

Appium

<http://appium.io/>

<https://github.com/appium/appium/tree/master/docs/cn>

<https://github.com/appium/appium/tree/master/sample-code>

https://static.javadoc.io/io.appium/java-client/7.0.0/io/appium/java_client/touch/package-summary.html

https://github.com/appium/java-client/tree/master/src/test/java/io/appium/java_client/android

本章大纲

- Appium介绍
- 官方实例
- Appium元素定位方式
- Appium在项目中的应用

Appium介绍

- Appium是一个**开源、跨平台**的测试框架，可以用来测试原生，移动Web及混合的移动端应用，支持ios， android平台的测试。
- C/S架构，基于**WebDriver JSONWireProtocol**协议统一接口，来驱动Apple系统的UIAutomation库，Android 系统的**UIAutomator**框架。
- <https://github.com/appium/appium>

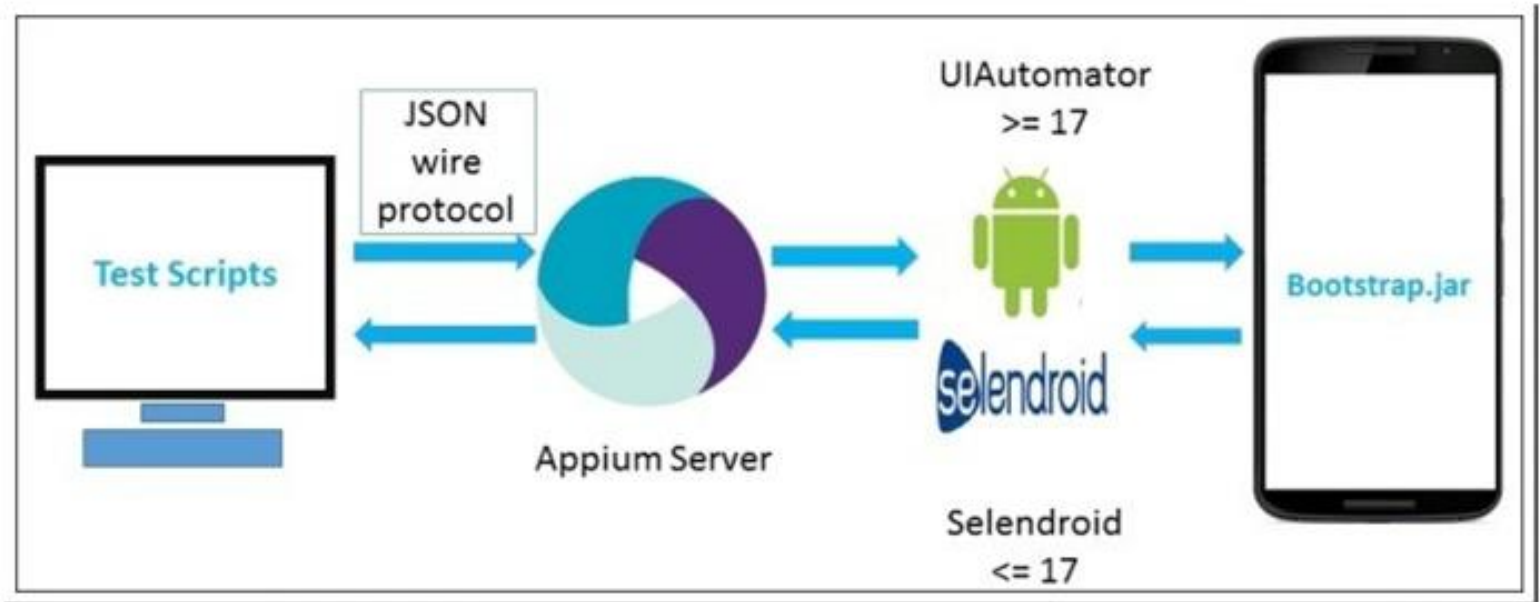
Appium设计理念

- 不需要为了自动化，而且重新编译或修改测试app
- 不必局限于某种语言或者框架来编写和运行测试脚本的运行
- 一个移动自动化的框架不应该在接口上重复造轮子
- 移动端自动化测试应该是开源的

