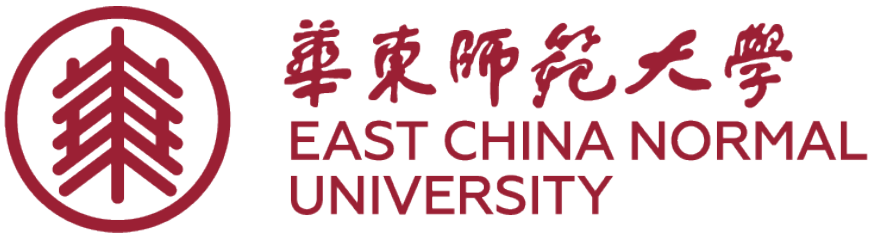


ECNU 校园插件项目报告

ECNU Campus Plugins - Project Report

关卓谦 张梓卫 王文锦

指导老师：陈良育

2025 年 1 月 4 日

华东师范大学软件工程专业

目录

一 项目概述

1 项目提出背景

Implementation — .1

本项目主题基于 ECNU 校园生活，通过全自动化获取登录缓存，应用置于托盘处自动跟随插件周期调用事件，从而实现 ECNU 校园的智能化。

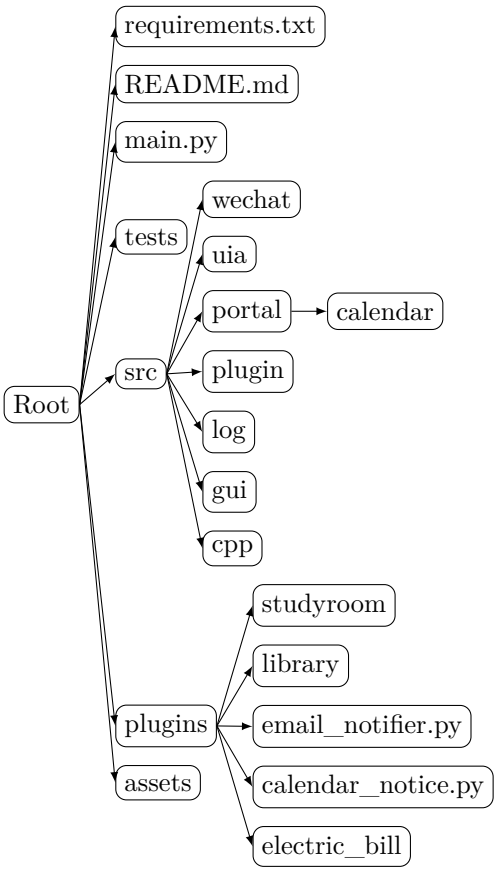
为了提升就读体验和同学们的生活质量，我们团队开发的本项目实现了：

- 课表一键生成、课前邮件提醒
- 课后研修间与图书馆的全自动预约
- 宿舍电费自动查询及充值提醒
- 适配 ECNU 多种系统的登录缓存，实现了插件框架，能够接收更多开发者的贡献与集成

2 项目相关信息

- 项目作者：关卓谦（10235101529）、张梓卫（10235101526）、王文锦（10235101510）
- 项目仓库（含部署方法及 Wiki 文档）：https://github.com/azazo1/ecnu-campus-plugins
- 本项目遵循 MIT 开源协议，项目仅用于课程学习交流，不用于商业用途。
- 本项目使用 Python 3.12 开发，基于 Pyside 6 实现图形界面。

二 项目整体架构



项目结构：
图中展示了项目的层次结构，主要组件包括：

- **assets:** 静态资源目录，包含各开发参考文件及图标等。
- **plugins:** 插件主目录，包含 `electric_bill`、`studyroom` 和 `library` 等模块。
- **src:** 主源代码目录，按照功能分为子模块，例如 `gui`、`portal`、`uia` 等。
- **tests:** 测试用例目录，包含对各模块的单元测试。
- **main.py:** 项目主入口。

三 项目模块详述

1 ECNU 登录缓存获取模块

万事开头难，打好地基方能致远。项目的根便在于此，观察 UIA 登录界面，验证码这一步卡住了，所以我们考虑使用扫码登录或点击活性链接的登录方式。

图 1: ECNU SSO 登录界面

Remark

我们首先通过 Selenium 和 WebDriver 工具实现了半自动登录校园网站的功能。

最初，我们的设想是通过发送邮件或微信消息的方式，使用户不定时点击以保持 Cache 的活性，为此我们设计了以下的内容来实现：

1 .1 自动化登录工作流程

1. **初始化**：使用 Webdriver 启动浏览器并导航到 ECNU 座位筛选页面，该页面会重定向到统一认证登录页面。
2. **基于二维码的登录**：
   - 如果未找到凭证，则进行二维码登录：使用 CSS 选择器定位到二维码登录按钮；
   - 之后提取二维码图片，解码以获取登录 URL，并将图片保存为临时文件；
   - 使用回调函数（`qrcode_callback`）将二维码发送给用户（例如，通过邮件或微信）。
   - 监控登录状态，处理二维码过期后的刷新。
3. **缓存提取**：
   - 登录成功后，调用一系列 `cache_grabbers` 函数提取必要的缓存数据。
   - 将这些缓存存储在 `LoginCache` 实例中，以供其他组件或插件使用。

