基于Excel的英语词汇辅助学习系统 V1.0

软件源程序

src/main.py

```
import yaml
# 注意：这里引用路径可能需要根据实际运行环境调整，此处保留源码逻辑
from config import load_config
from utils import setup_logging, create_backup, set_format
from processor import WordProcessor
def main():
    """主程序入口"""
    # 加载配置
    config = load_config()
    # 初始化日志
    setup_logging(config.get('log_level', 'INFO'))
    file_path = config['file_path']
    # 创建备份
    create_backup(file_path)
    # 初始化处理器并执行
    processor = WordProcessor(file_path, config)
    processor.process_all_sheets()
    # 保存结果
    processor.save()
    # 应用格式化
    set_format(file_path, config)
if __name__ == "__main__":
    main()
```

src/config.py

```
import yaml
def load_config(config_path='../config/config.yaml'):
    """读取配置文件并返回配置字典"""
    try:
        with open(config_path, 'r', encoding='utf-8') as f:
            return yaml.safe_load(f)
    except Exception as e:
        print(f"读取配置文件失败：{e}")
        return {}
```

config/config.yaml

```
file_path: "../xlsx/words.xlsx"
columns:
  word: 1
  phonetic: 2
  definition: 3
  example: 4
  status: 6
  wrong_history: 7
request:
  delay: 0.3
  retries: 3
  user_agent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
     Safari/537.3"
format:
  font:
    name: "Arial"
    size: 12
  sort: True
```

src/fetcher.py

```
import requests
from bs4 import BeautifulSoup
import time
import logging
def get_word_info(word, config):
    """从有道词典获取单词信息"""
    url = f"https://dict.youdao.com/w/{word}/"
    headers = {'User-Agent': config['request']['user_agent']}
    retries = config['request']['retries']
    for attempt in range(retries):
        try:
            response = requests.get(url, headers=headers)
            if response.status_code == 200:
                soup = BeautifulSoup(response.text, 'html.parser')
                # 获取音标
                phonetic = soup.find('span', class_='phonetic')
                phonetic_text = phonetic.text if phonetic else ''
                # 获取释义
                trans_container = soup.find('div', class_='trans-
                   container')
                definition = ''
                if trans_container and trans_container.find('ul'):
                    definition = trans_container.find('ul').find('
                       li').text or ''
                # 获取例句
                examples = soup.find('div', class_='examples')
                example_sentence = examples.find('p').text if
                   examples and examples.find('p') else ''
                return phonetic_text, definition, example_sentence
            else:
                logging.warning(f"请求失败，状态码：{response.
                   status_code}")
        except Exception as e:
            logging.error(f"获取单词 {word} 信息出错：{e}")
        if attempt < retries - 1:
            time.sleep(4)
    logging.error(f"单词 {word} 获取失败，已记录")
    with open('failed_words.txt', 'a', encoding='utf-8') as f:
        f.write(f"{word}\n")
    return None, None, None
```

