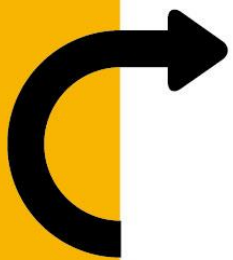


自动化测试基础 (PDF测试岗位课程)





第一部分 自动化测试概述

第二部分 Selenium2.0基础介绍

第三部分 Appium基础介绍

第四部分 SAT使用介绍



手工测试优点及不足

■ 手工测试

优点：

- 具有创造性，能通过探索性的方法来发现软件隐藏的漏洞和缺陷；
- 高能力的人能够保证高质量；

不足：

- 不稳定性，执行结果与人的能力、心态、情绪等因素关联较大，结果难以通过具体的指标衡量；
- 人的成本高；



自动化测试的定义

- 自动化测试是软件测试的一个重要组成部分

把以人为驱动测试行为转化为机器执行的一种过程

意图替代部分人工测试，提高核心价值。





自动化测试的目的

- 缩短测试周期，加快测试进度，从而加快产品发布进度
- 实现更大规模、更大频率的测试、提高测试覆盖率
- 降低测试成本、持久化测试成果
- 减少重复性枯燥测试工作，提高团队士气
- 保证回归测试的可控性和一致性
- 辅助测试人员完成手工无法完成的测试





自动化测试的优点

• 优点

- 自动化测试可以提高测试**效率**，使测试人员更加专注于新的测试模块的建立和开发，从而提高测试覆盖率;
- 其自动化测试更便于测试资产的数字化管理，使得测试资产在整个测试生命周期内可以得到**复用**，这个特点在功能测试和回归测试中尤其具有意义;
- 测试流程自动化管理可以使机构的测试活动开展更加**过程化**，这很符合CMMI过程改进的思想。



自动化测试的不足

• 不足

- 自动化测试**不会比手工测试发现的缺陷多**。工具做什么、怎么做都是由人事先预定义好了的。自动化测试并不会智能的帮助人现更多潜在的问题。
- 自动化测试在产品频繁变更的情况下**维护代价**会很高。
- 自动化测试对环境的**依赖性**远远大于手工测试。



自动化测试的误区

- 期望自动化测试能够完全取代手工测试
- 期望自动化测试发现大量的新缺陷
- 期望自动化测试能够智能的完成绝大多数工作
- 期望自动化测试是一劳永逸的



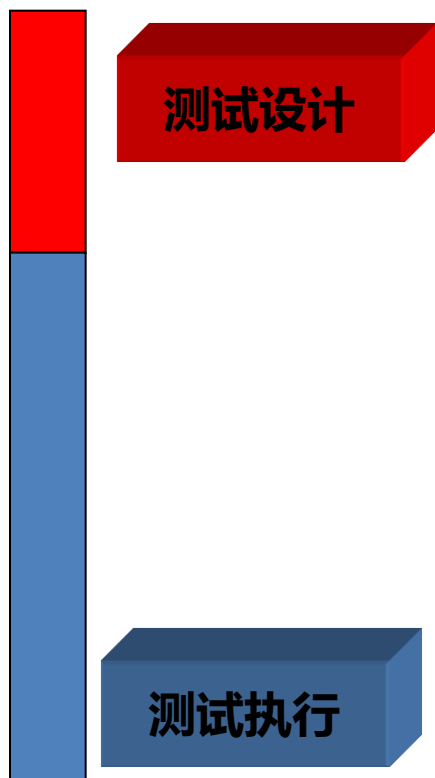
自动化测试不是万能的，它只是测试人员工具箱里的一件利器，它无法取代测试工程师的地位。





