05 WebDriver 高级应用

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

获取测试对象状态

- 是否存在NoSuchElementException
- 元素是否显示isDisplayed
- 是否被选中isSelected
- 是否可用（灰化状态）isEnabled

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

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);
}
```

使用JavascriptExecutor单击元素

click无法生效时，可以使用这个方法

```
public void testjsClick() {
    WebElement btn = driver.findElement(By.id("btn1"));
    JavaScriptClick(btn);
}

public void JavaScriptClick(WebElement element) {
    try {
        if (element.isEnabled() && element.isDisplayed()) {
            System.out.println("使用Javascript单击元素");
            JavascriptExecutor js = (JavascriptExecutor) driver;
            js.executeScript("arguments[0].click();", element);
        } else {
            System.out.println("元素无法单击");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

使用JavascriptExecutor操作页面的滚动条

- 滚动条到页面的最下面
- 滚动条到页面的某个元素
- 滚动条向下移动某个数量的像素

```
@Test
public void testScrolling() throws InterruptedException {
    driver.get("http://news.baidu.com/");
    WebElement link = driver.findElement(By
            .linkText("唐山迁安：苹果树“搬”进家 盆栽苹果正俏销"));
    JavascriptExecutor js = (JavascriptExecutor) driver;
    // 滚动条滑动至最下方
    js.executeScript("window.scrollTo(0,document.body.scrollHeight)");
    Thread.sleep(5000);
    // 滚动条滑动到指定元素的位置
    js.executeScript("arguments[0].scrollIntoView()", link);
    // 滚动条滑动至某像素位置
    js.executeScript("window.scrollBy(0,800)");
}
```

高亮显示正在被操作的页面元素

```
public void highlightElement(WebElement element){
    JavascriptExecutor js = (JavascriptExecutor) driver;
    //使用JavascriptExecutor将传入参数的页面对象的背景颜色和边框颜色分别定为黄色和红色
    js.executeScript("arguments[0].setAttribute('style',arguments[1]);",
            element,"backgroud:yellow;border:2px solid red;");

}
@Test
public void testHighlight() throws InterruptedException{
    WebElement inputSearch = driver.findElement(By.id("kw"));
    highlightElement(inputSearch);
    Thread.sleep(3000);
}
```

设置页面对象的属性值

```
public void setAttribute(WebElement element, String attributeName,
        String value) {
    js.executeScript("arguments[0].setAttribute(arguments[1]," +
        "arguments[2])", element, attributeName, value);
}

@Test
public void testSetAttibute() {
    wd.get("http://localhost:8032/test/07.html");
    this.setAttribute(wd.findElement(By.id("uid")), "size", "10");
    this.setAttribute(wd.findElement(By.id("uid")), "value", "星期2");
}
```

可使用removeAttribute删除页面元素的属性

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

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

失败截屏，文件名为当前时间

```
public void screenShot(String imageName) {
    File source_file = ((TakesScreenshot) driver)
            .getScreenshotAs(OutputType.FILE);
    try {
        FileUtils.copyFile(source_file, new File("D://demo//" + imageName
                + ".jpg"));
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

@Test
public void testscreenShot() throws IOException {
    this.driver.get("https://www.baidu.com/");
    SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd-HHmmss");
    String fileName = df.format(new Date());
    screenShot(fileName);
}
```

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

强制等待

- Thread.*sleep*（3000）

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

- 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。

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)

```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();
    }
```

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

- 需要使用JavaScript语句调用HTML5对象提供的内部变量和函数来实现各类操作，可以参考相关HTML5的接口文档来实现各种复杂的测试操作， HTML5的JS接口文档网址为http://html5index.org/