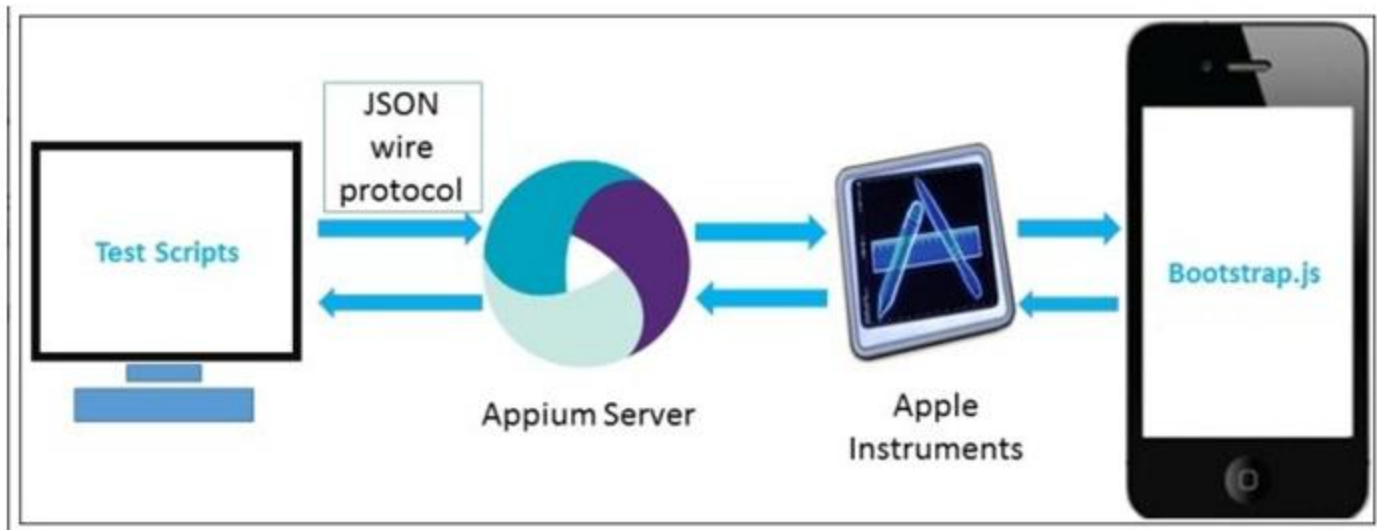
Appium的特点

- 跨架构: native 、 hybrid 、 webview
- 跨设备: android 、 ios 、 firefox os
- 跨语言: java 、 python、 ruby 、 nodejs 、 php
- 跨app: 可以在多个app交互
- 不依赖源代码
- 不限制测试框架和平台

Appium架构原理



Appium架构原理



Appium架构原理

(1) Appium服务器。Appium服务器是Appium框架的核心。它是一个基于Node.js实现的HTTP服务器。Appium服务器的主要功能是接受从Appium客户端发起的连接，监听从客户端发送来的命令，将命令发送给bootstrap.jar（iOS手机为bootstrap.js）执行，并将命令的执行结果通过HTTP应答反馈给Appium客户端。

