VMLogin 虚拟多登浏览器

软件使用说明书

Ver 1.0

扫一扫添加客服微信
Scan Add Customer
Service WeChat

系统要求

硬件要求

- RAM： 推荐 4GB;
- 1GB 的可用磁盘空间
- 有效的GPU

支持的操作系统

虽然VMLogin 支持x86（32 位）系统，但我们更推荐在x64（64 位）系统中使用VMLogin。

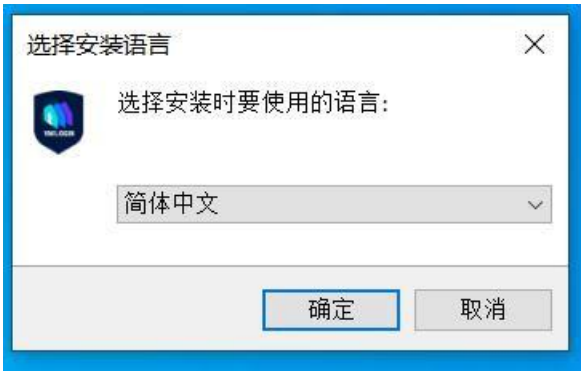
VMLogin 支持的 OS 有：

- Windows 10
- Windows Server 2016
- Windows 7
- Windows 8

安装软件

用户可以通过官方网址 vmlogin.com.cn 下载最新软件安装包，本次说明书使用的版本为 1.0.3.6 。

我们可以启动软件安装包程序，会显示选择安装语言界面：

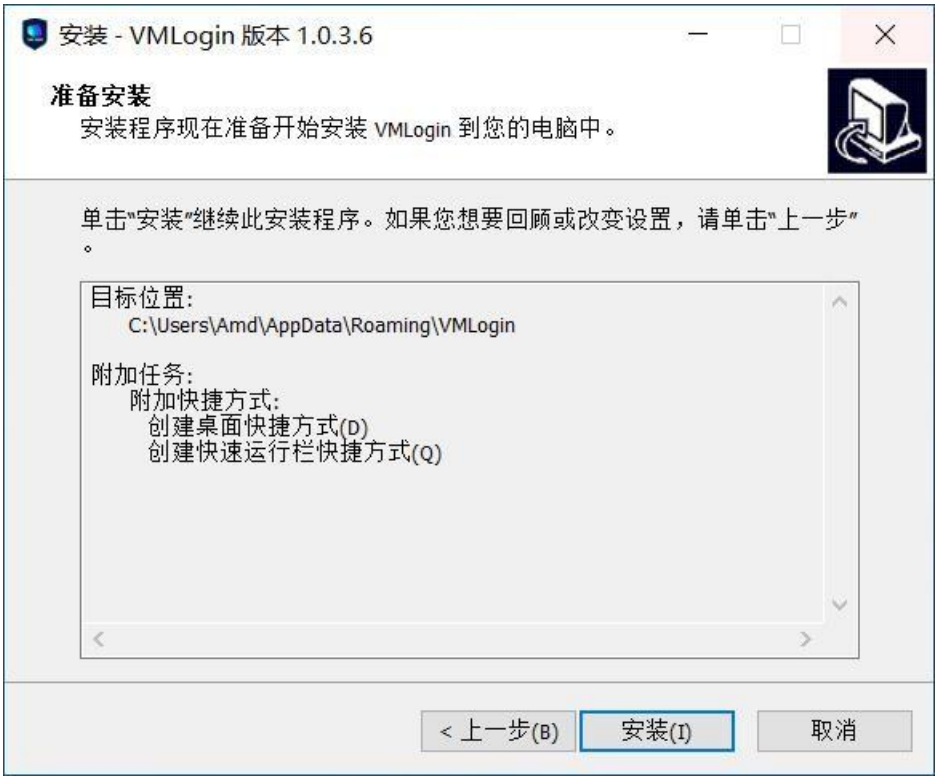


用户根据自己的语言喜好选择软件语言，也可以在软件安装后在设置里改变语言选项。目前支持简体中文和英文两种语言。

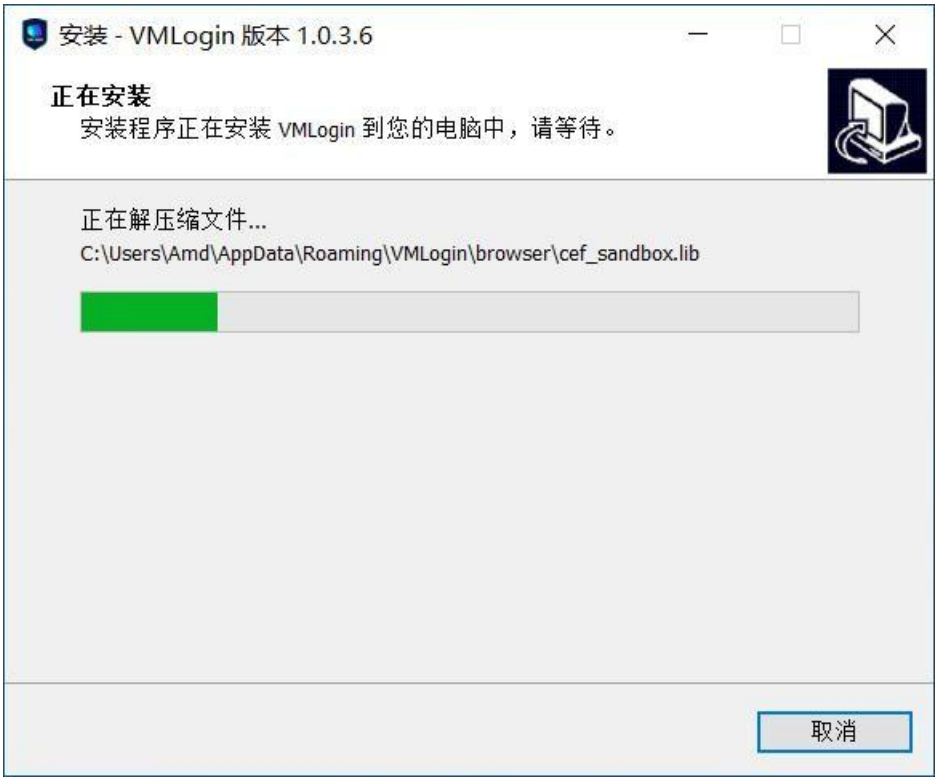
用户确定选择语言后，进入选择安装目录界面：

这里一般默认安装，按“下一步”按钮即可，如果用户自定义安装目录，请注意目录权限，要当前用户可写。

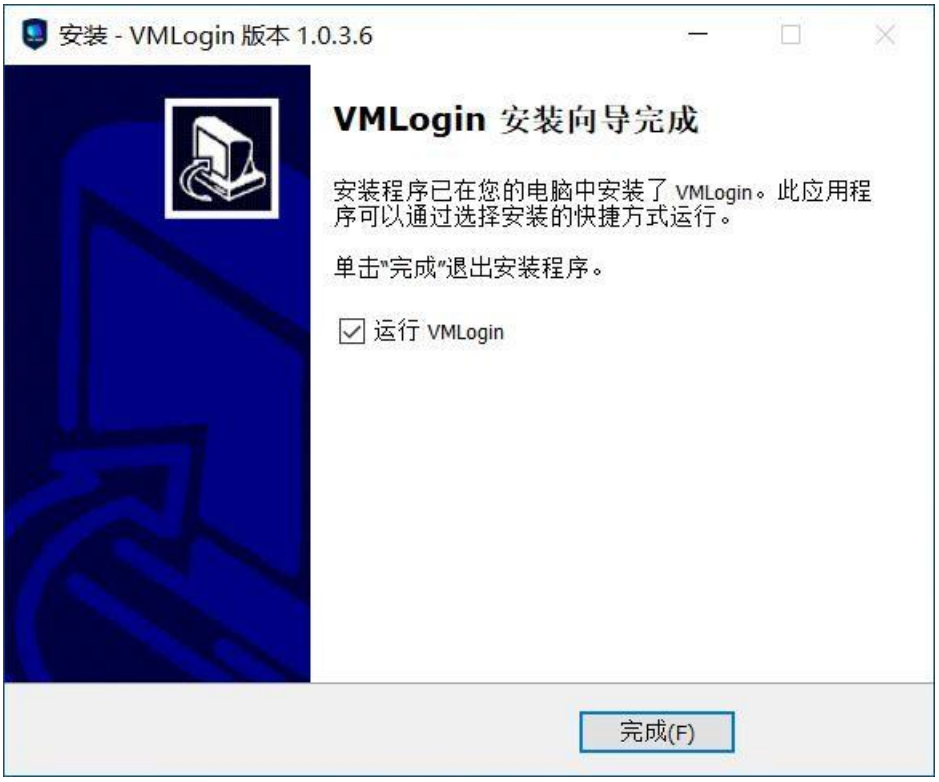
一般需要创建桌面快捷方式方便用户使用，这里我们默认点击“下一步”按钮继续。



此时我们点击“安装”按钮可以进行安装。



我们需要等待安装进度完成。



到这一步就软件安装就完成。

软件功能

1. 启动软件

我们启动软件会看到登陆界面，新用户需要创建新帐户登陆才能使用。

2. 注册帐户

新用户注册，需要使用 email 作为帐号，设置好密码，注册即可。

