

# Browser Use 功能介绍



https://github.com/browser-use/browser-use 这个有什么功能

你提供的仓库地址对应的项目是 Browser Use(也叫 browser-use),它的功能和作用如下:

### 🚀 项目功能概览

### 1. AI 浏览器代理(AI Browser Agent)

Browser Use 是一个开源 Python 库,让 AI 模型(如 OpenAI、Anthropic、DeepSeek 等)能够控制真实浏览器,通过 Playwright 自动打开网页、点击操作、填写表单、抓取数据等 —— 实现"让 AI 代替你动手"在浏览器里执行任务 GitHub+11 。

#### 2. 无需编程的自动化方案

除了 Python 脚本接口外,还提供 Web UI 与 CLI:

- Web UI(基于 Gradio)可让你通过聊天式交互命令驱动 agent 自动完成任务,无需写代码 GitHub。
- CLI 模式像在终端运行一个智能助手,指令控制 Web 浏览行为 GitHubdocs.browser-use.com。

#### 3. 多用例支持与示例丰富

常见应用场景包括:

- 从多个网站抓取数据(例如航班信息、招聘平台)
- 自动填写网页表单(投简历、订购流程)
- 模拟人工操作的窗口交互(Google Docs 编辑、采用多 Tab 浏览)
   文档提供了多个真实的 demo,例如让 AI 完成从抓取数据到写信再保存 PDF 的流程 browser-use.com
   docs.browser-use.com+8
   GitHub+8

#### 4. 模型和扩展的灵活支持

支持多款 LLM 模型(OpenAI、Anthropic、DeepSeek、Grok 等),并可定制插件、函数、Agent 设置等,还可以通过 **Model Context Protocol (MCP)** 与其他工具或 Claude Desktop 集成

GitHub+4

### **双** 技术结构与使用方式

- **安装与启动**:需要 Python 3.11+,通过 **pip install browser-use** 并安装 playwright,然后通过简单几行代码创建一个 **Agent(task=..., llm=...)** 执行任务 docs.browser-use.com+4 。
- **集成本地浏览器**:支持连接已有 Chrome/Firefox 用户数据目录,实现保持登录状态、录制屏幕等功能 GitHub。

# ₩ 社区反馈与优缺点

• Reddit 用户指出:

"Browser use on the other hand can be used to create autonomous agents to do pretty much anything inside a browser."

表示其可以用自然语言指令让 AI 代理跨多页完成复杂任务,这在普通浏览器插件或简单工具中不常见 Reddit+5。。

学术界也对这类工具保持审慎态度。一篇 2025 年文章指出,Browser Use 等工具虽然强大,但若不加限制,也可能被恶意内容诱导执行潜在危险操作(如远程代码执行、数据泄露等),需要结合执行安全机制予以防护 arxiv.org。

# ☑ 总结

Browser Use 是一个让大语言模型可以 控制浏览器、自动执行交互性操作的工具库。它适合:

- 需要 AI 自动执行网页任务的用户
- 想要构建自动化流程(如爬虫、自动填写、跨站操作)
- 不想或不擅长写代码,但希望通过聊天式界面让 AI 完成操作

不过也需要注意安全风险管理,以避免 agent 被恶意内容误引导执行敏感动作。

如果你想了解如何开始安装、配置具体模型,或构建一个实际任务流程的脚本示例,随时可以告诉我,我可以帮你写示例代码或演示流程。



所以,这是一个按脚本操作网页,而不是让AI智能操作网页的项目是吗

你问得非常关键,区别如下:

# ☑ Browser Use 是让 AI "智能操作网页",而不是单纯按脚本操作网页的工具。

### ● 不是脚本式工具像 Puppeteer/Playwright 般"死操作"

● 普通脚本(如 Puppeteer、Selenium)是你 **明确告诉浏览器该点击哪里、输入什么**;

• 而 Browser Use 是你只需用一句自然语言告诉它你的意图,例如:

