易场地软件系统重构报告

报告人：侯俊皓 夏宇翔 马萨 萨法汉 黄家维

时间：2025.06.12

一、 软件重构任务简述

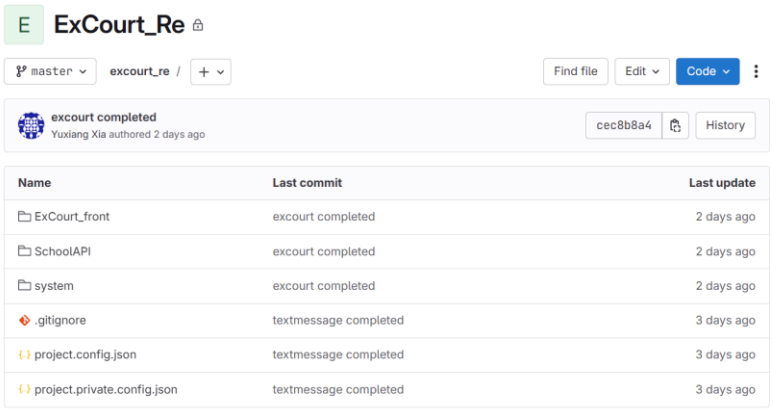
1. 重构动机

我们对于课程提供的几个选题都进行了代码阅读与分析，结合上学期末的软件交互汇报，对所有可重构项目有了具体的认识，其中有项目采用了 Go 语言等我们不熟悉的高级语言，有的项目本身比较完善，在框架上没有需要重构的关键点，而易场地项目采用我们熟悉的 python Flask + html 框架，使用微信开发者工具开发，是我们熟悉的软件开发架构，且通过初步分析，发现后端框架存在一些工程上不规范的问题，功能上也有一些实现逻辑有待改善，因此选择重构此项目，意在提升软件的性能、规范、用户体验。

2. 重构计划以及人员分工

重构计划：
1）代码初读和确认重构项目：全员参与，两周时间
2）易场地代码分析和重构计划确认：全员参与，两周时间
3）数据库架构更新：侯俊皓负责，一周时间
4）后端框架优化与实现算法改进：马萨、萨法汉负责，两周时间
5）前端适配新后端框架：夏宇翔、黄家维负责一周时间
6）前端优化与改进：夏宇翔、黄家维负责一周时间
7）重构完成后的整体测试：全员参与，一周时间

重构管理工具：gitlab　我们在 group 中创建了新的仓库，用来管理提交重构的代码。



人员分工：
侯俊皓　重构项目经理
夏宇翔　前端重构组长
马萨　　后端重构组长
萨法汉　参与后端重构
黄家维　参与前端重构

备注：其中侯俊皓、夏宇翔、马萨工作贡献较多

二、 现有软件系统分析

1. 软件系统架构分析

“ExCourt 易场地“是一个为学生设计的羽毛球场地共享预订与管理平台。系统包含微信小程序前端、基于 Flask 的 REST API 后端，以及 MySQL 数据库存储。
软件在功能方面支持学生注册账号、与他人交换场地预约、送场、组队、管理好友、聊天沟通，以及失物招领功能。

ExCourt 采用了经典的三层架构，其结构包括：
1）WeChat 小程序前端
功能：负责用户界面和交互，向用户展示信息，收集用户输入
2）Flask REST API 后端（各 Blueprint）
功能：处理业务逻辑，如用户认证、场地交换、好友管理等，封装并处理所有涉及业务规则的操作
3）MySQL 数据库
功能：负责数据的持久化存储，包括用户、场地、操作记录等表

架构上的优点包括：三层架构分工明确，职责清晰，可以各自独立部署，易于开发和维护。同时前端的代码管理结构清晰，用三个文件夹分别管理图片资源、工具资源和页面资源，文件的命名易理解，因此前端部分无需进行架构升级。

架构上也存在一些缺点：首先数据库采用 MySQL 数据库，其轻量级的特点易于实现，但是在复杂业务、数据分析、强一致性、高扩展性等应用场景中表现不佳，因此考虑到未来系统的扩展性和 SQL

数据兼容性，需要对数据库进行架构升级，采用 Postgresql 数据库。此外，原架构的后端方面存在一些 Blueprint 拆分、数据库连接池、应用配置不集中等工程上的问题，我们也会进行重构改进。