3. 软件功能

当我们登陆软件后，会看到软件的主界面。左侧功能栏可以切换进入软件相关功能界面。

A. 新建浏览器配置文件

新建浏览器配置文件-基本配置

显示名称：这里是设置配置文件的显示名称

备注信息：可以自定义一些配置文伯的备注信息

显示图标：可以选择一个图标方便识别

操作系统：这里设置浏览器所运行的系统类型

设置代理服务器

这里支持的代理服务器类型有：http、https、socks4、socks5、ipv6，并且支持用户验证。

WebRTC

WebRTC 是一种浏览器插件，通常被需要快速直接连接的网络应用程序所应用。WebRTC 通过UDP 协议来建立连接，因此它并不会通过你在浏览器配置文件中使用的代理服务器进行路由。即便您使用了代理，网站也能借此获取您真实的公共和本地IP 地址。该插件可被用于泄漏你的本地IP 地址或追踪媒体设备。

WebRTC 插件会泄漏什么信息

- 公网IP 地址
- 内网IP 地址
- 媒体设备的数量及其哈希值

WebRTC 的不同模式

Altered 替换模式 - WebRTC 的公共IP 和本地IP 地址会得到替换

Disabled 禁用模式 - WebRTC 插件将在浏览器配置文件中被禁用。

Real 真实模式 - WebRTC 插件将被启用，并会泄露您的真实IP 地址。如果你没有通过代理连网（例如，您通过 4G /5G 或直接固定电话连接），则应使用该模式。

替换模式的设置

在替换模式中，WebRTC 的参数——公共IP 和本地IP 参数，会在浏览器文件启动时被配置好，与浏览器配置文件设置中的IP 地址相匹配。

有效的内网IP 范围

内网IP 地址必须处于有效地址范围内，比如：

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

