06 WebDriver 常用API-1

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

访问某网页地址

String baseUrl="https://www.baidu.com/";

- 方法1：

```
@Test
public void visitURL(){
        driver.get(baseUrl);

}
```

- 方法2：

```
@Test
public void visitURL(){
        driver.navigate().to(baseUrl);
}
```

返回/前进/刷新

```
@Test
public void visitRecentURL() {
    driver.get(url1);
    driver.get(url2);
    driver.navigate().back();
    driver.navigate().forward();
    driver.navigate().refresh();
}
```

获取页面信息

```
@Test
public void getPage() {
    driver.get(baseUrl);
    String title = driver.getTitle();// 获得title
    String pageSource = driver.getPageSource();// 获得源代码
    String pageUrl = driver.getCurrentUrl();
    System.out.println(title + pageSource + pageUrl);
}
```

全屏显示

```
driver.manage().window().maximize()
```

句柄+网页title

使用getWindowHandle方法来获取当前浏览器窗口的句柄
使用getWindowHandles方法获取所有弹出的浏览器窗口的句柄

```
@Test
public void controlWindow() {
    wd.get("http://localhost:8032/MyMovie/");
    JavascriptExecutor js = (JavascriptExecutor) wd;
    js.executeScript("window.open('https://www.baidu.com/')");
    Set<String> windows = wd.getWindowHandles();
    for (String s : windows) {
        System.out.println(s);
        boolean status = wd.switchTo().window(s).getTitle().contains("My Movie");
        if (status) {
            wd.findElement(By.linkText("登录")).click();
        }
    }
    wd.close();
}
```

句柄+网页内容

```
public void identifyWindowByPageSource() {
    String pwindowHandle = driver.getWindowHandle();
    System.out.println(pwindowHandle);
    WebElement link1 = driver.findElement(By.tagName("a"));
    link1.click();
    Set<String> allwindowsHandles = driver.getWindowHandles();
    if (!allwindowsHandles.isEmpty()) {
        for (String windowHandle : allwindowsHandles) {
            if (driver.switchTo().window(windowHandle).getPageSource()
                    .contains("百度新闻")) {
                driver.findElement(By.tagName("a")).click();
                System.out.println("定位到第二个页面");
            }
        }
    }
    driver.switchTo().window(pwindowHandle);
    System.out.println(driver.getTitle());
}
```

关闭浏览器

close方法关闭当前的浏览器窗口；
quit方法不仅关闭窗口，还会彻底的退出webdriver，释放与driver server之间的连接。
所以简单来说quit是更加彻底的close，quit会更好的释放资源。

```
    @AfterMethod
    public void closeBrowser() {
        driver.quit();
//      driver.close();
    }
```

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

输入框

```
public void inputText() throws InterruptedException {
    driver.get(baseUrl);
    WebElement input = driver.findElement(By.id("uname"));
    input.clear();// 清空
    Thread.sleep(3000);
    input.sendKeys("hello");// 输入
    input.getText();
    input.submit();
    Thread.sleep(3000);
}
```

按钮单击

```
@Test
public void buttonClick() throws InterruptedException {
    driver.get(baseUrl);
    WebElement button = driver.findElement(By.id("u_submit"));
    button.click();
}
```

下拉列表框的操作

```
@Test
public void operateDropList() {
    Select dropList = new Select
            (driver.findElement(By.name("fruit")));
    assertFalse(dropList.isMultiple());// 是否允许多选
    assertEquals(dropList.getFirstSelectedOption().getText(), "桃子");
    dropList.selectByIndex(2);// 0表示第一个选项
    assertEquals(dropList.getFirstSelectedOption().getText(),  "桔子");
    dropList.selectByValue("xigua");
    dropList.selectByVisibleText("猕猴桃");
}
```

多选列表的操作

```
@Test
public void operateMultipleDropList() {
    Select dropList = new Select
            (driver.findElement(By.name("course")));
    assertFalse(dropList.isMultiple());// 是否允许多选
    dropList.selectByIndex(0);
    dropList.selectByValue("tiyu");
    dropList.selectByVisibleText("程序设计基础");
    dropList.deselectAll();//取消所有选中的状态
    dropList.selectByIndex(3);
    dropList.deselectByValue("shuju");
}
```

单选框/复选框的操作

```
@Test
public void operateRadio() {
    WebElement radioSex = driver.findElement(By
            .xpath("//input[@value='1']"));
    if (!radioSex.isSelected())
        radioSex.click();
    assertTrue(radioSex.isSelected());
}
```

```
@Test
public void operateCheckBox() {
    WebElement checkBox = driver.findElement(By
            .xpath("//input[@value='11']"));
    if (!checkBox.isSelected())
        checkBox.click();
    assertTrue(checkBox.isSelected());
    List<WebElement> checkboxs = driver.findElements(By.name("hobby"));
    for (WebElement checkbox : checkboxs) {
        checkbox.click();//选中所有复选框
    }
}
```

