# Python + selenium 开发爬虫

Selenium Python 绑定提供了方便的 API 来访问 Selenium WebDrivers，如 Firefox、Ie、Chrome、Remote 等。当前支持的 Python 版本为 2.7、3.5 及更高版本。

- **开源和可移植** ——Selenium 是一个开源和可移植的 Web 测试框架。
- **工具和 DSL** 的结合——Selenium 是工具和 DSL（领域特定语言）的结合，以进行各种类型的测试。
- **更容易理解和实现** ——Selenium 命令按照不同的类进行分类，这使得它更容易理解和实现。
- **减少测试执行时间** ——Selenium 支持并行测试执行，从而减少执行并行测试所花费的时间。
- **所需资源** 更少——与 UFT、RFT 等竞争对手相比，Selenium 需要更少的资源。
- **支持多种操作系统** ——Android、iOS、Windows、Linux、Mac、Solaris。
- **支持多种浏览器** - Google Chrome、Mozilla Firefox、Internet Explorer、Edge、Opera、Safari 等。
- **并行测试执行** ——它还支持并行测试执行，从而减少时间并提高测试效率。

# 安装selenium

安装Python3.x：官方下载页.

## Linux & Mac

```
pip install selenium
```

## Windows

从开始菜单点击运行（或者`Windows+R`）输入`cmd`,然后执行下列命令安装:

```
C:\Python35\Scripts\pip.exe install selenium
```

现在你可以使用Python运行测试脚本了。 例如：如果你创建了一个selenium的基本示例并且保存在了`C:my_selenium_script.py`，你可以如下执行:

```
C:\Python35\python.exe C:\my_selenium_script.py
```

## 安装网络驱动程序

可以安装 Firefox、Chromium、PhantomJs（现已弃用）等。

- 要使用 Firefox，您可能需要安装 GeckoDriver 下载地址

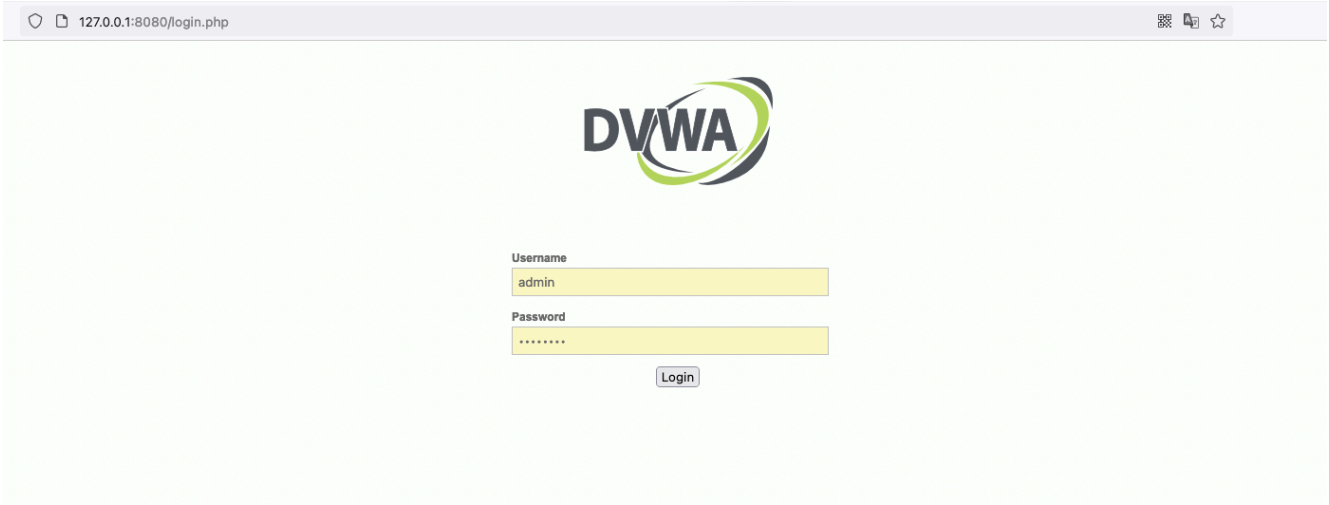
> 安装好之后，将可执行程序加入环境变量

# selenium 开发

先看一个Demo，通过Demo了解一下selenium的基本用法;