通常情况下，住宅用户的 C 组（第 3 个号码）为 0，D 组（第 4 个号码）为任意数字。企业在线用户在C 组和D 组中都可能有不同的数字，而不是 0。

一些内网IP 地址的例子：

- 192.168.0.21

- 192.168.0.23
- 172.16.0.168

一些典型的企业用户的本地IP 案例：

- 192.168.31.123
- 172.16.11.22
- 10.0.12.192

Navigator 参数

JS.Navigator 是一组Javascript 对象，它储存了各种参数及其值，用于描述所使用的计算机的细节。浏览器可以自由访问所有JS.Navigator 对象参数。由于它们具有一些独特性，特别是在各成分组合时，网站可以利用这些参数识别和追踪用户指纹。

网站也会分析这些设置的一致性，从而揭示指纹的变化。这样的分析可能会暴露浏览器指纹随机发生器的使用。

User-Agent(用户代理)

User-Agent(用户代理)是一种浏览器的原生短字符串。通过读取用户代理字符串，网站可以识别您的浏览器及操作系统的版本。

下面是一个用户代理值的示例

Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.2785.8Safari/537.36

在这个例子中，网站将推测用户使用的是 Windows 8.1 和 Chrome 62。“NT 6.3”是Windows 发行的不同版本。您可以在Wikipedia article

维基百科中查看其他已发行的版本。

创建浏览器配置文件时，遵循您在概览页面的关于操作系统过滤器的选择，用户代理值从 VMLogin 的指纹数据库被获取。您可以在导航栏页面查看浏览器配置文件的用户代理值。

如果您决定在VMLogin 中手动设置用户代理值，请确该值与平台值（Platform）保持一致。用户代理值和平台存在差异将是很严重的错误

Platform(平台)