检查文本内容是否出现

```
@Test
public void isElementTextPresent() {
    WebElement text = driver.findElement(By
            .xpath("//p[1]"));
    String contentText = text.getText();
    assertTrue(contentText.contains(contentText));
    assertEquals(contentText, "欢迎进入Selenium课程的学习");
}
```

窗口截屏

```
public void captureScreen() throws IOException {
    File source_file = ((TakesScreenshot) driver)
            .getScreenshotAs(OutputType.FILE);
    // 把当前浏览器打开的页面进行截图，保存到一个File对象中
    FileUtils.copyFile(source_file, new File("D:\\edutest\\test.jpg"));
    // 把File对象转换为一个jpg文件
}
```

练习：请把文件名修改为年月日-时分秒.jpg
例如20160830-165210.jpg

查看页面元素的属性

```
@Test
public void  getWebElementAttribute() {
    WebElement input = driver.findElement
            (By.id("uname"));
    String inputText = input.getAttribute("value");
    assertEquals(inputText, "默认的内容");
}
```

查看页面元素的CSS属性值

```
@Test
public void  getWebElementCssValue() {
    WebElement div = driver.findElement
            (By.id("div2"));
    String divwidth = div.getCssValue("width");
    assertEquals(divwidth, "200px");
}
```

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

操作iframe中的页面元素

```
public void testFrame() {
    // 进入左侧的frame页面
    driver.switchTo().frame("leftframe");
    WebElement pElement1 = driver.findElement(By.tagName("p"));
    assertEquals(pElement1.getText(), "左侧frame页面的文字");
    // 跳出左侧的frame页面，回到frameSet页面
    driver.switchTo().defaultContent();
    // 如果frame不存在id呢，可以使用索引号，从0开始
    driver.switchTo().frame(2);
    WebElement pElement2 = driver.findElement(By.xpath("//p"));
    assertEquals(pElement2.getText(), "右侧frame页面的文字");
}
```

使用iframe中的源代码来操作Frame

```
public void testFrameByPageSource() {
    // 找到页面上所有的frame对象，并储存到frames容器中
    List<WebElement> frames = driver.findElements(By.tagName("frame"));
    // 使用for循环遍历frames的所有frame页面，查找包含“左侧”frame的页面
    for (WebElement frame : frames) {
        driver.switchTo().frame(frame);
        if (driver.getPageSource().contains("左侧")) {
            WebElement pElement = driver.findElement(By.xpath("//p"));
            assertEquals(pElement.getText(), "左侧frame页面的文字");
            break;
        } else {
            // 如果没有找到指定的frame，则调用此行代码，返回到frameset页面，实现下一次循环
            driver.switchTo().defaultContent();
        }
    }
    driver.switchTo().defaultContent();
}
```

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

模拟键盘的操作

在webdriver的使用过程中，应该会用到使用工具来模拟用的鼠标、键盘的一些输入操作，比如说：

1、鼠标的左键点击、双击、拖拽、右键点击等；

2、键盘的回车、回退、空格、ctrl、alt、shift等；

```
@Test
public void clickKeys() {
    Actions action = new Actions(driver);
    action.keyDown(Keys.CONTROL);//按下
    action.keyDown(Keys.SHIFT);
    action.keyDown(Keys.ALT);
    action.keyUp(Keys.CONTROL);//释放
    action.keyUp(Keys.SHIFT);
    action.keyUp(Keys.ALT);
    action.keyDown(Keys.SHIFT).sendKeys("abcd").perform();
    //在浏览器的搜索框中输入大写的acd
}
```

模拟键盘的操作

```
public void testKeyOperate() throws InterruptedException{
    driver.get("http://localhost:8032/MyMovie/admin.php/Login/index.html");
    driver.findElement(By.name("username")).sendKeys("admin");
    Actions action  =new Actions(driver);
    action.sendKeys(Keys.TAB);
    action.sendKeys("admin").perform();
    action.sendKeys(Keys.ENTER).perform();
    Thread.sleep(5000);
}
```

模拟鼠标右键事件

```
@Test
public void rightClickMouse() {
    Actions action = new Actions(driver);
    action.contextClick(driver.findElement
            (By.id("uname"))).perform();
}
```

在指定元素进行鼠标悬浮

```
@Test
public void mouseoverElement() {
    WebElement link1 = driver.findElement
            (By.xpath("//a[@id='link1']"));
    Actions action = new Actions(driver);
    action.moveToElement(link1).perform();
}
```

指定元素上进行鼠标单击左键和释放的操作

```
public void mouseclickAndRelease() throws InterruptedException {
    driver.get("file:///D:/demo/0901api.html");
    WebElement div = driver.findElement(By.xpath("//a[@id='div2']"));
    Actions action = new Actions(driver);
    action.clickAndHold(div).perform();// 在指定控件单击左键不释放
    Thread.sleep(3000);
    action.release(div).perform();// 释放鼠标左键
    // 注意，clickAndHold和release 两个方法连起来使用，会被认为执行了一次事件click
    Thread.sleep(3000);
}
```