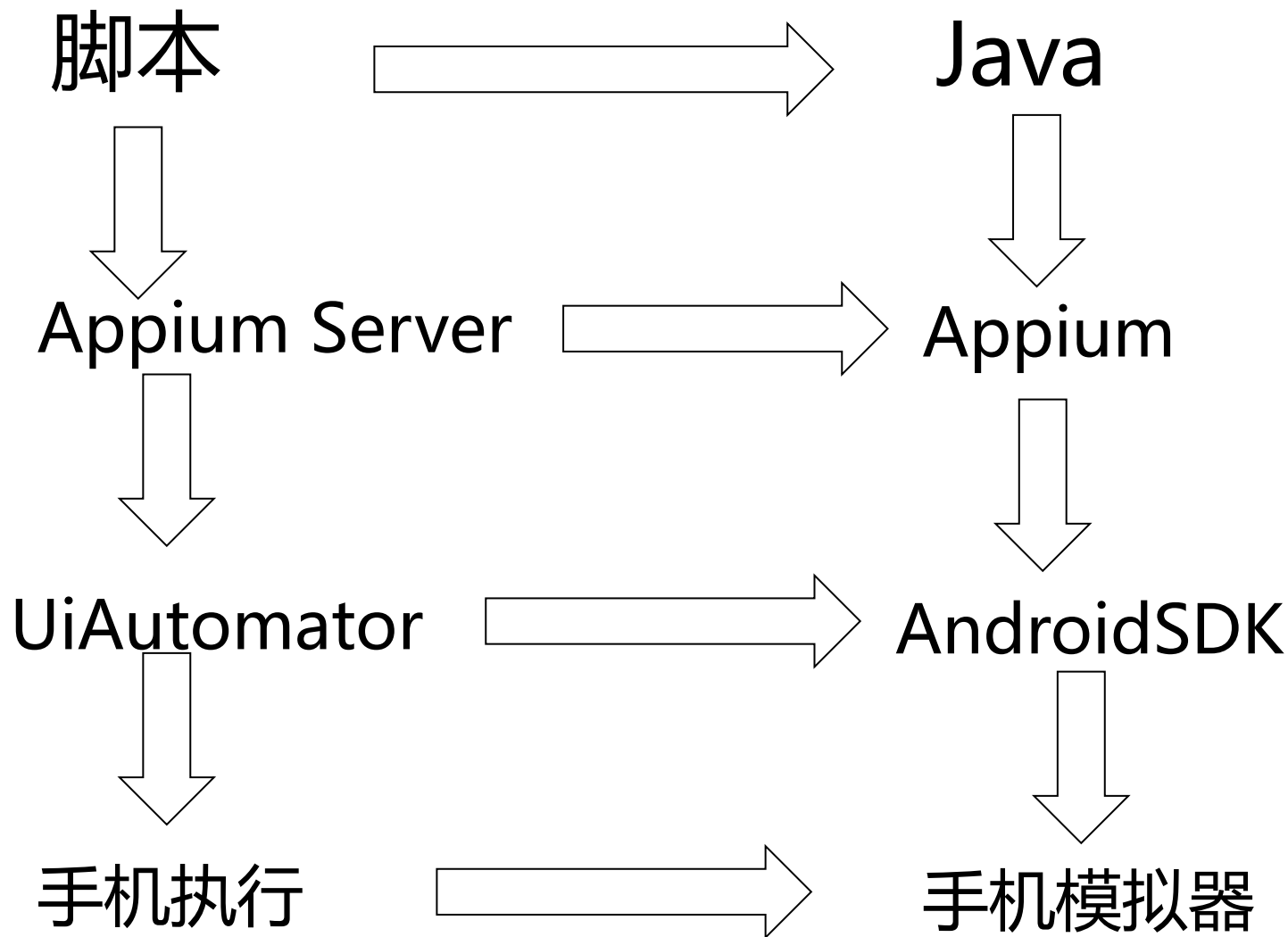
(2) Bootstrap.jar。Bootstrap.jar是在Android手机上运行的一个应用程序，它在手机上扮演TCP服务器的角色。当Appium服务器需要运行命令时，Appium服务器会与Bootstrap.jar建立TCP通信，并把命令发送给Bootstrap.jar；Bootstrap.jar负责运行测试命令。

(3) Appium客户端。它主要是指实现了Appium功能的WebDriver协议的客户端Library，它负责与Appium服务器建立连接，并将测试脚本的指令发送到Appium服务器。现有的客户端Library有多种语言的实现，包括Ruby、Python、Java、JavaScript（Node.js）、Object C、PHP和C#。Appium的测试是在这些Library的基础上进行开发的。

Appium的技术架构

- OS: Apple' s UIAutomation
- Android 2.3+: Google' s Instrumentation
- Android 4.2+: Google' s UIAutomator