平台的属性是一个Navigator 的对象参数，可以用于指明浏览器的编译平台。

新建浏览器文件时，平台值和用户代理同时被获取。两个值都会受到概览页面的操作系统过滤器的影响。您也可以在导航栏（Navigator） 页面手动设置这个值。

如果您要在VMLogin 中手动设置平台（Platform），请确保它和用户代理值相匹配。用户代理值和平台值存在差异是一个非常危险的信号。

桌面浏览器可用的平台值：

- Linux i686
- Linux armv7l
- MacIntel
- Win64
- Win32

移动浏览器可用的平台值：

- iPhone
- iPod
- iPad
- Android

您可以从Stackoverflow article 上获取可用Navigator.Platform 值的完整列表。

屏幕分辨率

分析屏幕分辨率值是浏览器指纹识别的常用方法。网站也会分析浏览器所体现的屏幕分辨率和实际可用的屏幕区域大小之间的差异。这种差异可以探测用户是否使用了着在线隐私工具，例如浏览器隐私附加元件。

在 VMLogin 中，屏幕分辨率从生成浏览器配置文件的指纹数据库中被提取。您也可以通过在常用分辨路列表中选择您需要的分辨率或者通过手动输入数值来人为更改此参数。此外，重新获取指纹时，您收到的屏幕分辨率下拉列表和屏幕分辨率都将根据您在概览页面所选的操作系统进行相应的调整。

VMLogin 将运行浏览器配置文件中所设浏览器分辨率的最大值。最大化浏览器窗口对大多数用户来说是十分常见的典型行为，我们不建议您在使用将其非最大化。我们也不建议在浏览器配置文件中使用大于

您真实屏幕的分辨率，因为网站会发现您没有使用最大化窗口。

将浏览器配置文件共享给他人时，我们建议浏览器配置文件所用的分辨率不应超过您的团队所使用的最小屏幕。例如，您的设备是 4K， 您同事使用FullHD（1920x1080）显示器。这时，我们建议您将浏览器配置文件的分辨率保持在 1920x1080 或更小值。否则，当您的团队在不同设备上打开该浏览器配置文件时，窗口的实际大小可能不同。

语言设置

Navigator 对象的语言设置可以帮助网站识别您的首选语言。网站会基于这个值，调整向您呈现内容的所用语言。与其他任意Navigator 对象值一样，它也可用于浏览器指纹识别。

新建浏览器文件时，会将其默认设置为人们普遍最常用的语言。VMLogin 并不会从指纹数据库中随机获取这个值，因为这会导致浏览器语言与IP 位置严重不匹配。例如，您在德国的某处却首选菲律宾语。

启用【字体】指纹保护

字体指纹识别是一种身份验证的方式。基于用户所使用的字体种类及字体在浏览器中被绘出的方式，网站可以进行指纹追踪。通常，网站在浏览器指纹识别时有两种运用字体的方法。

- 枚举字体列表
- 基于字体测量的指纹识别

您可以通过Browserleaks.com，从这里查看这些方法是如何被应用的。

枚举字体列表

收集电子设备上已安装的字列表的最常用方法是CSS 的自我检测。简而言之，该方法通过测量浏览器上特定字体显示的一个短语的宽度来获取您的字体列表。如果宽度匹配，就意味着您安装了这种字体。如不匹配，则推定这种字体没有被安装。

通过循环检测可能安装的字体列表及宽度，网站可准确窥探一台设备安装了哪些字。

VMLogin 使用一种特殊算法来对抗这种探测方法，允许您控制可供网站枚举的字体以反追踪。

启用硬件指纹【Canvas】保护

Canvas 是一种HTML5 API，用于在网页上绘出 2D 图像和动画。

除了上述功能之外，Canvas 也可以作为浏览器指纹识别的附加熵。根据 Englehardt 和 Narayanan 在普林斯顿大学的一项研究（2016），超过 5%的网站使用Canvas 来进行指纹识别。

综上，Canvas 通过命令浏览器绘制一个隐藏的Canvas 图像来实现指纹识别。在不同的机器上，这张图片的绘制结果略有不同；但如果机器相同，则图像也相同。图像被绘出后，它会被转换成一个哈希字符