Note

以上代码可以进入 `src/uia/login.py` 文件中查看，其中，核心函数 `get_login_cache` 已提供了完整的注释信息。但是，我们获得的 `login_cache` 是会失活而非永久的，这一点通过查阅华东师范大学开发者文档可以得知：https://developer.ecnu.edu.cn/vitepress/data/architecture/authorization.html

图 2: 缓存过期时间

1 .2 通过 OCR 识别验证码登录

后来意识到，这样需要手动点击链接来刷新缓存的方式，违背了我们全自动化、便捷的初心，于是我们决定攻破验证码的防线，便在 Github 上寻找到了一个较为优秀的项目：（DDDDOCR）：https://github.com/sml2h3/ddddocr。

我们使用一段 JavaScript 代码成功提取了验证码图片，它是通过 `<img>` 标签加载在网页中的：

```
EXTRACT_CAPTCHA_IMG_JS = f"""
let img = document.querySelector("{CAPTCHA_IMG_SELECTOR}");
let canvas = document.createElement('canvas');
let ctx = canvas.getContext('2d');

# 使用 canvas 画布在前端绘制图片，之后保存到文件中供 OCR 识别
canvas.width = img.width;
canvas.height = img.height;
ctx.drawImage(img, 0, 0, canvas.width, canvas.height);
return canvas.toDataURL('image/png');
"""
```

有时 OCR 的结果不是四位有效的验证码，所以我们添加了一行代码增大了准确性：

```
if len(captcha) != 4:
    driver.refresh()
```

尝试使用后，它有将近 $\frac{1}{3}$ 识别的准确率。此后，便可以实现自动化登录，解放双手了。较为幸运的是，华师并未设置验证码多次登录失败后禁止登录的逻辑，所以成功率较低的情况下也可以照常使用。

最终，我们获取的 `LoginCache` 置于项目根目录中的文件 `login-cache.pickle` 中，其中最重要的字段为 **Bearer** 字段以及 **cookie** 字段，这是在 **OAuth 2.0** 框架下在 HTTP 请求头中的认证类型，携带访问令牌通过 **Bearer Token** 来实现无状态的认证。

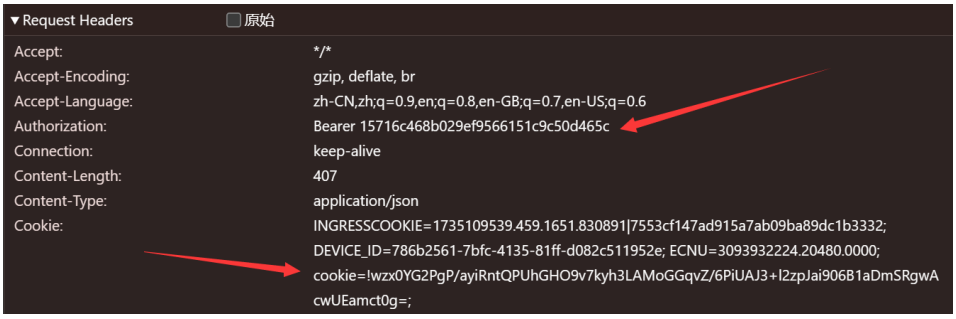


图 3: Main Cookie 及鉴权 Token

华东师范大学开发者平台提供了一个 OAuth 2.0 的 Playground 调试平台，教师可进入以下网站进行在线调试：https://developer.ecnu.edu.cn/oauth2playground/