HTML5中的视频播放器

- http://www.w3school.com.cn/tiy/t.asp?f=html5_video

```
@Test
public void testVideo() throws InterruptedException {
    driver.get("http://videojs.com/");
    WebElement video = driver.findElement(By.tagName("video"));
    JavascriptExecutor js = (JavascriptExecutor) driver;
    //获取视频的网络存储地址
    String source = (String) js.executeScript(
            "return arguments[0].currentSrc;", video);
    System.out.println(source);
    assertEquals(source, "http://vjs.zencdn.net/v/oceans.mp4");
    //获取播放时长
    Double videoDuration = (Double) js.executeScript(
            "return arguments[0].duration", video);
    System.out.println(videoDuration.doubleValue());
    //播放
    js.executeScript("return arguments[0].play()", video);
    Thread.sleep(2000);
    //暂停
    js.executeScript("return arguments[0].pause()", video);
}
```

HTML5中的绘画操作

- moveToElement(toElement,xOffset,yOffset)

// 以鼠标当前位置或者 (0,0) 为中心开始移动到 (xOffset, yOffset) 坐标轴

- 测试地址http://literallycanvas.com/

```
@Test
public void testCanvas1() throws InterruptedException {
    driver.get("http://literallycanvas.com/");
    WebElement canvas = driver.findElement(By
            .xpath("//*[@id='literally-canvas']/div[1]/div[1]/canvas[2]"));
    Actions drawing = new Actions(driver);
    drawing.clickAndHold(canvas).moveByOffset(10, 50).moveByOffset(50, 10)
            .moveByOffset(-10, -50).moveByOffset(-50, -10).release()
            .perform();
    Thread.sleep(5000);
    screenShot("test12.jpg");
}
```

HTML5中的存储对象

- http://www.w3school.com.cn/html5/html_5_webstorage.asp

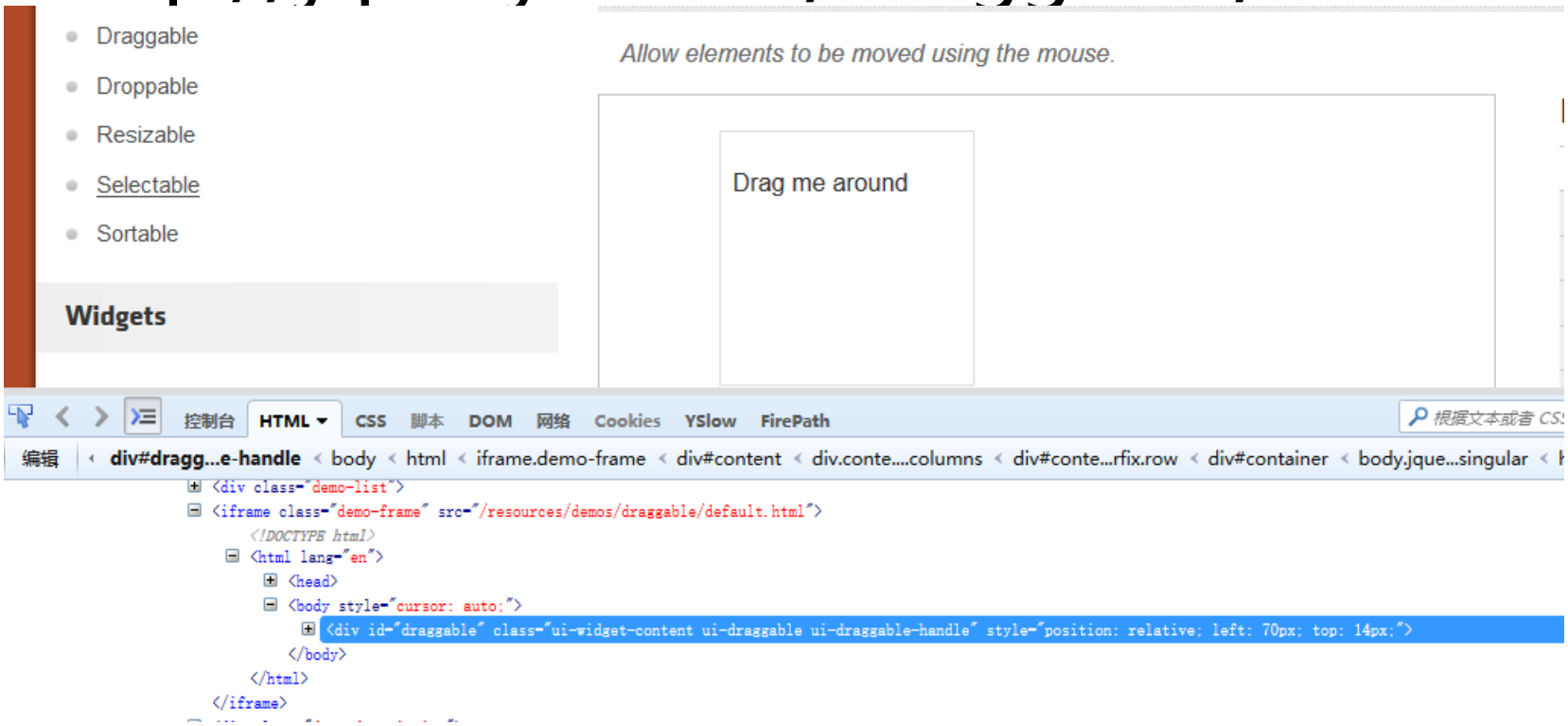
```
@Test
public void testLocalStorage() throws InterruptedException {
    driver.get("http://www.w3school.com.cn/tiy/loadtext.asp?f=html5_webstorage_local");
    String name1 = executeJs("return localStorage.lastname");
    System.out.println(name1);
    assertEquals(name1, "Smith");
    executeJs("localStorage.clear();");
    String name2 = executeJs("return localStorage.lastname");
    System.out.println(name2);

}

@Test
public void testSessionStorage() throws InterruptedException {
    driver.get("http://www.w3school.com.cn/tiy/loadtext.asp?f=html5_webstorage_session");
    String name1 = executeJs("return  sessionStorage.lastname");
    assertEquals(name1, "Smith");
    executeJs("sessionStorage.removeItem('lastname');");
    executeJs("sessionStorage.clear();");
}
```