串，被进一步用于身份验证的额外熵。

VMLogin 提供了三种不同的操作模式来控制浏览器文件的Canvas 指纹：噪声模式、关闭模式和封锁模式。

噪声模式（Noise mode）

当网站通过浏览器请求读取Canvas 函数时，噪声模式下的Canvas 屏蔽算法会中途拦截它，并向读出添加一个随机但始终会保持一致的噪声。为了更好地理解其工作原理，我们可以将其类比为一个“语音修正器”。当您使用一个有着特定预设的语音修饰器时，它会改变你的声音，使它与原来的声音有很大的区别，但随着时间的推移这种变化将保持一致。

由于读出添加了随机的噪声，如果网站应用了数据分析技术，便会发现指纹是 100%唯一的。

Your Fingerprint :

Signature	✓ 47327046
Uniqueness	100% (0 of 258561 user agents have the same signature)

关闭模式（Off Mode）

将 Canvas 设置成关闭模式后，网站将会得到您设备的真实Canvas 指纹。

将 Canvas 设置成“关闭模式”在某些情况下是有利的，比如当网站对 100%唯一的或监测到Canvas 被拦截这些情况有较糟糕的回应时。

请注意！在真实环境中，Canvas 指纹的哈希值并不是唯一的，因为世界各地有着与您的设备相同的副本。所以如果您显示了真实的 Canvas 指纹，您只会被划分到使用同一硬件的用户群。此外，通过变更其他指纹，您可以增加网站将您的浏览器配置文件视为单独身份的熵值。**封锁模式（Block mode）**

封锁模式完全禁止网站读取Canvas。当网站试图从浏览器配置文件中读出已被设置为封锁模式的Canvas 时，返回的值将为空。

这种情况处理方式完全取决于网站自身的权衡。然而，在检索 Canvas 对象数据的过程中发生浏览器错误的情况下，这样的事件也可能发生在没有专心隐藏自己的Canvas 指纹的用户身上。

在多台电脑上打开浏览器配置文件

请注意！如果您创建了一个将Canvas 设置为噪声模式的浏览器配置文件，并且在装有不同硬件的多个设备上打开它，网站就会知道 Canvas 的哈希值在多平台上运行时并不是持续的。

虽然添加的噪声是持续的，然而它在运行的设备上起到的只是过滤器的作用。所以，如果设备改变了，那么读出也会改变。

同一个浏览器配置文件在两个不同的设备上被打开。虽然这个浏览器配置文件的噪声是持续的，但Canvas 读出却仍旧不同。

如果您需要在多个设备上获得不变的读出，您可以尝试以下几种解决方法：

在硬件指纹设置为噪声模式的情况下，在配置相同的虚拟机（VM）或虚拟专用服务器（VPS）上运行VMLogin。由于这些设备是由同一种方式设置的，添加噪声后的 Canvas 指纹在多个设备上会保持一致。在有着相同硬件、驱动程序、操作系统的同一PC 型号上运行VMLogin。由于这些设备有着同样的硬件设置，被掩蔽后的系统指纹在多设备上会保持一致。

启用硬件指纹【AudioContext】保护

AudioContext 指纹（也被称作“音频指纹”）是设备音频栈的哈希衍生值。它的工作原理如下。基于您的音频设置和硬件，网站要求您的浏览器把播放音频文件的方式模拟为一个正弦函数。这个正弦函数被转化为一个哈希函数并发送给服务器，作为浏览器指纹识别中的附加熵。

在 VMLogin 中，您可以通过添加随机的持续性噪声来控制 AudioContext 的读出，或允许网站获取您设备的真实音频指纹。

Noise mode 噪声模式

通过在AudioContext 区开启噪声模式，VMLogin 会在浏览器层面修改音频堆，从而产生唯一的音频指纹。

由于音频栈是通过随机数值修改的，因此如果应用数据分析，网站会

发现指纹具有 100%唯一性。

关闭模式

将 AudioContext 屏蔽设置成关闭模式后，网站将会得到您设备的真实音频指纹。

在某些情况下，将模式设置为关闭可能是有利的，特别是当网站对 100%唯一的 AudioContext 读出回应较差时。

请注意！在现实世界中，音频指纹的哈希值并不是唯一的，因为与您的设备和音频栈相同的副本存在于世界各地。所以如果你显示了真实的音频指纹，只会被划分到使用同一音频硬件的用户段。此外，通过变更其他指纹，您可以增加网站将您的浏览器配置文件视为单独身份的熵值。