安装与配置



安装与配置

1. Android Studio 安装

2. JDK8:

D:\Users\think\AppData\Local\Android\sdk\platform-tools 安装

3. selenium: java-client-7.0.0.jar和selenium-server-standalone-3.141.59.jar

官网下载地址: <http://docs.seleniumhq.org/download/>

4. Appium:

<https://github.com/appium/appium-desktop/releases>

5. 模拟器

本章大纲

- Appium介绍
- 官方实例
- Appium元素定位方式
- Appium在项目中的应用

使用步骤

- 创建项目
- 创建module
- 创建文件夹
(libs, app

```
public class AndroidCreateSessionTest1 {  
    private AndroidDriver<WebElement> driver;  
    @BeforeClass  
    public void setUp() throws Exception {  
        File classpathRoot = new File(System.getProperty("user.dir"));  
        File appDir = new File(classpathRoot, s: "../appiumdemo/apps");  
        File app = new File(appDir.getCanonicalPath(), s1: "ApiDemos-debug.apk");  
        DesiredCapabilities capabilities = new DesiredCapabilities();  
        capabilities.setCapability(capabilityName: "deviceName", value: "Android Emulator");  
        capabilities.setCapability(capabilityName: "app", app.getAbsolutePath());  
        capabilities.setCapability(capabilityName: "appPackage", value: "io.appium.android.apis");  
        capabilities.setCapability(capabilityName: "appActivity", value: ".ApiDemos");  
        driver = new AndroidDriver<WebElement>(new URL(s: "http://127.0.0.1:4723/wd/hub"), capabilities);  
    }  
  
    @AfterClass  
    public void tearDown() {  
        driver.quit();  
    }  
  
    @Test()  
    public void testCreateSession() {  
        String activity = driver.currentActivity();  
        String pkg = driver.getCurrentPackage();  
        Assert.assertEquals(activity, expected: ".ApiDemos");  
        Assert.assertEquals(pkg, expected: "io.appium.android.apis");  
    }  
}
```

DesiredCapabilities

DesiredCapability是一个JSON对象，包含一组key和value值。它由客户端发送给服务端，告诉服务端期望的Capabilities（可以理解为一种能力）有哪些，然后服务端根据这些capabilities创建自动化会话（session）。

- 本次测试是启动浏览器还是启动移动设备？
- 是启动android还是启动ios？
- 启动android时，app的package是什么？
- 启动android时，app的activity是什么？

<https://github.com/appium/appium/blob/master/docs/cn/writing-running-appium/caps.md>

Desired Capabilities

- 常用参数

参数	用途
automationName	appium（默认）
deviceName	测试的设备名称
platformVersion	平台版本
appPackage	待测试的app的java package
appActivity	待测试的app的Activity名字
app	应用的绝对路径，注意一定是绝对路径。如果指定了appPackage和appActivity的话，这个属性是可以不设置的
platformName	IOS/Android/FirefoxOS
noReset	true
unicodeKeyboard	支持中文输入

查看包名与Activity

- 启动待测apk
- 开启日志输出: `adb logcat>findstr START`
- 或者 `aapt dump badging d:/todolist.apk`

本章大纲

- Appium介绍
- 官方实例
- Appium元素定位方式
- Appium在项目中的应用

APPIUM定位原生应用元素

- `findElement(By)`
- `findElementsById`
- `findElementByClassName`
- `findElementByAccessibilityId`
- `findElementByTagName`
- `findElementByXPath`

APPIUM定位原生应用元素

- resource-id表示 ID 查找By.id()
- text表示文本内容 查找By.name()
- class 查找用By.className() 由于app一个页面多数情况下会出现多个相同的className所以app测试一般不用。
- content-desc注释，在查找元素时与text一样都用By.name() 或用By.AccessibilityId()
- xpath查找用By.xpath，不常用

手机中定位一般用name就可以了。手机app页面简单，一般情况下不会出现相同的名称

通过ID定位元素

WebElement element

```
=driver.findElement(By.id( "com.example.android.contactmanager:id/addContactButton" ));
```

或者可以这样写：

```
driver.findElementById("com.example.android.contactmanager:id/addContactButton");
```

index	3
text	Add Contact
resource-id	com.example.android.contactmanager:id/addContactButton
class	android.widget.Button
package	com.example.android.contactmanager
content-desc	Add Contact
...	...

通过NAME定位元素

```
WebElement el=  
driver.findElement(By.name("Add Contact"));  
//或者这样  
WebElement el = driver.findElementByName("Add  
Contact");
```

这里的text可以认为是name，那么代码就是这样写的：

Node Detail	
index	3
text	Add Contact
resource-id	com.example.android.contactmanager:id/addContactButton
class	android.widget.Button
package	com.example.android.contactmanager
content-desc	Add Contact
checkable	false

通过ClassName定位元素

// 通过classname查找元素WebElement el=

```
driver.findElement(By.className("android.widget.Button"));
```

