Excel 邮件调度器

一个 Python 工具，用于读取 Excel (.xlsx) 文件并根据可配置的模板发送个性化邮件。支持多种认证方式，包括基础 SMTP、Gmail OAuth2 和 Outlook OAuth2。

English Documentation

功能特性

- 从 Excel 文件 (.xlsx) 读取收件人数据
- 使用模板占位符发送个性化邮件
- 多种认证方式：
  - 基础 SMTP（QQ邮箱、163邮箱等）
  - Gmail OAuth2
  - Outlook/Office365 OAuth2
- 监控模式：定期检查新的 Excel 文件并自动发送邮件
- 测试模式（dry-run）：预览邮件而不实际发送
- 可配置的日志记录

系统要求

- Python 3.10+
- 依赖包：

```
pyyaml
openpyxl
```

OAuth2 认证额外依赖：

- Gmail：`google-auth-oauthlib`、`google-api-python-client`
- Outlook：`msal`、`requests`

安装

1. 克隆仓库：

```
git clone <repository-url>
cd excel-email-scheduler
```

2. 创建虚拟环境并安装依赖：

```
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
pip install pyyaml openpyxl
```

3. Gmail OAuth2 额外依赖：

```
pip install google-auth-oauthlib google-api-python-client
```

4. Outlook OAuth2 额外依赖：

```
pip install msal requests
```

配置说明

复制 `config.yaml` 并根据需要修改：

认证方式

基础 SMTP（适用于 QQ邮箱、163邮箱等）

```yaml
auth_type: "basic"

smtp:
  host: "smtp.qq.com"
  port: 587
  use_tls: true
  username: "your_email@qq.com"
  password: "your_app_password"  # 使用授权码，非登录密码
```

Gmail OAuth2

```yaml
auth_type: "gmail_oauth2"

gmail_oauth2:
  credentials_file: "gmail_credentials.json"
  token_file: "gmail_token.json"
  scopes:
    - "https://www.googleapis.com/auth/gmail.send"
```

配置步骤：

1. 访问 Google Cloud Console
2. 创建项目并启用 Gmail API
3. 创建 OAuth2 凭据（桌面应用类型）
4. 下载 JSON 文件并保存为 `gmail_credentials.json`

Outlook OAuth2

```
auth_type: "outlook_oauth2"

outlook_oauth2:
  client_id: "your_client_id"
  client_secret: "your_client_secret"
  tenant_id: "common"
  token_cache_file: "outlook_token_cache.json"
```

配置步骤：

1. 访问 Azure 门户
2. 在 Azure Active Directory 中注册新应用
3. 配置 API 权限（Mail.Send）
4. 创建客户端密钥

邮件模板

```
email:
  from_address: "your_email@example.com"
  subject: "你好 {Name}"
  body: |
    {Name} 您好，

    您的工号是 {OrderNumber}。

    祝好，
    团队
```

使用 `{列名}` 作为占位符，列名需与 Excel 表头一致。

工作目录

```
workspace:
  path: "./workspace"
```

将 `.xlsx` 文件放在此目录中。

调度器（监控模式）

```
scheduler:
  enabled: false
  check_interval: 86400  # 检查间隔（秒），默认：1天
  processed_files_record: "./processed_files.json"
```

常用时间间隔：

- 1 小时：3600
- 1 天：86400
- 1 周：604800

Excel 文件格式

Excel 文件要求：

- 第一行：列标题
- 列名包含 "Leader" 的列将被用作邮件收件人地址

示例：

Name	Email	Leader Email	OrderNumber
张三	zhangsan@example.com	leader1@example.com	12345
李四	lisi@example.com	leader2@example.com	12346

使用方法

一次性处理

处理工作目录中的所有 Excel 文件：

```
python excel-email-scheduler.py
```

测试模式

预览邮件而不实际发送：

```
python excel-email-scheduler.py --dry-run
```

监控模式

持续监控新的 Excel 文件：

```
python excel-email-scheduler.py --watch
```

或在配置文件中启用：

```

```

```
scheduler:
  enabled: true
```

指定配置文件

```
python excel-email-scheduler.py --config /path/to/config.yaml
```

命令行参数

参数	说明
`--config PATH`	配置文件路径（默认：config.yaml）
`--dry-run`	测试模式，预览邮件而不发送
`--watch`	监控模式

监控模式工作原理

1. 启动时检查新的/修改过的 Excel 文件
2. 处理新文件并发送邮件
3. 将已处理的文件记录到 `processed_files.json`
4. 等待配置的时间间隔
5. 重复检查

文件通过修改时间和文件大小进行追踪。修改过的文件会被重新处理。

按 `Ctrl+C` 可优雅地停止调度器。

日志配置

在 `config.yaml` 中配置日志：

```
logging:
  level: "INFO"  # DEBUG, INFO, WARNING, ERROR
  file: "./email_scheduler.log"
```

许可证

MIT License