在前后端连接方面，部分功能算法存在问题，例如头像设置、图片发送、日期更新等，我们一一查找到了问题根源并进行改进。

代码味道分析
参考 https://www.refactoring.com/catalog/

a) 设计模式相关

原软件的设计模式整体上比较清晰，但仍存在一些问题：
1) 未使用工厂模式：原有 Flask 应用未采用 create_app()工厂函数，导致测试和部署环境不易切换，代码不具备良好模块化。
2 路由集中、无模块化：未使用 Flask Blueprint，所有路由集中在一个文件或函数，造成“巨石”式结构，维护上有一定困难。

b) 类封装相关

原软件的类封装合理且直观，但是在数据库访问、业务逻辑与 HTTP 处理上均为函数式实现，因此缺少数据和行为的封装，导致全局变量、函数交叉调用严重。
SQL 操作分散在各个函数中，未用类进行统一管理同样地，配置管理、状态管理等通用逻辑未封装为专门的类或对象。

c) 方法抽取相关

方法抽取上，原软件有一个重复代码与“大函数”问题：类似的业务逻辑，如学生查找、场地查询、数据校验等在多个路由中重复出现，缺少公共方法或工具函数抽取，可以对其尝试地做出改进。

d) 变量相关

软件中的变量命名有工程上的不便利问题，后端 API 地址、数据库连接参数、文件路径、HTTP 状态码、消息文本等在代码中多处硬编码，实际应当整合为一些全局定义，便于扩展和变化。

e) Bug

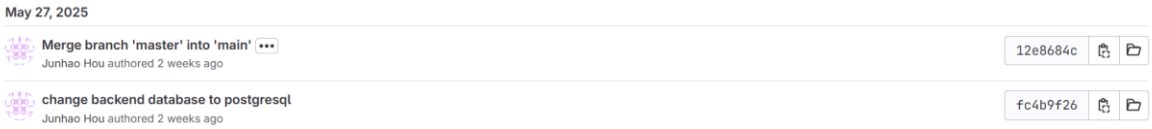
代码存在一些功能上的 bug：
1） 个人界面中的头像问题，不能设置头像且头像无法刷新

2） 聊天功能中的图片发送功能实际上并没有实现
3） 交换场地的数据库连接有一定问题，没有实际更改数据库内容
4） 聊天信息需要退出刷新才能看到，根据用户习惯应该是实时接受的
5） 最大人数的设置会失败

三、 对软件系统的改进

1. 对系统架构的改进

改进 1：升级数据库为 PostgreSQL
改进原因：数据库采用 MySQL 数据库，其轻量级的特点易于实现，但是在复杂业务、数据分析、强一致性、高扩展性等应用场景中表现不佳。
改进结果与好处：重新编写了 sql 文件，适配 PostgerSQL 的数据库格式，在后端代码中将所有 MySQL 方法更新为 PostgreSQL，使数据库的一致性、扩展性增强。



改进 2：应用工厂模式与 Blueprint 拆分（SCHOOL_API）
改进原因：将 Flask 项目初始化逻辑封装为 create_app() 工厂函数，并把路由按照功能分拆为 students 和 courts 两个 Blueprint，提升项目结构清晰度。
改进结果与好处：每个 Blueprint 可单独注册或取消，增强代码解耦，方便对不同功能模块做独立测试和维护，也符合 Flask 官方推荐的分层结构，后续如果要扩展新模块也很方便。

改进 3：基于环境的配置管理
改进原因：将数据库连接参数、服务器端口等敏感或环境相关配置迁移到 .env 环境变量文件中，通过 python-dotenv 加载到 Config 类中，避免直接写在代码里。
改进结果与好处：开发环境、测试环境、生产环境之间可以无缝切换配置，只需更换 .env 文件即可，保障重要信息如密码等不会被意外提交到代码仓库，提升安全性。

改进 4：数据库连接池与上下文管理器
改进原因：用 psycopg2 的连接池（psycopg2.pool）统一管理数据库连接，配合 Python 上下文管理器（with 语法）自动获取和释放游标，避免每个请求都新建和销毁数据库连接造成性能浪费。
改进结果与好处：高并发场景下能有效利用已有数据库连接，提升接口响应速度，自动回收资源，减少因连接泄漏导致的“too many connections”错误。