拖拽页面元素

```
public void dragPageElement() throws InterruptedException {
    driver.get("file:///D:/demo/0901api.html");
    WebElement div = driver.findElement(By.xpath("//div"));
    Actions action = new Actions(driver);
    action.dragAndDropBy(div, 0, 50).build().perform();
    Thread.sleep(5000);
}
```

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

执行JavaScript脚本

```
public void executeJavaScript() {
    // 声明JavaScript执行器对象
    JavascriptExecutor js = (JavascriptExecutor) driver;
    String title = (String) js.executeScript("return document.title");
    assertEquals("test07", title);
    System.out.println(title);
}
```

操作JavaScript 的Alert弹窗

```
public void testAlert() {
    WebElement btn = driver.findElement(By.id("su"));
    btn.click();
    // 获取Alert对象
    Alert alert = driver.switchTo().alert();
    // 使用alert.getText()获取对话框的文字
    assertEquals(alert.getText(), "这是一个alert弹出框");
    alert.accept();
}
```

```java
public void testConfirm() {
    WebElement btn = driver.findElement(By.id("btn2"));
    btn.click();
    // 获取Alert对象
    Alert alert = driver.switchTo().alert();
    // 使用alert.getText()获取对话框的文字
    assertEquals(alert.getText(), "这是一个confirm弹出框");
    alert.dismiss();//模拟【取消】按钮的操作
    //alert.accept();模拟【确定】按钮的操作
}
```

```java
@Test
public void testPrompt() {
    WebElement btn = driver.findElement(By.id("btn3"));
    btn.click();
    Alert alert = driver.switchTo().alert();
    // 使用alert.getText()获取对话框的文字
    assertEquals(alert.getText(), "这是一个prompt弹出框");
    alert.sendKeys("hello");
    alert.accept();
}
```

本章大纲

6.1 浏览器的操作

6.2 常见控件的操作

6.3 操作Frame中的页面元素

6.4 模拟键盘的操作

6.5 JavaScript的操作

6.6 隐式等待/显示等待

隐式等待

原理：隐式等待，就是在创建driver时，为浏览器对象设置一个全局的等待时间。这个方法是得不到某个元素就等待一段时间，直到拿到某个元素位置。过了这个时间如果对象还没找到的话就会抛出NoSuchElement异常。

注意：在使用隐式等待的时候，实际上浏览器会在你自己设定的时间内不断的刷新页面去寻找我们需要的元素

```
@Test
public void testImplictWait() {
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    try {
        WebElement input = driver.findElement(By.id("uname1"));
    } catch (NoSuchElementException e) {
        fail("元素不存在");
        e.printStackTrace();
    }
}
```

显式等待

等待的条件	方法
元素是否可用和被单击	elementToBeClickable(locator)
元素是否选中	elementToBeSelected(element)
元素是否存在	presenceOfElementLocated(locator)
元素是否包含特定的文本	textToBePresentInElement(locator, text)
页面元素值	textToBePresentInElementValue(locator, text)
title	titleContains(title)

显式等待

- 原理：显式等待,就是明确的要等到某个元素的出现或者是某个元素的可点击等条件,等不到,就一直等,除非在规定的时间之内都没找到,那么就抛出Exception。

```java
@Test
public void explictWait() {
    WebDriverWait wait = new WebDriverWait(driver, 10);
    wait.until(ExpectedConditions.titleContains("水果"));

    WebElement select = driver.findElement(By.id("peach"));
    wait.until(ExpectedConditions.elementToBeSelected(select));

    wait.until(ExpectedConditions.elementToBeClickable(By
            .xpath("//input[@type='checkbox']")));
    wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//p")));
    WebElement p = driver.findElement(By.xpath("//p"));
    wait.until(ExpectedConditions.textToBePresentInElement(p, "喜欢"));
}
```

自定义的显示等待

```
public void explictWait1() {
    try {
        WebElement input = (new WebDriverWait(driver, 10))
                .until(new ExpectedCondition<WebElement>() {
                    @Override
                    public WebElement apply(WebDriver d) {
                        return d.findElement(By.xpath("//*[@type='text]"));
                    }
                });
        assertEquals(input.getAttribute("value"), "西瓜");
        Boolean containText = (new WebDriverWait(driver, 10))
                .until(new ExpectedCondition<Boolean>() {
                    @Override
                    public Boolean apply(WebDriver d) {
                        return d.findElement(By.xpath("//p")).getText().contains("喜欢")
                    }
                });
    } catch (NoSuchElementException e) {
        fail("元素不存在");
        e.printStackTrace();
    }
```