// 或者这样使用

WebElement el =

```
driver.findElementByClassName("android.widget.Button");
```

Node Detail	
index	3
text	Add Contact
resource-id	com.example.android.contactmanager:id/addContactButton
class	android.widget.Button
package	com.example.android.contactmanager
content-desc	Add Contact

通过AccessibilityId定位元素

对应“content-desc”的属性值

// 通过AccessibilityId查找元素

```
WebElement el= driver.findElementByAccessibilityId("Add  
Contact");
```

Node Detail	
index	3
text	Add Contact
resource-id	com.example.android.contactmanager:id/addContactButton
class	android.widget.Button
package	com.example.android.contactmanager
content-desc	Add Contact
checkable	false

通过Xpath定位

Appium对于xpath定位执行效率是比较低的，也就是说遇到xpath的定位代码的时候执行比较慢。迫不得已的情况下**尽量不用**这个定位方式。

// 通过xpath查找元素

```
WebElement el=
```

```
driver.findElement(By.xpath("//android.widget.LinearLayout/android.widget.FrameLayout[2]/android.widget.LinearLayout/android.widget.Button"));
```

//或者用

```
WebElement el = driver.findElement(By.xpath("//*[@text=' Add Contact' ]"));
```

//或者用

```
WebElement el = driver.findElementByXPath("//*[@text=' Add Contact' ]");
```


控件的操作方法

Appium的辅助类TouchAction，主要针对手势操作，比如滑动、长按、拖动等。

长按

WebElement

```
node=driver.findElementById("toDoItemDetailTv");  
TouchAction dragNDrop = new TouchAction(driver)  
    .longPress(element(node)).perform();
```

滑动

```
private final ActionSupplier<AndroidTouchAction> verticalSwiping = ()->  
    new AndroidTouchAction(driver)  
        .press(element(driver.findElementByAccessibilityId("Gallery")))  
        .waitAction(waitOptions(ofSeconds(2)))  
        .moveTo(element(driver.findElementByAccessibilityId("Auto  
Complete")))  
        .release();
```

控件的操作方法

Appium的辅助类TouchAction，主要针对手势操作，比如滑动、长按、拖动等。

拖动

```
TouchAction dragNDrop = new TouchAction(driver)  
    .longPress(element(dragDot1))  
    .moveTo(element(dragDot3))  
    .release();  
dragNDrop.perform();
```

实例代码解释

1、`driver.resetApp();` 重置App (先close再launch)

2、打开指定的Activity

```
Activity activity = new Activity("io.appium.android.apis",  
".view.DragAndDropDemo");
```

```
driver.startActivity(activity);
```