改进 5：集中化应用配置（SYSTEM 服务）
改进原因：将 PF_PATH、QR_PATH、SWAGGER_URL、API_URL 等路径或 URL 配置集中写入

config.py，通过 current_app.config 动态读取，避免重复配置和硬编码。
改进结果与好处：将来需要调整文件路径、API 地址、开关某些功能时只需在配置文件里修改，无需在多个源码文件中查找替换，配置管理更高效也更不易出错。

2. 对设计模式的改进

改进 1：抽取数据访问层（DAL）
改进原因：将所有 SQL 查询与数据操作集中到 repository.py 文件，定义如 get_all_students()、get_court_by_id() 等函数，避免 SQL 语句分散在各个路由处理函数内。
改进结果与好处：数据访问逻辑和 HTTP 处理逻辑彻底分离，便于针对数据访问单元测试，其他模块也能复用这些数据访问函数，使路由代码更简洁易维护。

改进 2：标准化数据库游标
改进原因：统一使用 psycopg2.extras.DictCursor，允许通过 row['Court_id'] 这种名字访问数据库返回的数据，而不是通过 row[0] 这样的数字下标。
改进结果与好处：代码更具可读性和自解释性，后续如果数据库表结构有变，只需调整字段名即可，减少 bug 发生概率。

改进 3：移除未使用路由
改进原因：删除所有客户端都不再请求的历史或废弃接口，减少无用代码。
改进结果与好处：代码库更精简，维护成本更低，开发者无需为无效接口写文档或做安全加固，后期出错风险也更小。

3. 对关键算法的改进

改进 1：图片保存接口升级
改进原因：需要将聊天模块将图片保存路径集中配置。
改进结果与好处：/sendphoto 接口优化为先保存文件再更新数据库，并保证返回的 pic_url 一致，还增加了表单校验和更规范的 Socket.IO 通信。

改进 2：用户头像设置功能完善
改进原因：新建用户头像为空白，后端相关图片 URL 为空，但更改个人信息要求图片 URL 非空，必须上传头像才能更改，且上传头像后无法立即刷新。
改进结果与好处：在后端取消了对于图片 URL 的强制需求，实现头像与其他文本个人信息分离更改。上传头像后增加刷新页面，实现立即刷新头像。

该进 3：实现聊天信息发送实时性

改进原因：原软件聊天发送无法实时接收

改进结果与好处：直接修改页面更新逻辑，将原本重新进入页面的数据更新改为发送或接受后更新页面数据，实现实时收发信息。

改进 4：实现聊天发送图片功能

改进原因：软件设计中有发送图片功能，但代码中似乎没有实现

改进结果与好处：增加实现了图片实时发送，并能在历史聊天记录中显示。符合常见的聊天场景功能点。

改进 5：日期实时更新

改进原因：原软件更换日期时信息不会改变。

改进结果与好处：更改日期后刷新筛选内容，具体方法如下显示：

```
changeNav(event){
  const newIndex = event.currentTarget.dataset.index;
  //console.log(newIndex);
  this.setData({curDay:newIndex});
  // 重新高亮日期
  const days = this.data.days.map((day, i) => ({
      ...day,
    active: i === newIndex
  }));
  this.setData({ days });
  // 刷新筛选内容
  this.updateFilteredCourtSlots(this.data.selectedStatus);
},
```

改进 6：选择最大人数实现逻辑改进

关键原因：原代码用了赋值，而没有用 setdata，选择最大人数无效。

改进结果与好处：改用 setdata，同步更新页面显示。具体方法如图所示：

```
// 选择最多希望加入人数事件
bindMaxParticipantsChange: function (e) {
  const value = this.data.maxParticipantsArray[e.detail.value];
  this.setData({
      Max_num: value,
      selectedMaxParticipants: value   // 同步更新页面显示
  });
  console.log(this.data.Max_num);
},
```

改进 7：三元判断的处理后置

改进原因：前端代码中有大量的三元判断

改进结果与好处：将这些判断放在后端预处理，前端直接输出

```
const timeSlots = [
  '9:00 - 10:00', '10:00 - 11:00', '11:00 - 12:00', '12:00 - 13:00',
  '13:00 - 14:00', '14:00 - 15:00', '15:00 - 16:00', '16:00 - 17:00',
  '17:00 - 18:00', '18:00 - 19:00', '19:00 - 20:00', '20:00 - 21:00', '21:00 - 22:00'
];

const records = res.data.data.map(item => {
  const courtid_split = item.court_id.split('-');
  const timeIndex = Number(courtid_split[4]);
  return {
    courtid_split,
    timeSlot: timeSlots[timeIndex] || '未知时段',
    source: item.source,
    status: item.status
  };
});
```