"请登录 LinkedIn 并搜索 Michael Banana,然后把他的资料导出成 PDF。"

#### 🔁 它背后的 Agent 和 LLM 模型会自动:

- 1. 分析页面内容;
- 2. 理解任务意图;
- 3. 自主判断点击/输入/抓取路径;
- 4. 执行动作(通过调用 Playwright 或其他底层执行引擎)。

### 🔖 实质上,它是一个"LLM + 浏览器控制"的 Agent 框架

#### 内部原理是:

- 把页面结构用 Playwright 抓下来(转换成 DOM 的简化结构);
- 输入到 LLM 模型中(如 OpenAI GPT-4、Claude 3、DeepSeek 等);
- 模型返回"下一个操作"计划(点击哪个按钮、输入什么文字等);
- 由控制器执行这些动作;
- 重复反馈这个循环,直到任务完成。

### 

- **自主规划操作路径**(非你写死的 click("#submit"));
- **上下文感知**(知道你正在填写表单或在搜索某个产品);
- 语言层级控制(你用一句话指令,它完成复杂网页任务)。

# 🖈 总结一句话

Browser Use 是 "LLM + 浏览器控制器"的结合体,属于 AI 智能网页代理(AI browser agent)范畴,不是单纯的"网页自动化脚本库"。

#### 如果你想比较一下:

工具	操作方式	是否 AI 理解任务	是否需要自己编脚本
Selenium/Puppeteer	你写死点击/输入动作	🗙 不理解,只执行	✓ 需要
Browser Use	你用语言告诉它任务	✓ 理解网页与意图	ズ 不一定需要(可用 UI)

你如果想我帮你写一个真实的 Browser Use 使用示例(如:用 OpenAI 控制浏览器,自动查找招聘网站上的职位并导出),我可以马上提供代码。你也可以告诉我你的使用场景,我可以演示。



#### 它能在背景操作吗?还是会跟我抢滑鼠

这个问题问得非常实用——简单回答如下:

# 🔀 默认不会在"纯后台"运行,会和你抢鼠标和焦点

Browser Use 是基于 **Playwright 启动真实浏览器(如 Chrome)**来控制网页的,而不是像传统爬虫那样在无头(headless)模式或 API 层面处理。

#### 所以:

- 🗸 它确实可以自动执行网页任务;
- 🗙 它默认运行的是一个可见的浏览器窗口;
- X 你无法同时手动操作那个浏览器窗口,否则会"打断"或"干扰" agent 的判断;
- X 无法与其他前台任务并行无干扰操作,因为 agent 可能会点击、输入、滚动,会干扰你的使用。

### ☑ 但它也支持"无头模式"运行(不会抢你鼠标),方法如下:

你可以通过参数设置让 Browser Use 在无头(Headless)模式下运行 Playwright,这样:

- 浏览器窗口不会弹出;
- Agent 会在后台悄悄执行任务;
- 不会抢你鼠标或打断你的操作。

#### 举例代码:

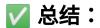
```
from browser_use import Agent
from browser_use.llms import OpenAI

agent = Agent(
    task="Search 'ChatGPT' on Google and get the first result.",
    llm=OpenAI(),
    headless=True # ② 设置为无头模式!
)

agent.run()
```

### ◎ 额外建议:

- 如果你在开发桌面端或 GUI 软件,可以用无头模式跑多个 Agent 实例,并将结果通过 socket 或 API 提供 到前端展示;
- 如果你在本地运行多个自动化脚本,强烈建议默认用 headless=True,除非你要调试 UI。



模式	会不会弹窗	会不会抢鼠标/焦点	适合谁
默认模式	✓会	✓会	手动调试、可视操作
Headless 模式	✗不会	✗不会	后台运行、自动化批处理

需要我给你写一个完整的 headless 运行范例或运行多个 agent 的脚本也可以。



我能用我自已的Gemini API KEY吗

### ☑ 是的,你可以用你自己的 Gemini API Key!

Browser Use 官方明确支持自定义使用 Gemini(Google)的 API Key,用它来作为 Agent 背后的 LLM。

# 🖋 如何使用你自己的 Gemini API Key

你只需要安装 google-generativeai 包,并在创建 LLM 时传入你的 API Key,例如:

