# SpringBoot+selenium模拟用户操作浏览器

## Selenium

Selenium是一个用于Web应用程序自动化测试的开源工具套件。它主要用于以下目的：

1. **浏览器自动化**：Selenium能够模拟真实用户在不同浏览器（如Chrome、Firefox、IE/Edge等）中的交互行为，通过编程方式控制浏览器执行一系列操作，例如点击按钮、填写表单、导航页面等。
2. **兼容性测试**：通过编写脚本在多种浏览器和操作系统环境下运行测试，确保Web应用在不同组合下都能正确工作。
3. **功能测试**：创建回归测试用例来验证软件功能是否按预期工作，以及用户需求是否得到满足。
4. **端到端测试**：进行从用户界面到后端服务的整体流程测试，检查系统组件间的集成效果。
5. **支持多种语言**：Selenium WebDriver API 可以使用Java、Python、C#、Ruby等多种编程语言实现测试脚本。
6. **与DevTools集成**：在较新的Selenium版本中，提供了对Chromium内嵌的Chrome DevTools的支持，允许开发者执行更深层次的浏览器操作和调试任务。

总的来说，Selenium帮助开发团队提高Web应用的质量保障效率，通过自动化测试减少手动测试的重复性和复杂性，并有助于持续集成和持续部署（CI/CD）流程的实施。

## 下载Selenium

Selenium的下载通常分为两个步骤：首先，你需要下载并安装适用于你开发环境的Selenium WebDriver库（针对不同编程语言有不同的库）；其次，根据你的测试需求下载对应浏览器的驱动程序。

以下是一些常见编程语言中Selenium WebDriver库的下载和安装方式：

**Python**：

使用pip工具进行安装：

```
pip install selenium
```

**Java**：

在Maven项目中添加如下依赖到pom.xml文件：

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>最新版本号</version>
</dependency>
```

或在Gradle项目中添加：

```
implementation 'org.seleniumhq.selenium:selenium-java:最新版本号'
```

如果不使用构建工具，可以直接从Maven仓库下载selenium-java.jar包。

**C#** ：

对于.NET环境，可以通过NuGet包管理器在Visual Studio中安装Selenium.WebDriver NuGet包。

下载浏览器驱动
---

安装完WebDriver库后，接下来需要下载浏览器驱动，例如：

* **ChromeDriver**：从ChromeDriver官网
([sites.google.com/a/chromium....](http://cxyroad.com/”https://sites.google.com/a/chromium.org/chromedriver/downloads%EF%BC%89%E4%B8%8B%E8%BD%BD%E4%B8%8E%E4%BD%A0%E7%9A%84Chrome%E6%B5%8F%E8%A7%88%E5%99%A8%E7%89%88%E6%9C%AC%E7%9B%B8%E5%8C%B9%E9%85%8D%E7%9A%84%E9%A9%B1%E5%8A%A8%E3%80%82”)
* **GeckoDriver**（Firefox）：访问GeckoDriver GitHub发布页面
([github.com/mozilla/gec...](http://cxyroad.com/”https://github.com/mozilla/geckodriver/releases%EF%BC%89%E4%B8%8B%E8%BD%BD%E9%80%82%E5%90%88%E4%BD%A0Firefox%E7%89%88%E6%9C%AC%E7%9A%84%E9%A9%B1%E5%8A%A8%E3%80%82”)
* **EdgeDriver** 或 **IEDriverServer**：分别在Microsoft WebDriver下载页面和Selenium官方IEdriver下载页获取相应驱动。

将下载的浏览器驱动放在系统的PATH路径下，或者在代码中明确指定驱动路径，即可配合WebDriver开始编写自动化测试脚本了。

获取最新版本的chrome和chromeDriver
[sites.google.com/a/chromium....](http://cxyroad.com/”https://sites.google.com/a/chromium.org/chromedriver/downloads”)

Java代码
======

yml配置文件
-------

```
system:
  #驱动信息
  driver:
    mode: 1  # 驱动模式（0：本地 1：远程）
    headless: 0  #是否显示浏览器（0：显示 1：不显示）
    name: webdriver.chrome.driver
    path: D:\drivers\chromedriver-win64\chromedriver.exe
    # 远程driver驱动地址
    remoteUrl: http://127.0.0.1:9515
    systemUrl: http://目标服务器地址
```

```

工具类：包含打开、关闭、刷新浏览器功能
--------------------

```

```
import cn.com.huacloud.basic.audit.dto.MyParam;
import
cn.com.huacloud.supervision.common.exception.ApplicationException;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import java.net.ConnectException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Set;
import java.util.concurrent.TimeUnit;