改进 8：实现交换和赠送场地的数据库连接

改进原因：在交换和赠送场地后，不会在数据库中修改，导致在日程表中仍然显示之前的场地

改进结果与好处：在确认交换或赠送场地后，对 schoolapi 发起请求，更新数据库，实现完整的交换流程。再打开场地信息可以看到更新。

4. 对代码与规范的改进

改进 1：新增 .gitignore 文件

改进原因：为了防止将如 .env 环境变量文件、venv/ 虚拟环境文件夹、PyCharm 或 VSCode 的 .idea/、.vscode/ 配置目录，以及 Python 构建产物等开发过程中自动生成的非项目源码文件意外加入 git 仓库。

改进结果与好处：这样可以让代码仓库只保留有用的业务代码和配置，保证不会把包含敏感信息（如数据库账号密码等）的文件上传到远程仓库，同时也让 PR 更干净，便于代码审查。

改进 2：新增 requirements.txt（SYSTEM & SCHOOL_API）

改进原因：将每个服务（如后端 SYSTEM、SCHOOL_API）的所有第三方依赖库例如 Flask、psycopg2、python-dotenv 等及其具体版本记录下来，便于团队协作和环境一致性。

改进结果与好处：新开发者只需运行 pip install -r requirements.txt 即可一键安装全部依赖，确保开发、测试、生产环境一致，也方便对依赖进行安全性和合规性检查。

改进 3：后端 URL 全局化

改进原因：原来项目中每次请求后端 API 都直接写死了 http://123.60.86.239:8000，分布在大约 60 处代码，维护困难且易出错。

改进结果与好处：现在把后端基础地址提取到 config.js 里，所有请求都用同一个常量。将来如果后端服务器地址变更，只需修改一处配置即可，避免遗漏和接口不一致的问题。

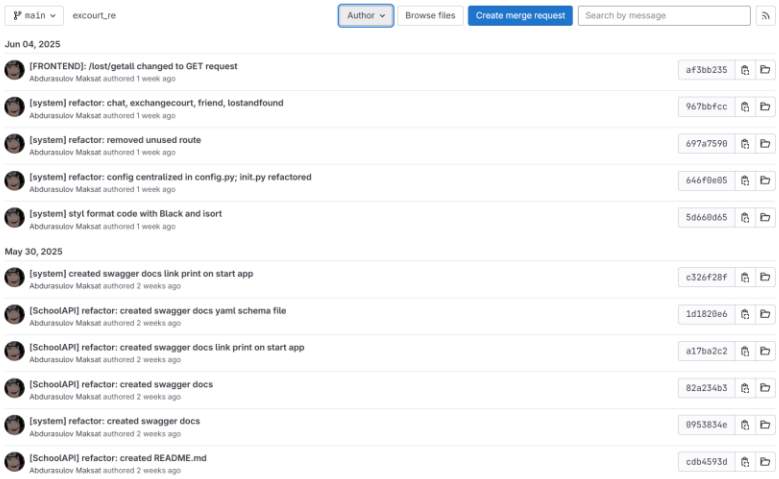
改进 4：引入 Marshmallow 请求校验
改进原因：为 POST 接口增加 StudentSchema、VerifySchema，强制校验请求体 JSON 的字段类型、格式和必填项，防止脏数据直接写入数据库。
改进结果与好处：如果输入数据不合规，接口会立即返回 400 错误，并带有清晰的错误提示，开发者和前端能快速定位问题，API 合约更明确，后端代码更健壮。

改进 5：SchoolAPI 代码自动格式化（Black & isort）
改进原因：统一使用 Black 工具自动整理代码风格，isort 自动规范 import 顺序，减少因代码风格不同引起的团队沟通成本。
改进结果与好处：所有代码风格一致，代码 diff 更清晰，代码审查时不再纠结于格式问题，整体代码可读性和团队协作效率提升。

改进 6：SYSTEM 代码自动格式化（Black & isort）
改进原因：对 SYSTEM 服务代码同样采用统一的自动格式化工具，保证风格一致。
改进结果与好处：团队成员在不同服务间切换时不会遇到风格差异，代码评审和维护更容易。