#### 示例代码:

```
from browser_use import Agent
from browser_use.llms import Gemini

agent = Agent(
    task="帮我搜索 ChatGPT 的优缺点并整理成 bullet points",
    llm=Gemini(api_key="你的_API_KEY"),
    headless=True
)

agent.run()
```

### 🤪 安装依赖:

如果你还没装 google-generativeai,先运行:

```
bash
pip install google-generativeai
```

# ☑ 支持模型列表(Gemini)

目前 Gemini() 类默认使用:



- gemini-pro(文字处理模型)
- 若使用 gemini-pro-vision(支持图像输入),需要你明确传入参数

# 🢡 如果你用 .env 管理 key:

可以这样设置环境变量,自动读取:

env

GOOGLE\_API\_KEY=你的\_API\_KEY

然后代码里可以不用传 key,Gemini 会自动从环境变量中读取。

### ▶ 注意事项:

项目	说明
频率限制	免费 Gemini API 有调用速率限制(如每分钟/每天)
权限范围	确保你的 key 有访问 Gemini Pro 的权限
输入输出长度限制	Gemini 对输入 token 和输出字符数也有上限,注意复杂任务拆分
不支持网页搜索	默认 Gemini 没有联网能力,不能实时获取网页内容(Agent 会靠浏览器获取)

# ✔ 总结

问题	答案
能不能用自己 Gemini Key?	✓ 可以
如何使用?	Gemini(api_key="你的key")
支持哪些模型?	gemini-pro / gemini-pro-vision
有联网吗?	🗙 没有(网页由浏览器操作获取)

#### 如果你要我提供:

- ☑ 示例任务(如让 Gemini 控制浏览器登录网页或抓数据);
- **▽** 多模型比较(Gemini vs GPT vs Claude in Browser Use);

都可以告诉我,我会帮你一步步搭好。



示例任务(让 Gemini 控制浏览器登录网页google cloud console 创建新项目,并提取项目名称/编号/ID)

下面是一个完整示例,展示如何使用你自己的 Gemini API Key 让 Browser Use 代理自动:

 $\boxed{\hspace{0.5cm}}$  打开 Google Cloud Console → 登录 → 创建新项目 → 获取新项目的名称、编号、ID,并输出。

#### ↑ 前提条件(请先确认):

- 1. 你已经有一个可用的 Google Cloud 账号;
- 2. 你在本机 Chrome 浏览器中已经登录并启用了 Cloud Console;
- 3. 你愿意让 Browser Use 使用你的用户目录(复用登录状态);
- 4. 你有自己的 Gemini API Key。

### 🧠 示例脚本(自动创建新项目并提取信息)

```
python
from browser_use import Agent
from browser_use.llms import Gemini
# 替换为你的 Gemini API Key
GEMINI_API_KEY = "你的API_KEY"
agent = Agent(
   task="""
       1. 打开 Google Cloud Console (https://console.cloud.google.com/)
       2. 确保登录状态(若未登录,使用当前浏览器用户目录自动登录)
       3. 创建一个新的项目,名称为"test-agent-project"或任何默认值
       4. 创建成功后,提取并输出该项目的:
          - 项目名称 (Project Name)
          - 项目编号 (Project Number)
          - 项目 ID (Project ID)
   llm=Gemini(api_key=GEMINI_API_KEY),
   headless=False, # 可改为 True,如果不需要看到操作过程
   user_data_dir="C:/Users/你的用户名/AppData/Local/Google/Chrome/User Data", # 👈 复用已登录的
Chrome 状态
   browser="chrome"
agent.run()
```