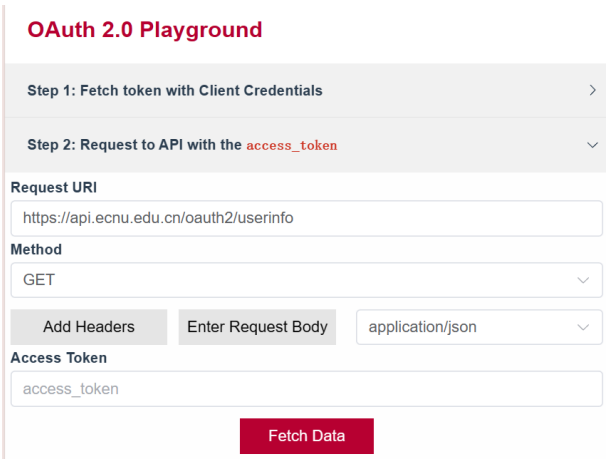
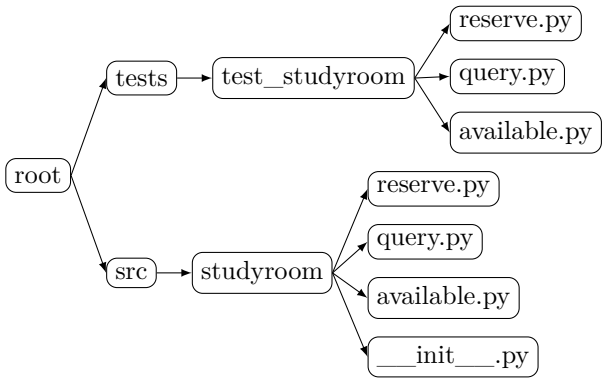


图 4: OAuth 2.0 Playground

2 研修间预约模块

2 .1 代码框架及分析

此部分代码位于 `src/studyroom/` 中。



模块说明：

Query (`query.py`)
该模块仅实现对 `url` 的简单请求，不作任何数据处理。它通过相应的 API 获取研修室的可用性和详细信息。

Available (`available.py`)
`available.py` 模块负责处理房间可用性数据。它分析当前和已预订的时间段，确定研修室可预约的时间段。

Reserve (`reserve.py`)
`reserve.py` 模块管理预约流程。它包括进行预约、检查预约状态以及在特定条件下自动取消预约的功能。

图 5: 研修间的目录结构及模块说明

2 .2 研修间预约实现原理

华东师范大学研修间预约最少预约时长在 60 分钟以上，采用贪心思想，我们尽量预约更长的时间段以供使用，即使不能使用还可以及时离开。

查看研修间的预约情况，我们首先需要计算出可用时间都有哪些：

图 6: 研修间状态示例

研修间预约时，需要提交的表单样式如下，我们不采用浏览器自动操作的形式，而是截取提交时发送的 url 请求：

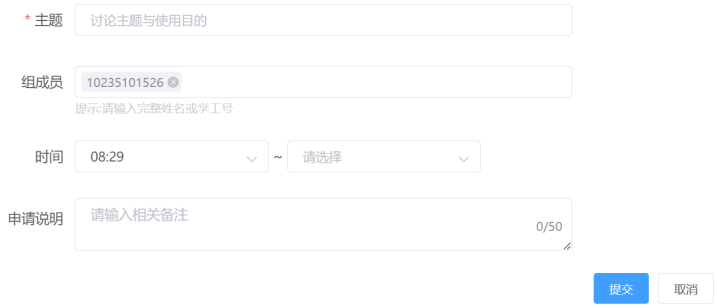


图 7: 研修间预约表单

抓包得到 POST 请求的 Url 如下：https://studyroom.ecnu.edu.cn/ic-web/reserve
只需传入相关的字段即可：

图 8: 研修间预约请求

```python
headers = {
    "Cookie": f"ic-cookie={ic_cookie}",
} # 从 StudyroomCache 中获取 ic-cookie

# 从 _fetch_userInfo 获取用户 ID
appAccNo = self._fetch_userInfo().get("accNo"
    )

payload = {
    "sysKind": 1, # 系统类型，默认为 1
    "appAccNo": appAccNo,
    "memberKind": 1, # 成员类型，默认为 1
    "resvBeginTime": resvBeginTime,
    "resvEndTime": resvEndTime,
    "testName": testName,
    "resvMember": [appAccNo],
    # 默认预约人员列表只有当前用户
    "resvDev": resvDev,
    "memo": memo,
}
```

Listing 1: 预约请求需要传入的字段

加载入网页时，我们抓包发现，URL：URL: https://studyroom.ecnu.edu.cn/ic-web/roomDevice/roomAvailable 拥有查询当前类别的研修间的功能，其返回的字段如下：

```
{'devId': 3676503,                          # 设备 ID
 'devName': '普陀校区单人间C421',            # 设备名称
 'minResvTime': 60,                         # 最小预约时间
 'openTimes': [{'openEndTime': '22:00',     # 开放结束时间
                'openLimit': 1,             # 最少预约人数
                'openStartTime': '08:00'}], # 开放开始时间
 'resvInfos': [{'resvBeginTime': '2024-12-26 '  # (People No.1) 的预约信息
                                 '17:01:00',
                'resvEndTime': '2024-12-26 '
                               '21:01:00',
                'resvStatus': 1093}]},
```