3、`driver.getContext()`

获取当前上下文，返回值含**NATIVE_APP**则为原生控件；返回值含**WEBVIEW**则为web控件。

Appium处理纯web应用元素定位

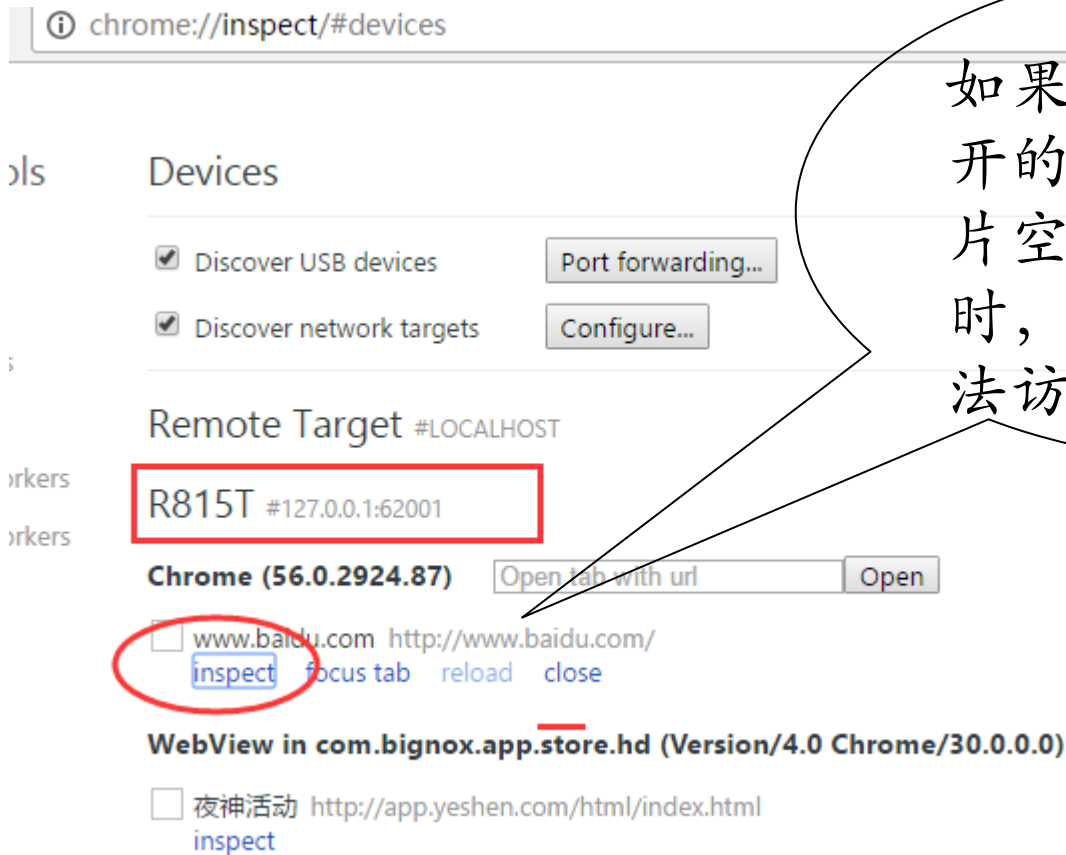
在PC上的Chrome浏览器中安装ADB插件来捕获页面元素



Appium处理纯web应用元素定位


在PC机上打开chrome浏览器，然后在地址栏输入：

chrome://inspect/#devices



如果你点击inspect打开的DevTools窗口一片空白，且刷新无效时，那极有可能是无法访问谷歌服务器

Appium处理纯web应用元素定位



Developer Tools - <https://m.baidu.com/?from=844b&vit=fps>

← → ↻ <https://m.baidu.com/?from=844b&vit=fps>

百度一下

input#index-kw-se-input 424px x 45px

关注 新闻 小说 视频 图片 地图

基层代表热议精准扶贫精准预防让纪检监察触角延伸到每一个贫困户

轿车起火烧死2岁女孩 父亲跪地痛哭(图)

代女相亲三年 天河公园挂上万份资

男女相亲要注意什么

都柏林举行三棋比赛 驻爱大使颁奖支持国粹传播

新浪体育 10:09

Elements Console >>

```
<form data-formposition="i"
class="se-form focus" id="index-
form" action="/from=844b/s"
method="get" autocomplete="off">
  <div class="con-wrap">
    ...
    <input type="text"
      autocomplete="off"
      autocorrect="off"
      maxlength="64" id="index-
      kw" name="word" class="se-
      input" data-sug="1" == $@
    >
    <div class="se-inner">
      ...
    </div>
  </div>
  <div id="index-box">...</div>
  <input type="hidden" name=
  ...>
</form>
```

... #index-form div input#index-kw.se-input

Styles Event Listeners DOM Breakpoints >>

:hov .cls +

element.style {

}

?from=844b...
#index-card
.se-input {

border-radius: 0px;

border-radius: 0px;

margin: 0px;

padding: 0px;

margin: -6px 42px 7px 31px;

padding: 7px 31px 7px 31px;

Filter Show all

- ▶ background-a... scroll
- ▶ background-c... border-...
- ▶ background-c... rgba(0...
- ▶ background-i... none
- ▶ background-o... padding...

Appium处理纯web应用元素定位

```
public class AndroidCreateWebSessionTest1 {  
    private AndroidDriver<WebElement> driver;  
    @BeforeClass  
    public void setUp() throws MalformedURLException {  
        DesiredCapabilities capabilities = new DesiredCapabilities();  
        capabilities.setCapability(capabilityName: "deviceName", value: "Android Emulator");  
        capabilities.setCapability(capabilityName: "browserName", value: "Chrome");  
        driver = new AndroidDriver<WebElement>(new URL(s: "http://127.0.0.1:4723/wd/hub"), capabilities);  
    }  
    @AfterClass  
    public void tearDown() { driver.quit(); }  
    @Test  
    public void testCreateWebSession() throws URISyntaxException {  
        driver.get(new URI(s: "http://www.google.com").toString());  
        String title = driver.getTitle();  
        Assert.assertEquals(title, expected: "Google");  
    }  
}
```