@Slf4j
@Component
public class WebDriverUtils {

    @Autowired
    private Environment env;

    /**
     * 是否显示浏览器（0：否 1：是）
     */
    private static String headless;
```

```
/**
 * 驱动远程地址
 */
private static String driverRemoteUrl;

private static String systemUrl;

@PostConstruct
public void init() {
    headless = env.getProperty("system.driver.headless");
    driverRemoteUrl = env.getProperty("system.driver.remoteUrl");
    systemUrl = env.getProperty("system.driver.systemUrl");
}

/**
 * 等待时间
 */
public final static int timeSeconds = 10;

private static String getRemoteUrl(String driverRemoteUrl) {
    String remoteUrl = null;
    if (StringUtils.isNotBlank(driverRemoteUrl)) {
        if (driverRemoteUrl.contains(",")) {
            String[] split = driverRemoteUrl.split(",");
            remoteUrl = split[(new Random()).nextInt(split.length)];
        } else {
            remoteUrl = driverRemoteUrl;
        }
    }
    return remoteUrl;
}

/**
 * 获取驱动
 *
 * @return
 * @throws ConnectException
 */
public static WebDriver getDriver() {

    log.info("进入启动浏览器");
    ChromeOptions chromeOptions = new ChromeOptions();
    // 设置浏览器是否可见
    if ("1".equals(headless)) {
        chromeOptions.addArguments("--headless");
        chromeOptions.addArguments("--disable-gpu");
        // headless模式下必须设置这两个参数，否则会报连接超时
```

```
            chromeOptions.addArguments("--proxy-server='direct://'");
            chromeOptions.addArguments("--proxy-bypass-list=*");
        } else {
            chromeOptions.addArguments("--no-sandbox");
        }
        WebDriver driver = null;
        log.info(String.format("driverRemoteUrl:%s",  driverRemoteUrl));
        DesiredCapabilities dc = DesiredCapabilities.chrome();
        dc.setCapability(ChromeOptions.CAPABILITY, chromeOptions);
        try {
            driver = new RemoteWebDriver(new
URL(getRemoteUrl(driverRemoteUrl)), dc);
        } catch (Exception e) {
            e.printStackTrace();
            throw new ApplicationException(-1, "RemoteWebDriver连接远程
驱动失败");
        }
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(timeSeconds,
TimeUnit.SECONDS);
        driver.manage().timeouts().pageLoadTimeout(timeSeconds,
TimeUnit.SECONDS);
        driver.manage().timeouts().setScriptTimeout(timeSeconds,
TimeUnit.SECONDS);
        return driver;
    }

    /**
     * 关闭浏览器驱动
     *
     * @param driver
     */
    public static void quitBrowser(WebDriver driver) {
        try {
            if (driver != null) {
                driver.close();
                driver.quit();
                log.info("浏览器关闭成功");
            }
        } catch (Exception e) {
            log.error("浏览器关闭异常 - {}", e.getMessage());
            throw new ApplicationException(-1, e.toString());
        }
    }

    public static synchronized boolean openBrowser(WebDriver driver) {
        log.info("即将打开浏览器");
        try {
```

```
            driver.get(systemUrl);
            return true;
        } catch (Exception e) {
            log.error("打开浏览器异常,{}",e);
            return false;
        }
    }

    /**
     * 刷新浏览器
     * @param driver
     * @return
     */
    public static synchronized void refreshBrowser(WebDriver driver) {
        driver.navigate().refresh();
    }

    private static boolean alertExists(float timeout, WebDriver driver) {
        long start = System.currentTimeMillis();
        while ((System.currentTimeMillis() - start) < timeout * 1000) {
            try {
                driver.switchTo().alert();
                return true;
            } catch (Exception e) {
            }
        }
        return false;
    }

    /**
     * 关闭登录消息提醒弹窗
     *
     * @param timeout
     * @param driver
     */
    private static void closePrompt(float timeout, WebDriver driver) {
        long start = System.currentTimeMillis();
        while ((System.currentTimeMillis() - start) < timeout * 1000) {
            try {
                close_ymPrompt(driver);
            } catch (Exception e) {
                log.error("无法找到登陆消息提醒弹窗");
            } finally {
                sleep("500");
            }
        }
    }
```

```
    public static void close_ymPrompt(WebDriver driver) {
        WebElement closePromptEle =
driver.findElement(By.className("ymPrompt_close"));
        closePromptEle.click();
    }