它只为我们提供了 'resvInfos' 字段，这应该是用于前端供渲染黄色已预定时间段的给用户的，所以我们需要将它与 'openTimes' 字段结合起来，得到研修室的可用时间段。

Note

- 将 [openStartTime, currentTime] 区间视为不可用时间段。例如当天 18:55 P.M 前的时间段都无效。
- 仅 AvailableTime > minResvTime 时，才将这段区间设置为 AvailableTime, 如果不超过最小预约时间，则不予考虑。

返回的信息字典中，我们通过程序包含了一个新字段 availableInfos，这样就可以提供给用户可用的时间段了。

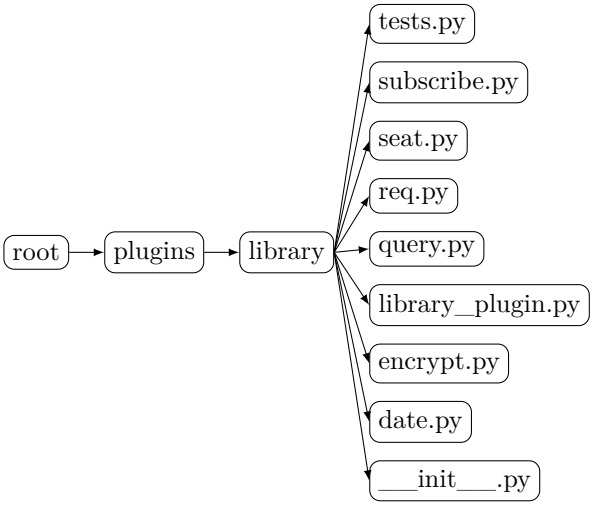
之后，使用 reserve.py 中的函数 _fetch_userInfo 来获取 appAccNo，这是识别用户身份的字段，在发送 reserve 请求时，需要传入该字段作为 Payload 的一部分。最后通过调用 reserve_room 函数即可完成全自动预约。

Note

每一次预约都会形成一个唯一的预约编号，称为 uuid，我们在进入个人中心时，网页会自动调用查询的接口：https://studyroom.ecnu.edu.cn/#/ic/userinfo ，所以，我们也可以通过抓包获取 uuid，便可以知道用户当前是否有预约了，这为后续的自动取消预约提供了保障。

