指定端口或者地址

```
# 接管9333端口的浏览器，如该端口空闲，启动一个浏览器
browser = Chromium(9333)
browser = Chromium('127.0.0.1:9333')
```

手动打开的浏览器

如果需要手动打开浏览器再接管，可以这样做：

1. 右键点击浏览器图标，选择属性
2. 在“目标”路径后面加上`--remote-debugging-port=端口号`（注意最前面有个空格）
3. 点击确定
4. 在程序中的浏览器配置中指定接管该端口浏览器

文件快捷方式的目标路径设置：

```
"D:\chrome.exe" --remote-debugging-port=9333
```

然后就可以用上面的方法接管端口

bat 文件启动的浏览器

可以把上一种方式的目标路径设置写进 bat 文件（Windows系统），运行 bat 文件来启动浏览器，再用程序接管。

新建一个文本文件，在里面输入以下内容（路径改为自己电脑的）：

```
"D:\chrome.exe" --remote-debugging-port=9333
```

指定独立端口和数据文件夹

每个要启动的浏览器使用一个独立的`ChromiumOptions`对象进行设置：

```
from DrissionPage import Chromium, ChromiumOptions

# 创建多个配置对象，每个指定不同的端口号和用户文件夹路径
co1 = ChromiumOptions().set_paths(local_port=9111, user_data_path=r'D:\data1')
co2 = ChromiumOptions().set_paths(local_port=9222, user_data_path=r'D:\data2')

# 创建多个页面对象
tab1 = Chromium(addr_or_opts=co1).latest_tab
tab2 = Chromium(addr_or_opts=co2).latest_tab

# 每个页面对象控制一个浏览器
tab1.get('http://DrissionPage.cn')
tab2.get('https://www.baidu.com')
```

`auto_port()`方法

这种方式创建的浏览器是全新不带任何数据的，并且运行数据会自动清除，浏览器无法复用。

`auto_port()`支持多线程，多进程使用时由小概率出现端口冲突。
多进程使用时，可用`scope`参数指定每个进程使用的端口范围，以免发生冲突。

ChromiumOptions进行设置需关闭已启动的系统浏览器，否则会连接失败。

```
from DrissionPage import Chromium, ChromiumOptions

co = ChromiumOptions().auto_port()

tab1 = Chromium(addr_or_opts=co).latest_tab
tab2 = Chromium(addr_or_opts=co).latest_tab

tab2.get('http://DrissionPage.cn')
tab1.get('https://www.baidu.com')
```

使用 ini 文件

```
from DrissionPage import ChromiumOptions

ChromiumOptions().use_system_user_path().save()
```

默认配置下，由 DrissionPage 创建的浏览器，用户文件夹在系统临时文件夹的`DrissionPage\userData`文件夹内，以端口命名。

假如用 DrissionPage 默认配置在 9222 端口创建一个浏览器，那么用户数据就存放在`C:\Users\用户名\AppData\Local\Temp\DrissionPage\userData\9222`路径。

这个用户文件夹不会主动清除，下次再使用 9222 端口时，会继续使用。

如果使用`auto_port()`，会存放在系统临时文件夹的`DrissionPage\autoPortData`文件夹内，以端口命名。

如`C:\Users\用户名\AppData\Local\Temp\DrissionPage\autoPortData\21489`。

这个用户文件夹是临时的，用完会被主动清除。

自定义位置

如果要指定用户文件夹存放位置，可用`ChromiumOptions`对象的`set_tmp_path()`方法。

也可以保持到 ini 文件，可省略每次设置。

```
from DrissionPage import ChromiumOptions

ChromiumOptions().set_tmp_path(r'D:\tmp').save()  # 保存到ini文件
```

启动设置

```
from DrissionPage import Chromium, ChromiumOptions

# 创建配置对象（默认从 ini 文件中读取配置）
co = ChromiumOptions()
# 设置不加载图片、静音
co.no_imgs(True).mute(True)
co.incognito()  # 匿名模式
co.headless()  # 无头模式
```

```
# 设置启动时最大化
co.set_argument('--start-maximized')
# 设置初始窗口大小
co.set_argument('--window-size', '800,600')
# 使用来宾模式打开浏览器
co.set_argument('--guest')
# 设置超时
co.set_timeouts(base=10)
# 设置重试和间隔
co.set_retry(3,1.5)
# 以该配置创建页面对象
#'normal'：阻塞进程，等待所有资源下载完成（默认）
#'eager'：DOM 就绪即停止加载
#'none'：网页连接成功即停止加载
co.set_load_mode()
#该方法用于设置浏览器代理。
#该设置在浏览器启动时一次性设置，设置后不能修改。且不支持带账号的代理。
#如果需要运行时修改代理，或使用带账号的代理，可以用插件自行实现。
co.set_proxy('http://localhost:1080')
#忽略证书错误访问
co.ignore_certificate_errors()
#无痕模式启动浏览器
co.incognito()
#禁用js
co.no_js()
#设置header头
co.set_user_agent(user_agent='Mozilla/5.0 (Macintos.....')
#设置超时
co.set_timeouts()
page = Chromium(addr_or_opts=co)
```