Drag和Drop

- 使用Actions通过来拖动和移动元素位置
- 测试地址：
  http://jqueryui.com/draggable/



```java
@Test
public void testDrag() throws InterruptedException {
    driver.get("http://jqueryui.com/draggable/");
    WebElement frame1 = driver.findElement(By.className("demo-frame"));
    driver.switchTo().frame(frame1);
    WebElement draggable = driver.findElement(By
            .xpath("//div[@id='draggable']"));
    new Actions(driver).dragAndDropBy(draggable, 200, 100).build()
            .perform();
    Thread.sleep(5000);
}
```

本章大纲

5.1 获取测试对象状态

5.2 执行js操作

5.3 窗口截屏

5.4 Webdriver的三种等待方式

5.5 H5元素

5.6 代码检查点

代码检查点

- 检查方式 元素是否正常 try catch
- 验证页面是否存在某文字
- 验证页面是否存在某元素
- 验证某一个元素是否包含某些文字
- 验证元素的属性title
- 验证某个元素对应的value必须是某值

富文本框

- 富文本框的实现常见技术用到了Frame标签，并且在Frame中实现了一个常见的HTML网页结构，所以使用普通定位方式无法获取
- 方法1：使用切换子页面语句实现富文本框的输入

  driver.switchTo().frame(“iframe”);
- 方法2：找到紧邻的元素，使用Tab键的方法

方法1：wd.switchTo().frame("xhe0_iframe");
wd.findElement(By.*tagName("body")).sendKeys("这个电影值得一看");*
    Thread.*sleep(5000);*
wd.switchTo().defaultContent();
方法2：
Actions action =**new Actions(wd);**
action.sendKeys(Keys.***TAB);***
action.sendKeys(Keys.***TAB);***
action.sendKeys(Keys.***TAB);***
action.sendKeys("hello").perform();
action.sendKeys(Keys.***TAB);***
action.sendKeys(Keys.***ENTER).perform();***