3 图书馆预约模块

3 .1 代码框架及分析



模块说明：

Date (`date.py`)
负责处理日期相关功能，例如格式化和时间操作。

Encrypt (`encrypt.py`)
提供加密和解密功能，用于数据传输安全。

图 9: ECNU 图书馆预约模块说明

3 .2 图书馆预约实现原理

4 课前提醒模块

4 .1 代码框架及分析

4 .2 课表获取及数据处理原理

图 10: ECNU Portal 课表页面

Remark

进入电脑端 Portal 主页面右侧，我们可以看到自己的课表，使用抓包工具得知，实际上，本课表是存在一个请求 url 的。

那么我们通过上述获得的 `login_cache`，通过 requests 库发起请求，即可实时获取课表。

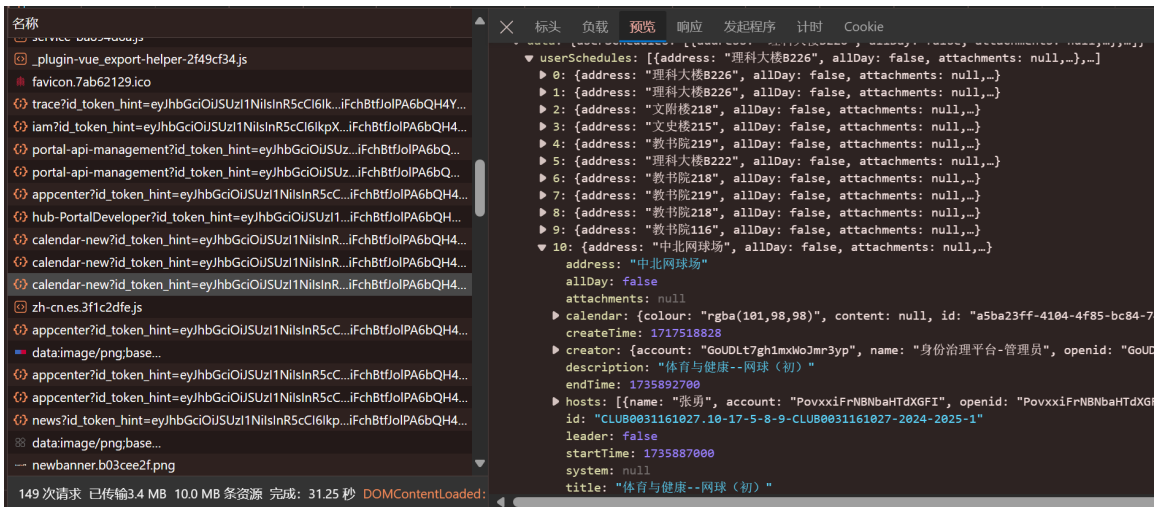


图 11: Portal 课表请求

这个 POST 请求采用的是 GraphQL 的查询格式，我们只需要使用 filter 过滤器查询自己所需要的字段即可。

```
USER_SCHEDULES = """
query ($filter: ScheduleFilter, $userId: String) {
  userSchedules(filter: $filter, userId: $userId) {
    address # 上课地点
    hosts {
      name # 教师名字
    }
    description # 课程信息和描述
    endTime # 结束时间
    startTime # 开始时间
  }
}
"""
```

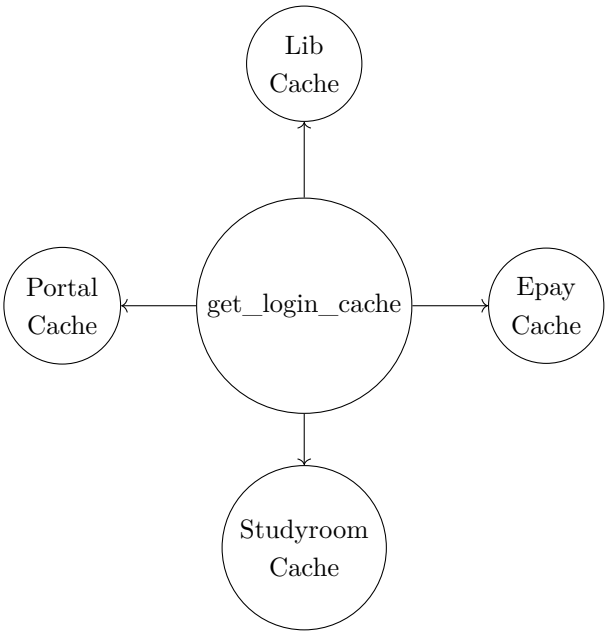
我们原定的设计是查询当前时刻至次日该时刻，用户的课程安排，如果有课程，上课前的一定时段发送邮件提醒用户去上课。可以通过插件配置来配置上课前多久提醒用户，相关内容在后面进行详述。

Remark

该功能配置一次后，便可以在后台轮询调用，后续我们考虑加入手机端的定位功能，查询用户是否已经到达指定的上课地点，便不发送邮件。

5 插件装载（PluginLoader）模块

5 .1 Cache 分发



PluginLoader 缓存分发功能：

- **Lib Cache：** 用于图书馆管理系统。
- **Studyroom Cache：** 用于研修间自动预约系统。
- **Portal Cache：** 用于公共数据库页面的缓存，目前仅用于课表的获取。
- **Epay Cache:** 用于华东师范大学校园卡管理页面的自动查询电费功能。
- **More Cache...：** 适配的框架可以获取其余 ECNU 页面的 Cache 以供开发者使用。

功能说明：PluginLoader 是一个插件管理器，它可以加载、卸载和管理插件。通过调用 `get_login_cache()` 函数，PluginLoader 实现各缓存的分发，各缓存分发后供相应的插件调用。

Remark

更多适配：我们已经实现了基本的 Cache 抓取框架，以供后续的开发者调用，调用示例如下：

```
get_login_cache(cache_grabbers=[{MoreCache}.grab_from_driver])
```

除此之外，每一个插件都拥有自己的 Routine 事件周期，Plugin 在注册时需要附带该属性，以保证 PluginLoader 能够轮询执行插件的例行任务（通过 PluginLoader 中定义的 `poll()` 函数来执行）。

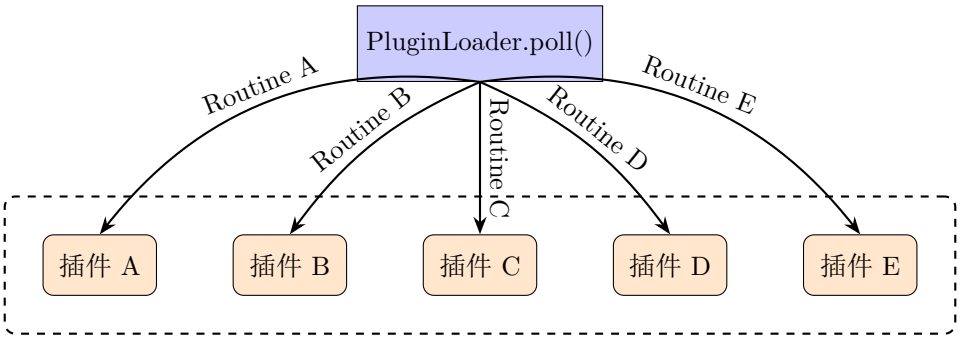
5 .2 基于 Routine 的事件调度逻辑

Note

PluginLoader 根据预定义的频率（Secondly、Minutely、Hourly、Daily、Weekly），调用每个插件的 `on_routine` 方法。插件会根据任务频率接收并处理任务，完成对应的功能逻辑。