手工测试和自动化测试

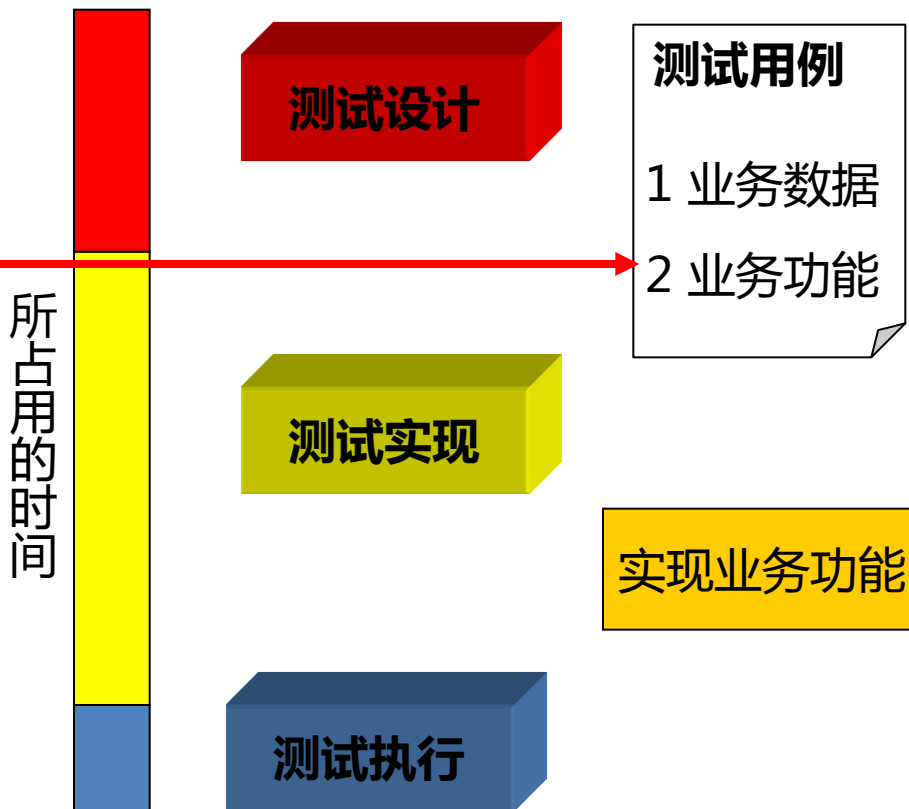
• 手工测试



测试用例

- 1 输入数据
- 2 操作
- 3 预期结果

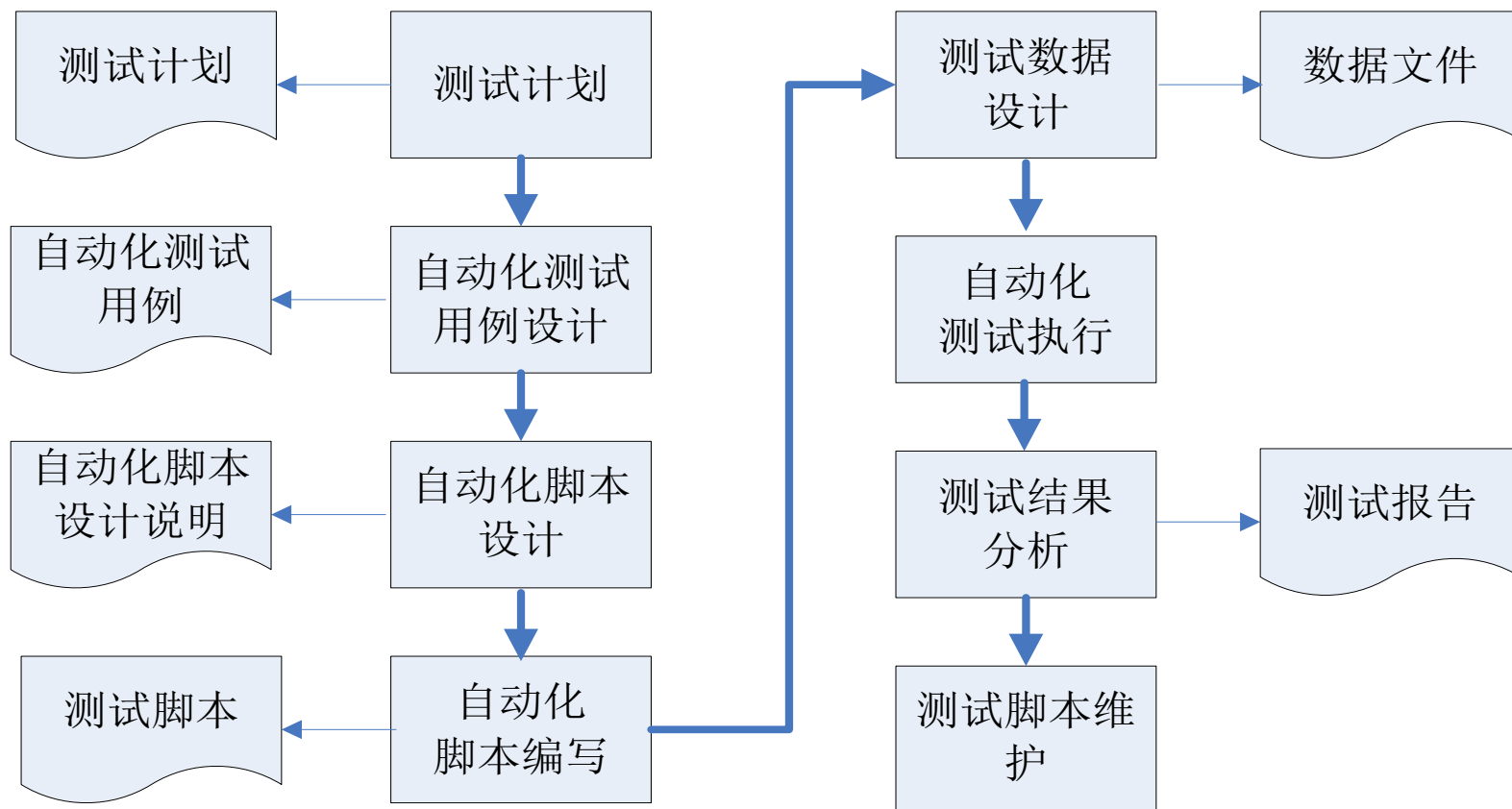
• 自动化测试





(功能)自动化测试基本流程

• 流程活动图





(功能)自动化测试基本流程

● 主要流程说明

1. 测试计划

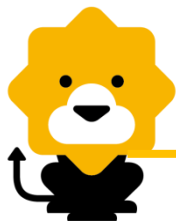
与以前的测试计划过程一致，只是在原来的测试计划中，添加对项目实施自动化测试所需的资源、测试范围、测试进度的描述。该过程产出物为《测试计划》。

2. 自动化测试用例设计

根据《测试计划》、《软件需求规格说明书》、《系统测试用例》设计出针对自动化测试的测试用例。测试用例的粒度精确到单个功能点或流程，对于各个功能点的业务规则，通过对脚本添加相应的检查点来进行测试。过程产出《自动化测试用例》。

一般来讲，基于Web功能测试需要覆盖以下几个方面：

- 1) 页面链接测试，确保各个链接正常；
- 2) 页面控件测试，确保各个控件可靠；
- 3) 页面功能测试，确保各项操作正常；
- 4) 数据处理测试，确保数据显示准确、处理精确可靠；
- 5) 模块业务逻辑测试，确保各个业务流程畅通；



(功能)自动化测试基本流程

3. 自动化脚本设计

根据《软件需求规格说明书》、《自动化测试用例》、《系统原型》、《系统设计说明书》编写《自动化脚本设计说明书》，其主要内容包括：分析当前项目，设计出适合的脚本基本架构，针对特殊自动化测试用例设计可行的脚本编写方法，设计特殊检查点的实现方式，并对潜在的技术难点提出解决方案。该过程的产出物是《自动化脚本设计说明书》。

4. 自动化脚本编写

根据《软件需求规格说明书》、《自动化测试用例》、《系统原型》、《自动化脚本设计说明书》，录制、调试、编写各个功能点的自动化测试脚本，并添加检查点，进行参数化。该过程还需要编写**数据文件处理脚本、日志文件处理脚本、数据库处理脚本、公共检查点处理脚本**等等。该过程的产出物是各个功能点的自动化测试脚本和其他公共处理脚本。