启用硬件指纹【WebGL】保护

WebGL 是一种 JavaScript 浏览器API，用于在网页上呈现 3D 图像。网站可利用WebGL 来识别您的设备指纹。通常，网站可以用两种方法来做到这一点：

WebGL 报告——完整的WebGL 浏览器报告表是可获取、可被检测的。在一些情况下，它会被转换成为哈希值以便更快地进行分析。 WebGL 图像 ——渲染和转换为哈希值的隐藏 3D 图像。由于最终结

果取决于进行计算的硬件设备，因此此方法会为设备及其驱动程序的不同组合生成唯一值。这种方式为不同的设备组合和驱动程序生成了唯一值。

您可以通过Browserleaks test 检测网站来查看网站可以通过该API 获取哪些信息。

WebGL 元数据掩蔽

当您将WebGL 设为噪音模式时，WebGL 元数据就会被VMLogin 所掩蔽。这是一种旧有机制，以后我们将通过分别隐藏WebGL 元数据和图像来改进这一功能。

启用元数据掩蔽后，VMLogin 将根据从指纹数据库中获取的值来更改WebGL 供应商和渲染器参数。

关闭模式

将 WebGL 屏蔽设置成关闭模式后，网站将会得到您设备的真实WebGL 报告和图像哈希值。

设置成关闭模式在某些情况下是有利的，比如当网站对 100%唯一的或监测到WebGL 读出被拦截这些情况有较糟糕的回应。

媒体设备指纹设置

WebRTC 是一种浏览器插件，通过直接的P2P 连接，促使网页内的音频和视频通话，从而无需安装额外插件或其他本地应用。为了使这个插件工作，WebRTC 会连接到您的媒体设备，例如麦克风、摄像头和耳机。网站可通过以下两种方式利用这一追踪机制：

- 设备枚举
- 媒体设备ID 追踪

您可以在Browser leaks test website 中查看这两种身份验证方法。

设备枚举

这种方法依赖于检索用户已安装的麦克风、摄像头和耳机的完整列表来运作。虽然仅这个数字不足以明确地定位用户，但它仍然可以发挥一定作用。

在 VMLogin 中，您可以根据需要来控制浏览器配置文件中不同媒体设备的数量。

您可以在以下范围内更改参数：

- 视频输入（网络摄像头的数量）：0-1
- 音频输入（麦克风的数量）：0-4
- 音频输出（扬声器的数量）：0-4

理论上说，每个用户的设备数量都可以有超出上述范围。但由于这种情况并不常见，这些数量就被我们设定在了通常在线用户所用的范围内。

媒体设备ID

为了使WebRTC 正常工作，网站不仅需要知道您拥有的设备数量和类型。为了建立完善的实时通讯，唯一的设备标识符也是必须的。您可以联想一下您的设备地址。当然，浏览器不会允许网站得知你设备的完整型号名称，它们会用哈希值来替代，这就是设备ID。与此同时， 网站也可以使用这些值用于用户识别。

由于媒体设备ID 对每一个用户来说都是唯一的，因此它在浏览器指纹识别中是一种特别有效的技术。

在 VMLogin 中，当这种功能被开启后，每一个设备的真实设备ID 都会得到掩蔽。

如果您关闭了媒体设备掩蔽功能，您的真实设备 ID 就会被网站获取。即便你开启了WebRTC 的 IP 掩蔽，这种情况仍旧会发生。

默认首页：每次启动该配置的浏览器时，会自动打开这个网址。

B. 浏览器配置文件管理

浏览器配置文件列表，可以使用右键菜单来进行相关操作。

右键菜单中可以:

启动浏览器：启动配置文件对应的浏览器

编辑配置文件：对浏览器配置进行修改

移动配置文件分组：可以移动配置文件到指定分组

分享配置文件：可以所配置文件分享给其它用户

转移配置文件所有权：可以把配置文件转移给其它用户

删除配置文件：从自己的帐户删除浏览器配置文件

添加分组

可以管理分组列表，同时可以添加新组名称。

Cookie 导入

浏览器cookie，又称为HTTP cookie，是用于存储用户数据、保存在用户本地终端的数据。Cookie 的主要功能之一是用户认证。通过在VMLogin 中使用 cookie 导入功能，您可以在启动的浏览器中导入您从常规浏览器导出的cookie。

如何在 VMLogin 中导入 Cookie

可以在启动的浏览器中导入JSON 和 Netscape 格式的 Cookies。在导入JSON 格式的cookie 之前，您需要使用 JSON 校验器检查文件的格式是否正确。

