)
    return logging.getLogger(name)
```

uvicorn □□□□□

```
#! /usr/bin/env python3
"""
FastAPI □□□□□□
"""

import uvicorn
from config import settings
from common_logger import get_logger

log = get_logger(__name__)

LOGGING_CONFIG = {
    "version": 1,
    "disable_existing_loggers": False,
    "formatters": {
        "default": {
            "fmt": "%(asctime)s - %(name)s - %(levelname)s - %(message)s",
        },
        "access": {
            "fmt": "%(asctime)s - %(name)s - %(levelname)s - %(message)s",
        },
    },
}


if __name__ == "__main__":
    log.info(f"□□ {settings.app_name} v{settings.app_version}")
    log.info(f"□□□□□: http://{settings.host}:{settings.port}")
    # print(f"API □□: http://{settings.host}:{settings.port}/docs")
    log.info("-" * 50)

    uvicorn.run(
        "main:app",
        host=settings.host,
        port=settings.port,
        reload=settings.reload,
        log_level=settings.log_level.lower(),
        log_config=LOGGING_CONFIG,
```

```
)
```

□□□□

from□□□json body□□□□□□□□□

Query□ Form□ Body□ □□□□ ,□□□□□□□□□□□□□□□□□□□□body□□

□□□□□from□□

```
@app.post("/query_1")
def query_1(q: Any = (None), f: Any = (None)):
    return {"code": 0, "message": "success", "data": {"q": q, "f": f}}
```

□□□□□json body

```
class B_Model(BaseModel):
    b: str | int | Any = None


@app.post("/query_2")
def query_2(
    q: str = Query(None),
    b: B_Model = Body(None),
    token: str = Header(None, alias="token"),
    user: str = Cookie(None, alias="user"),
):
    return {
        "code": 0,
        "message": "success",
        "data": {"q": q, "b": b, "token": token, "user": user},
    }
```

使用中间件（拦截器）

标准做法

```
# LoginMiddleware.py

from fastapi import  Request,status
from starlette.middleware.base import BaseHTTPMiddleware
from fastapi.responses import JSONResponse

WHITE_LIST={
    "/",
    "/docs",
    "/health"
}

# 需要登陆中间件
class  LoginMiddleware(BaseHTTPMiddleware):
    async def dispatch(self,request: Request, call_next):
        token = request.headers.get("token", "")
        if request.url.path in (WHITE_LIST):
            return await call_next(request)
        if not token:
            return JSONResponse(
                status_code=status.HTTP_401_UNAUTHORIZED,
                content={"code": 401, "message": "Missing or invalid token"}
            )

        response = await call_next(request)
        return response

# 中间件示例
class  XPowerByMiddleware(BaseHTTPMiddleware):
    async def dispatch(self,request: Request, call_next):
        response = await call_next(request)
        response.headers["x-powered-by"]="iovhm-com © 2022"
        return response
```

```
# main.py

from LoginMiddleware import LoginMiddleware,XPowerByMiddleware

app.add_middleware(LoginMiddleware)
app.add_middleware(XPowerByMiddleware)
`
```

□□□□□

```
# □□
pip install sqlalchemy aiomysql -i https://pypi.tuna.tsinghua.edu.cn/simple
```