箭头上的"Routine" 统一表示调度逻辑，具体任务频率的细节由配置文件决定。例如：

- `Routine: Secondly` → 调用插件 A 的 `on_routine`。
- `Routine: Daily` → 调用插件 D 的 `on_routine`。



5 .3 插件配置管理

> **Note**
>
> 此部分代码如 root/src/plugin/config.py 所示。
> 我们使用装饰器来实现插件的注册，注册信息存储于 Registry 类的字典中，每个插件对应着一个 Record 对象。

示例性的插件注册代码如 root/plugins/calendar_notice.py 中所示：

```
@register_plugin(
    name="calendar_notice",
    configuration=PluginConfig().add(
        TimeItem(
            name="notice_before_class_start", default_value=datetime.time(0, 10),
            description="上课提前提醒时间 (提前h小时m分钟)"
        )
    ),
    routine=Routine.MINUTELY,
    ecnu_cache_grabber=PortalCache.grab_from_driver
)
```

文件中有 PluginConfig 和 PluginContext 类，它们的功能如下：

- **PluginConfig**：插件的配置项集合。插件可继承 PluginConfig 类并添加多个 ConfigItem 子类来描述其配置需求。
- **ConfigItem**：每个配置项通过继承 ConfigItem 类及其子类（如 TextItem、NumberItem 等）来定义。

功能说明：PluginConfig 提供了一种结构化的方式来定义插件的配置项，支持配置项的序列化和反序列化，方便配置的保存和加载。插件在注册时通过 PluginConfig 指定其需要的配置项，框架会自动处理配置的管理。

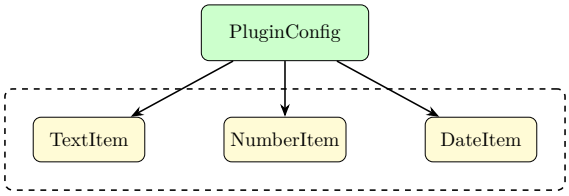
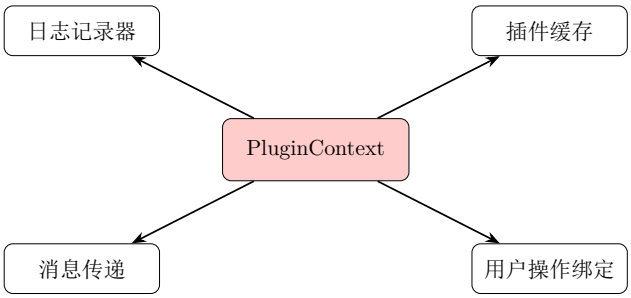


图 12: PluginConfig 和 ConfigItem 的关系示意图

5 .4 插件上下文管理

此部分代码见 root/src/plugin/context.py 所示。

> **Note**
>
> 为了保证项目的整洁和规范性，插件创建文件和记录日志等操作请使用生命周期函数和事件函数中提供的 PluginContext 进行。它为每个插件提供了一个上下文环境，使插件能够与框架进行交互和访问必要的资源。



PluginContext 的功能组成：

- **日志记录**：插件专属的日志记录（信息、警告、错误等）。
- **插件缓存**：插件可以通过 ctx.get_cache() 访问自己的持久化缓存，用于存储跨会话的数据。缓存支持数据的读取和写入，确保插件状态的持久性。
- **消息传递**：允许插件间通信。
- **用户操作绑定**：插件可以绑定用户界面的操作（如按钮），相应的回调函数相应用户的操作。

6 测试模块

在本项目中，我们采用了 Python 标准库中的 unittest 模块进行单元测试，以确保项目核心功能的稳定性和正确性。

- **测试环境的隔离**：通过 setUp() 和 tearDown() 方法，unittest 提供了对每个测试用例的独立初始化和资源回收，避免测试用例之间的相互干扰。
- **断言机制**：unittest 提供了多种断言方法（如 assertEqual()、assertTrue()），用以验证测试结果是否符合预期。

