04-异常处理器代码片段

异常处理器 Python

```python
import traceback

from fastapi import HTTPException, Request
from fastapi.responses import JSONResponse
from sqlalchemy.exc import IntegrityError, SQLAlchemyError
from starlette import status

# 开发模式：返回详细错误信息
# 生产模式：返回简化错误信息
DEBUG_MODE = True  # 教学项目保持开启


async def http_exception_handler(request: Request, exc: HTTPException):
    """
    处理 HTTPException 异常
    """
    # HTTPException 通常是业务逻辑主动抛出的，data 保持 None
    return JSONResponse(
        status_code=exc.status_code,
        content={
            "code": exc.status_code,
            "message": exc.detail,
            "data": None
        }
    )


async def integrity_error_handler(request: Request, exc: IntegrityError):
    """
    处理数据库完整性约束错误
    """
    error_msg = str(exc.orig)

    # 判断具体的约束错误类型
    if "username_UNIQUE" in error_msg or "Duplicate entry" in error_msg:
        detail = "用户名已存在"
    elif "FOREIGN KEY" in error_msg:
        detail = "关联数据不存在"
    else:
        detail = "数据约束冲突，请检查输入"

    # 开发模式下返回详细错误信息
    error_data = None
    if DEBUG_MODE:
        error_data = {
```

```
            "error_type": "IntegrityError",
            "error_detail": error_msg,
            "path": str(request.url)
        }

    return JSONResponse(
        status_code=status.HTTP_400_BAD_REQUEST,
        content={
            "code": 400,
            "message": detail,
            "data": error_data
        }
    )


async def sqlalchemy_error_handler(request: Request, exc: SQLAlchemyError
):
    """
    处理 SQLAlchemy 数据库错误
    """
    # 开发模式下返回详细错误信息
    error_data = None
    if DEBUG_MODE:
        error_data = {
            "error_type": type(exc).__name__,
            "error_detail": str(exc),
            "traceback": traceback.format_exc(),
            "path": str(request.url)
        }

    return JSONResponse(
        status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
        content={
            "code": 500,
            "message": "数据库操作失败，请稍后重试",
            "data": error_data
        }
    )


async def general_exception_handler(request: Request, exc: Exception):
    """
    处理所有未捕获的异常
    """
    # 开发模式下返回详细错误信息
    error_data = None
    if DEBUG_MODE:
        error_data = {
```

```
            "error_type": type(exc).__name__,
            "error_detail": str(exc),
            # 格式化异常信息为字符串，方便日志记录和调试
            "traceback": traceback.format_exc(),
            "path": str(request.url)
        }

    return JSONResponse(
        status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
        content={
            "code": 500,
            "message": "服务器内部错误",
            "data": error_data
        }
    )
```