**效果：通过selenium自动登录DVWA靶机系统**

1. 首先开启DVWA `docker start dvwa`




2. 查看DVWA登录页面前端代码，我们的目标是实现自动登录，那么首先就需要获取到登录的输入框，通过查看页面源代码发现，输入框分别为两个input 标签，一个`name="username"`，另一个`name="password"`


3. 通过上一步确定了具体的标签后，我们需要有以下三步动作来实现自动登录；
    1. 通过程序定位标签；
    2. 输入账号密码；
    3. 点击登录；

通过以上逻辑，确定自动化登录代码

```
from selenium import webdriver
from selenium.webdriver.common.by import By

print("Start geektime Selenium Test Project....")
# 设置User-Agent
profile = webdriver.FirefoxOptions()
profile.set_preference("general.useragent.override",
                       "geektime.com")
# 使用火狐浏览器环境
driver = webdriver.Firefox(options=profile)
```

```
# 设置访问的地址
driver.get("http://localhost:8080")
# 使用XPATH方式获取html标签节点，此处为获取 input 标签中名称为 username 的节点
user_elem = driver.find_element(By.XPATH, "//input[@name='username']")
# 获取input标签，名称为password的节点
pass_elem = driver.find_element(By.XPATH, "//input[@name='password']")
# 清空节点默认值
user_elem.clear()
# 清空节点默认值
pass_elem.clear()
# 设置username标签节点内容为 admin
user_elem.send_keys("admin")
# 设置password标签节点内容为 password
pass_elem.send_keys("password")

# 获取登陆按钮节点
login_butten_elem = driver.find_element(By.XPATH, "//input[@Name='Login']")
# 点击登陆按钮
login_butten_elem.click()
# 打印User-Agent
print(driver.execute_script("return navigator.userAgent"))
# 关闭浏览器环境
driver.close()
```

使用cookie直接登录

```
import time

from selenium import webdriver

print("Start geektime Selenium Test Project By Use Cookie....")
# 使用火狐浏览器环境
driver = webdriver.Firefox()

# 设置访问的地址
# 添加Cookie
driver.get("http://127.0.0.1:8080")

driver.add_cookie({
    'name': 'PHPSESSID',
    'value': 'qq53p7m5ppq5t6a6gmbede2f27'
})
time.sleep(1)
driver.get("http://127.0.0.1:8080/index.php")


# 关闭浏览器环境
# driver.close()
```

## selenium 各方法含义

| 方法 | 描述 |
| --- | --- |
| add_cookie | 将 cookie 添加到当前会话。 |
| back | 在浏览器历史中倒退了一步。 |
| close | 关闭当前窗口。 |
| create_web_element | 创建具有指定 element_id 的 Web 元素。 |
| delete_all_cookies | 删除会话范围内的所有 cookie。 |
| delete_cookie | 删除具有给定名称的单个 cookie。 |
| get_cookie | 按名称获取单个 cookie。 如果找到则返回 cookie，如果没有则返回 None。 |
| get_cookies | 返回一组字典，对应于当前会话中可见的 cookie。 |
| quite | 退出驱动程序并关闭每个关联的窗口。 |
| refersh | 刷新当前页面。 |
| set_page_load_timeout | 设置在引发错误之前等待页面加载完成的时间量。 |
| set_script_timeout | 设置脚本在 execute_async_script 调用期间在引发错误之前应等待的时间量。 |
| current_url | 获取当前页面的 URL。 |
| page_source | 获取当前页面的源代码。 |
| title | 返回当前页面的标题。退出驱动程序并关闭每个关联的窗口。 |
| refersh | 刷新当前页面。 |
| set_page_load_timeout | 设置在引发错误之前等待页面加载完成的时间量。 |
| set_script_timeout | 设置脚本在 execute_async_script 调用期间在引发错误之前应等待的时间量。 |
| execute_script | 执行 js 脚本 |

## 爬取极客时间网站课程及链接

代理IP获取网站

`https://free.kuaidaili.com/free/inha/`

```
from selenium import webdriver
from selenium.webdriver.common.by import By
```