测试用例的结构设计：

以下是测试模块的代码结构：

```
class TestCalendar(unittest.TestCase):
    def setUp(self):
        init()  # 初始化日志记录器
        self.cache = load_cache()  # 加载登录缓存
        self.calendar = CalendarQuery(self.cache.get_cache(PortalCache))

    def test_user_schedules(self):
        now = datetime.datetime.now()
        pprint(self.calendar.query_user_schedules(
            int(now.timestamp() * 1000),
            int((now + datetime.timedelta(days=1)).timestamp() * 1000),
        ))

    def test_school_calendar(self):
        school_calendar = self.calendar.query_school_calendar()
        pprint(school_calendar)
```

测试框架的执行流程：

- 在 setUp() 方法中完成初始化工作，包括日志系统的初始化和登录缓存的加载。
- 使用 test_user_schedules() 方法测试用户日程的获取功能，通过时间戳计算和校验确保数据范围的准确性。
- 使用 test_school_calendar() 方法验证校历的查询功能，确保校历数据的完整性和准确性。

7 其他工具模块

7 .1 LateX 课表生成器

本模块位于 root/tools/classtable 之下。对应的测试文件为 test/test_latex。

在之前的课前提醒模块中，我们通过 GraphQL 的 filter 过滤器可以选中需要查询的时间戳，若要查询整个学期的课表，只需要将时间戳定于本周和下周即可，因为考虑到有一些课程是单周上的，这样能满足大部分人的需求。所以当我们获得了本周和下周的课表后，只需要做一次过滤，采用集合过滤法，指定一个字段作为 id，便能够实现两周课表的聚合。

```
    seen = set()
    unique_classes = []
    for cls in double_week_class_table:
        # 创建唯一标识：(课程, 星期, 时间)
        identifier = (cls.title, weekday)
        if identifier not in seen:
            seen.add(identifier)
            unique_classes.append(cls)
        else:
            project_logger.info(f"去除重复课程：{cls.title} 在星期{weekday + 1} {class_time}")
```

然后按照 cls 给出的 LaTeX 课表的格式，用 python 自动填充好，再使用 os 模块进行命令行编译即可。

图 13: LateX 课表生成示例

7 .2 基于 CPP 实现文件复制到系统剪贴板

> **Note**
>
> 本模块位于 `src/cpp/copyfile_build` 中。请查阅相关代码。
> 我们使用 `CF_HDROP` 数据格式进行 Windows 的文件复制入剪贴板，这一部分可参考 Microsort 写给开发者们的参考文档：https://learn.microsoft.com/en-us/windows/win32/shell/clipboard#cf_hdrop

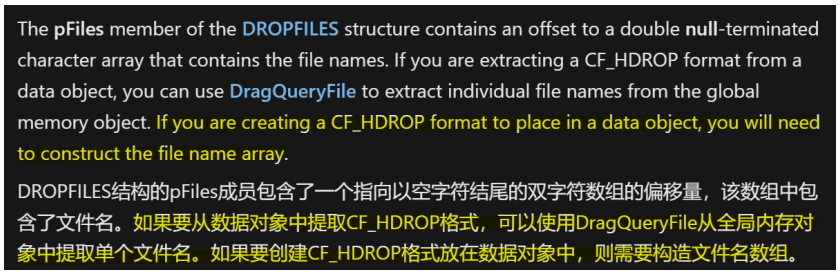


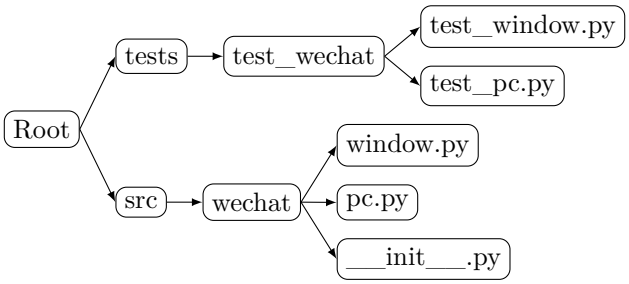
图 14: Microsoft Study Doc - CF_HDROP

CopyFileToClipboard 函数是用来将一个文件路径复制到剪贴板的，基于 Windows API 实现。使用 **CF_HDROP** 数据格式，它是 Windows 剪贴板的标准格式，用于表示文件路径。具体步骤：

1. 调用 **OpenClipboard** 打开剪贴板，清空剪贴板数据 (**EmptyClipboard**)。
2. 分配内存 (**GlobalAlloc**)，创建一个 **DROPFILES** 结构体，其中存储文件路径。
3. 使用 **GlobalLock** 锁定分配的内存并填充数据，调用 **SetClipboardData** 将文件路径放到剪贴板中。
4. 最后释放内存并关闭剪贴板 (**CloseClipboard**)。