### 通 说明解析:

参数	用途	
task	告诉 Agent 需要"登录 → 创建项目 → 抓取信息"	
llm	使用 Gemini API,智能理解网页结构和流程	
headless=False	显示浏览器窗口(方便调试)	
user_data_dir	指定你本地 Chrome 浏览器的用户数据目录,这样 AI 就可以复用你的登录状态,无需手动输入密码	
browser	指定使用 Chrome 浏览器,避免用 Firefox 等默认值无法正常登录 Google	

# 🥯 安装依赖(如果还没装):



bash

pip install browser-use google-generativeai playwright playwright install # 安装浏览器内核

### ★ 注意事项:

- 登录问题:如果你没设定 user\_data\_dir,AI 会卡在登录页(它目前不会自动填 Google 帐号密码);
- GCP 项目数量限制:默认每个帐号每天最多创建几个项目;
- 任务需要耐心: AI 操作网页需要几秒到几十秒不等;
- 模型稳定性建议:Gemini 会比 GPT-4 稍不稳定在复杂网页任务中,若遇失败可以改用 GPT-4。

### ◆ 扩展功能建议(可加到 task):

- "启用 Billing 功能";
- "打开 IAM 页面,添加成员";
- "创建服务账号,并下载 JSON 凭证"。

如果你愿意,我也可以帮你把这段脚本封装成 CLI 或 Gradio UI 界面,让你像聊天一样控制。要试试吗?



输出呢,提取的信息保存哪?

好问题!默认情况下,browser\_use 的 Agent 不会自动将你提取的信息保存到文件或变量中——它会"说"出结果(即输出到终端或日志中),但**你需要自己在任务中明确告诉它如何输出或保存信息**。

### ☑ 如何让 AI 输出提取结果(项目名称、编号、ID)

在 task 中加入明确指令,例如:

text

请在任务最后以如下格式输出结果:

项目名称: <名称> 项目编号: <编号> 项目ID: <ID>

#### 完整任务变成这样 👇:

```
python

task = """

1. 打开 Google Cloud Console (https://console.cloud.google.com/)
```

# 🦰 如何保存提取信息到变量或文件

目前 browser-use 的 Agent 结构是运行过程自动输出,但你可以用以下方式增强:

#### 方法 1:解析输出字符串

```
python

output = agent.run()
print("Agent 原始输出:", output)

# 提取结果 (简单文本提取逻辑)
import re

project_name = re.search(r"项目名称[::]\s*(.+)", output)
project_number = re.search(r"项目编号[::]\s*(\d+)", output)
project_id = re.search(r"项目ID[::]\s*([a-z0-9\-]+)", output)

print("☑ 提取结果:")
print("名称:", project_name.group(1) if project_name else "未提取")
print("编号:", project_number.group(1) if project_number else "未提取")
print("ID:", project_id.group(1) if project_id else "未提取")
```

### 方法 2:让 Agent 输出成 JSON 格式(推荐)

改写指令,加一句:

```
text

请用以下 JSON 格式输出:

{

    "项目名称": "...",

    "项目编号": "...",

    "项目ID": "..."

}
```

这样你可以直接用 json.loads(output) 解析它。

# 写入文件(例如保存为 project\_info.json)

```
import json

info = {
    "project_name": project_name.group(1),
    "project_number": project_number.group(1),
    "project_id": project_id.group(1)
}

with open("project_info.json", "w", encoding="utf-8") as f:
    json.dump(info, f, ensure_ascii=False, indent=2)
```

# ☑ 总结

你想要做什么	方法
控制输出格式	在 task 中加上明确格式要求(文本或 JSON)
提取信息存变量	output = agent.run() 后用正则或 JSON 解析
保存提取信息到文件	json.dump() 保存为.json/.txt/.csv等
自动化处理多个项目任务输出	可封装为循环调用脚本,每次运行生成不同 project_info 文件

需要我直接给你一个能自动保存为 JSON 的完整脚本吗?还是你想把它改成批量任务(比如一键创建多个项目)?