Appium处理纯web应用元素定位

注意:

`\Appium\node_modules\appium\node_modules\appium-chromedriver\chromedriver\win下`

chromedriver的版本是否支持当前的被测浏览器版本

Appium定位混合应用元素

混合应用是原生APP+webview组成的，可以简单的理解为一个原生app的外壳，内部全是html页面。在处理这样的app的定位的时候，需要先定位原生APP上的按钮或者链接，使用Context切换，然后点击按钮或者链接，然后经过appium提供的方法，进入webview页面，最后像纯web应用元素定位可以提供的定位工具和方法进行元素定位了。

Appium定位混合应用元素

使用`driver.getContextHandles()`; 获取app的所有handles, 原生应用会有一个**NATIVE_APP** 的handle, webview也会有一个WEBVIEW_XXX_XXX_XX的handle, 确定了webview的handle之后, 使用: `driver.context(handle的字符串)`; 进入webview页面。

```
Set<String> contexts= driver.getContextHandles();
for(String context: contexts) {
    System.out.println(context);
}
driver.context((String) contexts.toArray()[1]);
WebElement inputBox=driver.findElement(By.id("index-kw"));
inputBox.sendKeys("JAVA");
```

显示等待

显示等待就是等待下一个需要定位的元素，元素出现就停止等待，开始执行代码。

如：等待10秒，每两秒检查一次是否成功

```
WebDriverWait wait = new WebDriverWait(driver, 10, 2000);
wait.until(new ExpectedCondition<WebElement>() {
    @Override
    public WebElement apply(WebDriver input) {
        return driver.findElement(By.name("ADD"));
    }
}).click();
```

隐式等待

隐式等待是等待页面加载时间，设置一个超时时间，如果在设置的时间内页面加载完成就立即继续执行。如果一直不能完全加载，规定时间到后会报出异常。此处设置是全局设置。

- `pageLoadTimeout`是等待页面加载
- `implicitlyWait`是识别对象超时时间，如果在规定的时间内无法定位元素就报错
- `setScriptTimeout`设置异步脚本超时时间，也就是有`async`属性的JS脚本

```
driver.manage().timeouts().pageLoadTimeout(1000, TimeUnit.SECONDS);  
driver.manage().timeouts().implicitlyWait(1000, TimeUnit.SECONDS);
```

本章大纲

- Appium介绍
- 官方实例
- Appium元素定位方式
- Appium在项目中的应用

测试脚本设计思想

第一，被测试程序**主要变化的地方**是什么，是否适合用UI自动化测试。如果应用程序UI变化概率比较小，代码变动主要是下层逻辑，这样的程序比较适合做UI自动化测试。如果UI变化大，那么UI自动化脚本维护成本就会很大，自动化测试投入产出比不高。

第二，被测试的**程序是什么类型**的应用。比如游戏类的测试，可能很多的画面都是通过OpenGL直接渲染的，Appium无法找到OpenGL直接渲染出来的画面里的元素，而且从UI上去验证游戏画面非常困难，在这种情况下，如果通过UI实施自动化测试可能需要大量的后期人工检查。

测试脚本设计思想

第三，自动化测试的目标是什么，是否对测试的运行时间有要求。如果自动化的目标是快速地回归，要求测试脚本短时间内完成大批脚本的运行的话，此时可能不适合用Appium。

第四，自动化测试是否要脱机执行。比如性能测试中的耗电量测试，必须断开与电脑的连接，否则USB线会给手机充电。由于Appium是必须与电脑连接的，以上的场景就不能通过Appium来实施自动化，可以考虑选择UIAutomator。

第五，如果选择Appium来实施自动化测试，什么语言比较合适。

框架设计

base里面有个BaseTest. java, 这个类的主要作用是启动和关闭APP的作用

pages存放页面元素类, 每一个java类, 都是一个页面, 存放的都是对应页面的所有元素及操作

testcases存放测试用例的地方, 在这个包下, 还会有很多子包, 子包的个数根据测试的系统的模块来划分, 比如有登录模块, 首页模块等, 那么子包的名字就应该写成login、home

utils封装了各种工具类, 包括读取excel, appium api封装类, 读取数据库类, 读取属性文件类和生成driver的类等。