    /**
     * 点击弹窗
     *
     * @param driver
     */
    private static void clickConfirmEle(WebDriver driver) {
        try {
            long start = System.currentTimeMillis();
            // 点击确定
            while (System.currentTimeMillis() - start < 2 * 1000) {
                WebElement saveconfirm =
driver.findElement(By.id("ymPrompt_btn_0"));
                saveconfirm.click();
                sleep("500");
            }
        } catch (Exception e) {
            log.debug("没有确定弹窗");
        }
    }

    /**
     * 获取属性值
     *
     * @param eleName
     * @param attributeName
     * @return
     */
    public static String getAttributeByEle(WebElement webElement, String
eleName, String attributeName) {
        WebElement webEle =
webElement.findElement(By.name(eleName));
        return webEle.getAttribute(attributeName);
    }

    /**
     * 休眠
     *
     * @param millis
     */
    public static void sleep(String millis) {
        try {
            long waittingTime1 = Long.parseLong(millis);
```

```
            Thread.sleep(waittingTime1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static String findWindow(WebDriver driver, String title) {// 找到
属于该标题的窗口句柄
        Boolean find = false;
        String findhandle = null;
        long start = System.currentTimeMillis();
        while (!find && (System.currentTimeMillis() - start) < 10 * 1000) {
            Set<String> handles = driver.getWindowHandles();
            for (String handle : handles) {
                if (driver.switchTo().window(handle).getTitle().equals(title)) {
                    findhandle = handle;
                    find = true;
                    break;
                }
            }
        }
        return findhandle;
    }


    private static List<WebElement> findWebElements(WebDriverWait
wait, String ele) {
        try{
            return
wait.until(ExpectedConditions.presenceOfAllElementsLocatedBy(By.name
(ele)));
        }catch (Exception e){
            log.error("未查询到元素");
        }
        return null;
    }

    /**
     *
     * @return 当前打开窗口的最后一个句柄
     */
    public static String getLastHandle(WebDriver driver) {
        Set<String> Allhandles = driver.getWindowHandles();//获取当前打开
窗口的所有句柄
        List<String> lst = new ArrayList<String>(Allhandles);
        return lst.get(lst.size()-1);
    }
```

}

```

模拟用户浏览器操作
---

我把这段代码也放到工具类里了

```

```
    /**
     * 模拟登录某个系统
     * @param driver
     * @param wait
     * @param username
     * @param password
     * @return
     */
    public static synchronized boolean loginSystem(WebDriver driver,
WebDriverWait wait, String username,
                                                   String password) {
        log.info("即将登录，页面地址 - [{}]", systemUrl);
        try {
            log.info("即将登录，登录信息 - [账号: {}]", username);

            sleep("1000");
            WebElement usernameElement =
driver.findElement(By.id("name"));
            // 输入用户名
            usernameElement.sendKeys(username);
            WebElement passwordElement =
driver.findElement(By.id("pwd"));
            // 输入密码
            passwordElement.sendKeys(password);

            // 点击登录
            log.info("点击登录");
            WebElement loginElement = driver.findElement(By.id("login"));
            loginElement.click();
            // 等待2秒，防止目标系统反应过慢
            sleep("2000");
            //获取弹框值判断是否登录成功
            Alert alert = driver.switchTo().alert();
            String text = alert.getText();
```

```
            alert.accept();
            log.info("登录成功");
            return true;
        } catch (Exception e) {
            log.error("登录异常,{}",e);
            return false;
        }
    }

    /**
     * 模拟填充表单内容并保存
     * @param driver
     * @param param
     * @return
     */
    public static synchronized boolean sendGzjs(WebDriver driver,
MyParam param) {
        try {
            log.info("开始填充表单：{}", param.toString());
            WebElement ndElement = driver.findElement(By.id("nd"));
            // 输入年度
            ndElement.sendKeys(param.getNd());

            WebElement yfElement = driver.findElement(By.id("yf"));
            // 输入月份
            yfElement.sendKeys(param.getYf());

            WebElement contentElement =
driver.findElement(By.id("content"));
            // 输入内容
            contentElement.sendKeys(param.getContent());

            log.info("点击保存");
            WebElement loginElement = driver.findElement(By.id("save"));
            loginElement.click();

            // 等待2秒，防止目标系统反应过慢
            sleep("2000");

            //获取弹框值判断是否保存成功
            Alert alert = driver.switchTo().alert();
            String text = alert.getText();
            alert.accept();
            if ("目标系统返回的错误信息!".equals(text)) {
                log.info("发送失败");
                return false;
            }
            log.info("发送成功");
```

```
            return true;
        } catch (Exception e) {
            log.error("发送异常,{}",e);
            return false;
        }
    }
```

原文链接: https://juejin.cn/post/7374295163625783305