改进 7：API 消息语言一致性（统一为英文）
改进原因：所有接口返回的消息内容统一为英文，去除中英混杂，便于前端统一处理并面向国际用户。
改进结果与好处：API 响应风格专业统一，客户端解析更直接，减少因语言不一致导致的解析和显示问题。

改进 15：接口修正与优化
改进原因：为更符合 HTTP 语义标准 ExchangeCourt 模块将重复上传时的错误码从 402 改为 409，。LostAndFound 模块将 /lost/getall 从 POST 改为 GET，遵循 REST 规范，数据查询不带请求体。
改进结果与好处：各模块 API 更加健壮、易用，调用逻辑和 HTTP 标准保持一致，方便前端和第三方集成。



四、 重构后软件系统测试报告

（报告中展示部分重要的测试说明，具体请详见软件测试报告）

经过软件重构后的易场地小程序项目，重构后的小程序基于 python flask 后端和 postgreSQL 数据库，前端开发环境为微信开发者工具，小程序意在为校园羽毛球场地的送场、交换等操作提供便利，并支持聊天等其他功能。

测试机型：需要 windows 系统 PC，安卓系统手机，对硬件配置没有要求，市面上常见的电脑和智能手机都可。

用途及特殊说明：PC 端主要用于后端 API 测试，包括压力测试等，同时可以进行模拟手机端的小程序功能测试，手机端主要用于功能测试。

我们从功能角度出发进行了测试用例的设计，涵盖了界面显示、注册、登录、聊天、场地、我的、失物招领这几大板块，测试用例参考筛选自原软件测试用例，删除了极少数说明模糊、原软件没有的功能测试样例。

功能点/issue	测试用例 id	测试用例	优先级
可视化页面	1	可以查看我的场地和场地状态	最高
可视化页面	2	可以查看可交换场地	最高
可视化页面	3	可以查看可组队场地	最高
可视化页面	4	可以查看可接受的送场	最高

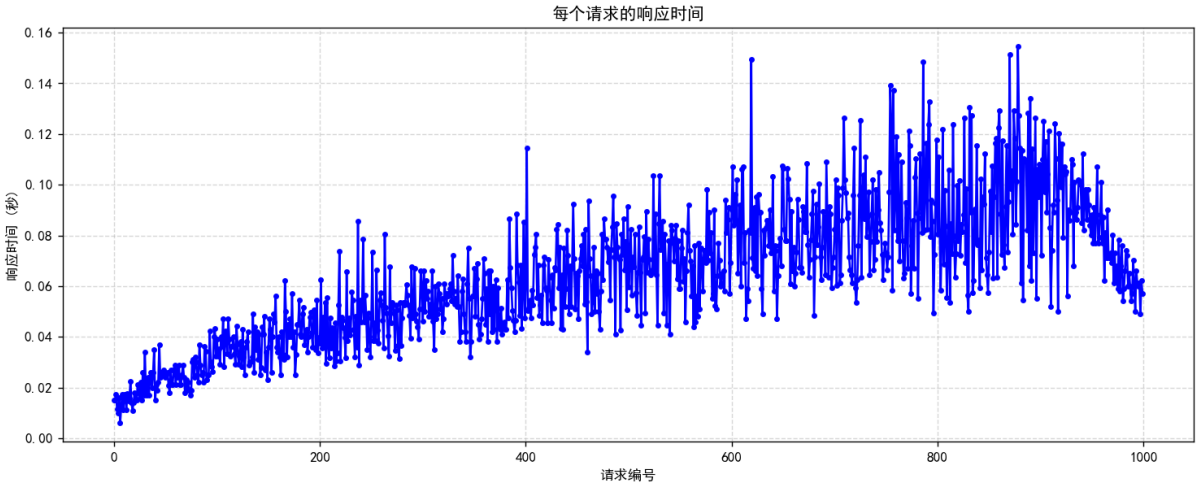
发布可交换场地	5	用户发布自己希望交换的场地	高
场地状态更新：待交换	6	发布交换场地后，场地状态自动更新为“待交换”	高
场地状态更新： 删除待交换标签	7	场地交换成功后，系统会删除该场地的“待交换” 标签	中
场地提供者发布可组队场地	8	场地提供者发布自己希望组队的场地	高
场地需求者点击可组队场地申请组队	9	场地需求者申请希望组队的场地	高
场地提供者处理拼场信息	10	场地提供者审核并处理拼场申请	高
场地需求者拿到相关场地信息	11	场地需求者拿到相关场地信息	高
场地提供者发布可送出的场地	12	用户发布自己希望送出的场地	高
场地需求者点击获得场地	13	场地需求者获得场地，并拿到相关信息	高
发布失物信息	14	用户发布失物信息，帮助他人找回遗失物品	高
查看失物信息	15	用户查询失物信息，寻找自己丢失的物品	高
认领	16	认领者可以认领失物	高
错领申诉	17	丢失者若发现失物被别人领走，可以发起申诉	高
查找用户	18	用户可以通过输入用户名或用户 ID 来查找系统中其他用户， 并加好友	高
发送即时消息	19	用户之间进行即时消息通信	中
查看聊天记录	20	用户可以查看与另一用户的历史聊天记录	中
用户注册	21	新用户注册账号	极高
用户登录	22	用户使用账号和密码登录	极高