7 .3 微信交互接口

> **Note**
>
> 微信交互接口是我们最初的设计设想，Python 接管键鼠实现自动化办公，主要使用的是 uiautomation 库，它可以实现对 Windows 系统的 UI 自动化操作。
>
> 在项目中，我们实现了部分和微信有关的接口，能够为后续的开发者们提供逻辑参考。在此我们提供一个示例，将生成的 LaTeX 课表通过 CPP 文件复制到微信指定窗口并发送。



- **描述**: 封装微信的各种操作，如发送消息、图片、文件等。
- **方法**:
  1. `open_window()`: 唤起微信窗口并获取窗口控制对象。
  2. `close_window()`: 关闭微信主窗口。
  3. `search(cls, pattern: str)`: 在微信搜索框中输入搜索内容并执行搜索。
  4. `locate_chat(cls, name: str | None = None)`: 获取聊天窗口的输入框控件。
  5. `switch_to(cls, name: str)`: 切换到指定名称的聊天窗口。
  6. `send_message(cls, name: str, text: str)`: 向指定聊天对象发送文本消息。
  7. `send_img(cls, name: str, img: str | typing.BinaryIO)`: 向指定聊天对象发送图片。
  8. `send_file(cls, name: str, filepath: str)`: 向指定聊天对象发送文件。

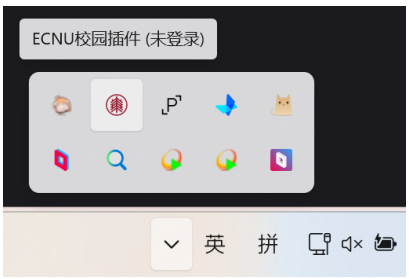
四 项目主要界面贴图

1 基于 Pyside 6 的主界面

2 插件配置页面

3 托盘后台界面

通过运行 main.py 后，托盘中会出现一个图标，当鼠标悬停于其上时，可以查看当前的登录状态，使用鼠标右键退出该插件程序。



(a) 处于任务栏的托盘

(b) 托盘退出按钮

图 15: 托盘界面与退出按钮

五 总结

1 项目前景

本项目的开发过程是我们团队不断挑战自我、解决问题的过程。通过项目，我们不仅掌握了多个技术栈（如 Python 自动化、插件框架设计、C++ 系统调用等），还提高了团队的协作能力、代码规范性和需求分析能力。

展望未来，本项目具备良好的扩展性，后续可以集成更多插件，覆盖更多校园生活场景。例如：

- 支持课表与日历软件（如 Google Calendar）的无缝同步。
- 集成更多校园生活服务，例如自动化导出成绩单、图书馆借阅管理、校园卡消费查询等。

通过持续迭代和优化，本项目有潜力为华东师范大学的师生带来更加便利的校园生活体验。

2 项目收获

关卓谦

Thought 五 .1

在本项目中

张梓卫

Thought 五 .2

我是本项目的组织者和相关 Idea 的提出者，也是该课程报告的主要撰写人，最初，我通过获取窗口 `PID` 实现了微信的简单接口，得益于良好的团队框架，后来使用团队中完善的 `Cache_grabber` 框架，让我能够马上上手研修间预约模块的开发，熟悉了数据处理和 API 调用的结合应用，当然，LateX 课表生成器的开发让我对编译有了更深入的理解，

在不断翻看关卓谦的 `PluginLoader` 的代码到理解，最终写出一份详解的报告从提出到实现，从实现到推翻重来，一切都有迹可循。通过团队交流，我增强了自己的跨领域的开发编程能力，掌握了 Git 多种不同的使用方式，同时跟随团队的项目规范，掌握了如何通过良好的代码架构提高代码复用性和维护性。

同时，我还学到了如何利用 Python 的自动化工具（Selenium-Wire 和 requests）实现复杂的登录流程。翻看华东师范大学开发者文档时，涉及到鉴权，还了解到了 OAuth 2.0 和 JWT 等的认证机制。

是非常愉悦的一次项目经历！

王文锦

Thought 五 .3

在本项目中

3 注意事项

- 本项目遵循 MIT 开源协议，LaTeX 宏包部分使用 LPPL 授权。
- 华东师范大学校徽图案版权归华东师范大学所有。
- 项目使用 Pycharm 开发，课程报告使用 LaTeX 编写。

4 参考资料

- https://learn.microsoft.com/en-us/windows/win32/shell/clipboard#cf_hdrop
- https://developer.ecnu.edu.cn/oauth2playground/
- https://github.com/sml2h3/ddddocr。
- https://developer.ecnu.edu.cn/vitepress/data/architecture/authorization.html