Cookie 格式样例：

[{"name": "xs", "path": "/", "value": "2:cDMG:2:1582872010:-1:-1", "domain": ".vmlogin.com.cn", "secure": true, "expires": 1589094326, "session": false, "hostOnly": false, "httpOnly": false}, {"name": "fr",

"path": "/", "value": "3v1si8FPVJV.AAWWjnhtd", "domain": ".vmlogin.com.cn", "secure": true, "expires": 1589094326, "session": false, "hostOnly": false, "httpOnly": false}]

C. 我的帐户

在我的帐户栏目中可以查看自己的帐户信息，可以配置浏览器自动化设置。

浏览器自动化允许您在 VMLogin 的浏览器配置文件中自动执行任务。从创建简单的自动化脚本到复杂的 Web 爬虫，可以搜索、收集 Web 数据并与之交互。

VMLogin 浏览器自动化基于 Selenium WebDriver。

通常情况下，如果您运行 Selenium 代码，首先将连接到 Chrome 驱动， 然后设置您所需要的功能。而将 VMLogin 与 Selenium 代码结合使用时， 您无需这样操作。您将使用 Remote Web Driver 程序，通过本地端口连接到 Vmlogin 应用或某浏览器配置文件， 设置所需功能， 在预定义的浏览器配置文件中执行Selenium 命令。

支持的语言

Selenium 框架提供了多种可搭配使用的语言，因此 VMLogin 自动化也可以在多种编码语言上运行。但是目前，我们仅为 Java和Python 供技术支持。

在 VMLogin 中使用 Selenium

定义 VMLogin 端口

您需要提前定义软件端口以使用 Selenium 自动化。以下是定义端口的方法：

在软件《我的帐户》中打开启用浏览器自动化设置，并在监听端口中设置能使用端口，这里默认是 35000，另外你也可以设置一个访问密码。

随后，您就可以通过定义的端口连接到 VMLogin 了。

Java 案例：

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.StringWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Base64;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

import org.apache.commons.lang3.RandomStringUtils;
import org.apache.commons.lang3.RandomUtils;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.client.fluent.Executor;
import org.apache.http.client.fluent.Form;
import org.apache.http.client.fluent.Request;
import org.apache.http.util.EntityUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.WebDriverWait;

import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.pp.tst.email.EmailUtil;


public class BrowserProfile {
 public static void main(String[] args) throws Exception {
 BrowserProfile bp = new BrowserProfile();
 String profileId = "0034667E-0A21-4F1C-90E4-937C0AE7181A";

 URL url = new URL(bp.startProfile(profileId));
 System.out.println(url.getAuthority());
 Thread.sleep(3000);

 System.setProperty("webdriver.chrome.driver",
"D:\\vmlogin\\chrome\\86.0.4240.75\\chromedriver.exe");// 指到驱动位置
 ChromeOptions chromeOptions = new ChromeOptions();
 chromeOptions.setExperimentalOption("debuggerAddress", url.getAuthority());
 WebDriver driver = new ChromeDriver(chromeOptions);

 driver.get("https://vmlogin.com.cn/");
 driver.close();
 }

 @SneakyThrows
 private String startProfile(String profileId) {
 String url =
"http://127.0.0.1:35000/api/v1/profile/start?skiplock=true&profileId=" +
profileId;
 URL obj = new URL(url);
 HttpURLConnection con = (HttpURLConnection) obj.openConnection();
 con.setRequestMethod("GET");
 BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
 String inputLine;
 StringBuffer response = new StringBuffer();
 while ((inputLine = in.readLine()) != null) {
 response.append(inputLine);
 }
 in.close();
 System.out.println(response);
 HashMap data = new ObjectMapper().readValue(response.toString(),
HashMap.class);
 return (String) data.get("value");
 }
}
```

接口还可以传入代理服务器信息，如果传入代理信息会覆盖配置文件里的代理信息，这种覆盖是临时性的，不会真的修改配置文件，只对自动化接口有效：

http://127.0.0.1:35000/api/v1/profile/start?automation=true&profileId=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx&proxytype=socks5&proxyserver=ip&proxyport=1080&proxyusername=&proxypassword=

代理类型可能是这四种：

proxytype=socks5proxytype=socks4proxytype=http=https

代理用户名和密码可以不传为空。