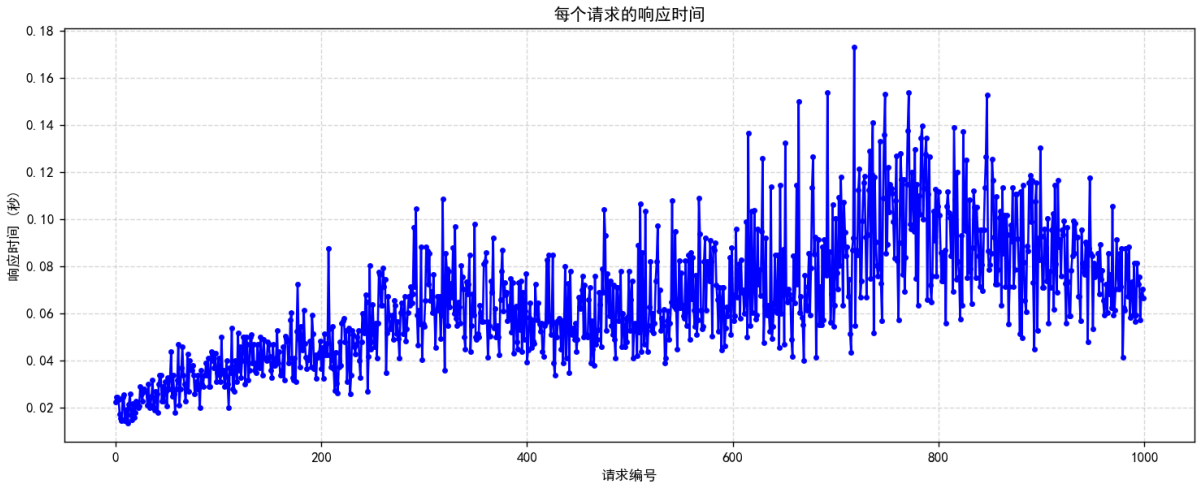
退出登录	23	已登录用户可以从系统安全退出	极高
修改用户信息	24	用户修改个人信息	中
管理员信誉分管理	25	系统管理员可以对用户的信誉分进行查看、 修改和管理	中
管理员用户信息管理	26	系统管理员可以查看、修改、删除用户信息	中

上述测试样例列表满足需求覆盖要求和测试覆盖要求，并且所有的测试样例都成功通过了测试，表明软件不存在明显缺陷，重构实现较好。

我们还对系统的资源占用，响应时间、并发性能等进行了测试。
我们模拟了 1000 个用户发送 API 请求来进行压力测试，最大并发也设置为 1000 进行并发测试，结果显示系统对于高压力下并发请求的相应良好，都能做到在极短时间内成功相应。
同时系统的资源占用较少，不会出现高内存和 cpu 占用等情况。

下面是登录和获取场地信息接口的测试结果:






```
(excourt) C:\Users\22895\Desktop\pytest>python test.py
总请求数: 1000
成功数: 1000
失败数: 0
平均响应时间: 0.064 秒
最大响应时间: 0.155 秒
最小响应时间: 0.006 秒

(excourt) C:\Users\22895\Desktop\pytest>python test.py
总请求数: 1000
成功数: 1000
失败数: 0
平均响应时间: 0.066 秒
最大响应时间: 0.173 秒
最小响应时间: 0.014 秒
```

五、 参考文献

Fowler, M. (1999), *Refactoring: Improving the Design of Existing Code* , Addison-Wesley , Boston, MA, USA .