获取指定标签页

```
from DrissionPage import Chromium

browser = Chromium()
tab1 = browser.get_tab(1)  # 获取列表中第一个标签页的对象
tab2 = browser.get_tab('5399F4ADFE3A27503FFAA56390344EE5')  # 获取指定id的标签页对象
tab3 = browser.get_tab(url='DrissionPage.cn')  # 获取第一个url中带'DrissionPage.cn' 的标签页对象
tabs = browser.get_tabs(url='DrissionPage.cn')  # 获取所有url中带'DrissionPage.cn' 的标签页对象
```

新建标签页并获取对象

```
from DrissionPage import Chromium

browser = Chromium()
browser.new_tab(url='http://DrissionPage.cn')
```

获取点击后出现的标签页

```python
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.get('https://DrissionPage.cn')
ele = tab.ele('.wwads-cn wwads-horizontal').ele('tag:img')
if ele:
    tab2 = ele.click.for_new_tab()  # 点击并获取新tab对象
    tab2.set.activate()
    ele2 = tab2.ele('确认访问', timeout=5)
    if ele2:
        ele2.wait(.5).click()
else:
    print('支持开源作者，请关闭广告屏蔽功能，谢谢。')
```

多标签页协同

```python
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.get('https://gitee.com/explore/all')

links = tab.eles('t:h3')
for link in links[:-1]:
    # 点击链接并获取新标签页对象
    new_tab = link.click.for_new_tab()
    # 等待新标签页加载
    new_tab.wait.load_start()
    # 打印标签页标题
    print(new_tab.title)
    # 关闭新打开的标签页
    new_tab.close()
```

设置超时和重试

```python
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.get('http://DrissionPage.cn', retry=1, interval=1, timeout=1.5)
```

选择模式加载

```python
from DrissionPage import Chromium

tab = Chromium().latest_tab
#normal()：常规模式，会等待页面加载完毕，超时自动重试或停止，默认使用此模式
#eager()：加载完 DOM 或超时即停止加载，不加载页面资源
#none()：超时也不会自动停止，除非加载完成
tab.set.load_mode.eager()  # 设置为eager模式
tab.get('http://DrissionPage.cn')
```

`none`模式技巧

跟监听器配合，可在获取到需要的数据包时，主动停止加载。

```
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.set.load_mode.none()  # 设置加载模式为none

tab.listen.start('api/getkeydata')  # 指定监听目标并启动监听
tab.get('http://www.hao123.com/')  # 访问网站
packet = tab.listen.wait()  # 等待数据包
tab.stop_loading()  # 主动停止加载
print(packet.response.body)  # 打印数据包正文
```

跟元素查找配合，可在获取到某个指定元素时，主动停止加载。

```
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.set.load_mode.none()  # 设置加载模式为none

tab.get('http://www.hao123.com/')  # 访问网站
ele = tab.ele('中国日报')  # 查找text包含“中国日报”的元素
tab.stop_loading()  # 主动停止加载
print(ele.text)  # 打印元素text
```

可等待到页面到达某种状态时，主动停止加载。比如多级跳转的登录，可等待 title 变化到最终目标网址时停止。

```
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.set.load_mode.none()  # 设置加载模式为none

tab.get('http://www.hao123.com/')  # 访问网站
tab.wait.title_change('hao123')  # 等待title变化出现目标文本
tab.stop_loading()  # 主动停止加载
```

获取cookie

```
from DrissionPage import Chromium

tab = Chromium().latest_tab
tab.get('https://www.baidu.com')

for i in tab.cookies():
    print(i)

#cookies().as_str(): 'name1=value1; name2=value2'格式的字符串
#cookies().as_dict(): {name1: value1, name2: value2}格式的字典
#cookies().as_json(): json 格式的字符串
```