src/processor.py

```
import sys
from openpyxl import load_workbook
import time
from fetcher import get_word_info
import logging
class WordProcessor:
    def __init__(self, file_path, config):
        self.file_path = file_path
        self.config = config
        self.wb = load_workbook(file_path)
        self.cols = config['columns']
    def process_sheet(self, sheet):
        """处理单个工作表"""
        # 初始化表头
        if sheet.cell(row=1, column=self.cols['status']).value !=
           '处理状态':
            sheet.cell(row=1, column=self.cols['status']).value =
               '处理状态'
        if sheet.cell(row=1, column=self.cols['wrong_history']).
           value != '错误历史':
            sheet.cell(row=1, column=self.cols['wrong_history']).
               value = '错误历史'
        for row_idx in range(2, sheet.max_row + 1):
            # 确保行数据足够长，如果列不足，插入新列
            if sheet.max_column < self.cols['wrong_history']:
                sheet.insert_cols(self.cols['wrong_history'])
            # 获取单元格对象
            word_cell = sheet.cell(row=row_idx, column=self.cols['
               word'])
            status_cell = sheet.cell(row=row_idx, column=self.cols
               ['status'])
            wrong_history_cell = sheet.cell(row=row_idx, column=
               self.cols['wrong_history'])
            # 状态检查
            if status_cell.value == '已处理':
                logging.info(f"跳过已处理单词: {word_cell.value}")
                continue
            if status_cell.value == '失败':
                logging.info(f"跳过失败单词: {word_cell.value}")
                continue
            # 开始处理
            if word_cell.value and status_cell.value is None:
                phonetic, definition, example = get_word_info(
                   word_cell.value, self.config)
                sheet.cell(row=row_idx, column=self.cols['phonetic
                   ']).value = phonetic if phonetic else 'N/A'
                sheet.cell(row=row_idx, column=self.cols['
                   definition']).value = definition if definition
                    else 'N/A'
                sheet.cell(row=row_idx, column=self.cols['example
                   ']).value = example if example else 'N/A'
                if not phonetic or not definition or not example:
                    wrong_history_cell.value = word_cell.value
                    if not phonetic and not definition and not
                        example:
                         status_cell.value = '失败'
                         logging.error(f"处理失败: {word_cell.value
                             }")
                         # 写入失败日志
                         with open('../failed_words.txt', 'a',
                             encoding='utf-8') as f:
                               f.write(f"{word_cell.value} in {sheet
                                   .title}\n")
                    else:
                         logging.warning(f"部分成功: {word_cell.
                             value}")
                status_cell.value = '已处理'
                logging.info(f"已处理: {word_cell.value}")
                time.sleep(self.config['request']['delay'])
    def process_all_sheets(self):
        """处理所有工作表"""
        for sheet_name in self.wb.sheetnames:
            logging.info(f"开始处理工作表: {sheet_name}")
            self.process_sheet(self.wb[sheet_name])
    def save(self):
        """保存工作簿"""
        self.wb.save(self.file_path)
        logging.info(f"处理完成, 保存到 {self.file_path}")
```

src/utils.py

```
import shutil
import os
import sys
from datetime import datetime
from openpyxl import Workbook, load_workbook
from openpyxl.styles import Font, Alignment
import logging
def setup_logging(log_level='INFO'):
    """初始化日志配置"""
    logging.basicConfig(
        filename='word_processor.log',
        level=getattr(logging, log_level.upper()),
        format='%(asctime)s - %(levelname)s - %(message)s'
    )
def create_backup(file_path):
    """创建文件备份"""
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    file_name = os.path.basename(file_path)
    dir_name = os.path.dirname(file_path)
    if not os.path.exists(f"{dir_name}/backup/"):
       os.makedirs(f"{dir_name}/backup/")
    backup_path = (f"{dir_name}/backup/" + f"{file_name}_backup_{
       timestamp}.xlsx")
    shutil.copy(file_path, backup_path)
    logging.info(f"备份已创建: {backup_path}")
    return backup_path
def set_format(file_path, config):
    """设置格式（保留原有颜色）"""
    try:
        wb = load_workbook(file_path)
        new_font_name = config['format']['font']['name']
        new_font_size = config['format']['font']['size']
        for sheet in wb.worksheets:
            for row in sheet.iter_rows():
                for cell in row:
                    # 1. 获取单元格当前的字体对象
                    current_font = cell.font
                    # 2. 创建一个新的Font对象，保留原有颜色和属性
                    updated_font = Font(
                        name=new_font_name,
                        size=new_font_size,
                        bold=current_font.bold,
                        italic=current_font.italic,
                        color=current_font.color,  # 保留原有颜色！
                        strike=current_font.strike,
                        underline=current_font.underline,
                        vertAlign=current_font.vertAlign,
                        charset=current_font.charset,
                        scheme=current_font.scheme,
                        family=current_font.family,
                        outline=current_font.outline,
                        shadow=current_font.shadow,
                    )
                    cell.font = updated_font
                    cell.alignment = Alignment(wrap_text=True)
        wb.save(file_path)
        logging.info(f"字体已设置 (保留原有颜色): {file_path}")
    except Exception as e:
        logging.error(f"处理文件 {file_path} 时发生错误: {e}")
```