PS：自动化测试和手工测试都是测试范畴，思想基本一致，只是表现形式不同，比如自动化用例需要转为自动化脚本。

5. 自动化测试数据设计

根据《软件需求规格说明书》、《自动化测试用例》设计出对各个功能点和相关业务规则进行测试的输入数据和预期输出，填写入对应的数据文件中。该过程的产出物是各个功能点的数据文件。



(功能)自动化测试基本流程

6. 自动化测试执行

搭建环境是保证测试工作正常开展的一项重要工作。

搭建好测试环境，根据《自动化测试用例》，执行自动化脚本，对系统进行自动化测试，并自动记录测试结果到日志文件中。

7. 自动化测试结果分析

自动化执行完成后，需要对测试结果文件中报告错误的记录进行比较、分析，如果确实是由于被测系统的缺陷导致，则提交缺陷报告。对自动化测试的结果进行总结，分析系统存在的问题，提交《测试报告》。

8. 自动化测试脚本维护

如果系统发生变更时，对自动化测试脚本和相关文档包括《自动化测试用例》、《自动化脚本设计说明书》进行维护，以适应变更后的系统。



自动化测试基本流程-用例设计

更适合手工测试，而不适合自动化测试：

- 一个新开发的不稳定功能。
- 一个不够稳定的测试用例。
- 一个需要人来判断和干涉的测试用例。
- 测试结果很难准确预测的测试用例。



自动化测试基本流程-用例设计

而另一些场合则更适合自动化测试，不适合手工测试：

- 运行时间很长的测试用例。
- 运行重复次数较多的测试用例。
- 每轮测试都将进行测试的测试用例。
- 高度冗余的任务或场景。
- 乏味且人工容易出错的工作。

总结：用例实现自动化，应优先考虑高风险、低复杂度、能尽快达到投资回报的测试用例。



自动化测试基本流程-脚本开发1

自动化脚本开发按照发展，从最初到现在大概分为以下几种：

■ 线性脚本的编写

说明：脚本为顺序的简单录制和回放，即基于录制和回放的自动化测试

特点：非常低的脚本开发成本，要求代码能力较低，不需要计划和设计；

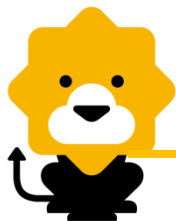
测试数据在脚本中，维护成本较高。

■ 结构化脚本的编写

说明：在编写的脚本中使用结构控制，基于开发语言或者是脚本语言来编写测试脚本

特点：比线性开发脚本成本较高，要求有一定的代码能力，需要简单的计划和设计；

测试数据在脚本中，维护成本相对低一些，但依赖编写者的编码能力。



自动化测试基本流程-脚本开发2

自动化脚本开发按照发展，从最初到现在大概分为以下几种：

■ 共享脚本的编写

说明：把代表应用程序行为的脚本在其他脚本之间共享

特点：比结构开发脚本成本较高，要具有调整代码的编程技巧，需要计划和设计；
测试数据也是硬编码的，维护成本比线性脚本编写要低一些。

■ 数据驱动的编写

说明：把测试数据从脚本中分离出去，存储在外部的文件中

特点：需要脚本参数化和编程成本比共享的编写要高一些，要具有较高的调整代码编程技巧，需要更多的计划和设计；
测试数据独立存储在数据表或者外部文件，维护成本较低。



自动化测试基本流程-脚本开发3

自动化脚本开发按照发展，从最初到现在大概分为以下几种：

■ 关键字驱动脚本的编写

说明：把检查点和执行操作的控制都维护在外部数据文件(**我们使用的脚本方式**)

原理：1. 关键字驱动测试是数据驱动测试的一种改进类型

2. 主要关键字包括三类：被操作对象 (Item)、操作 (Operation) 和值 (value)

用面向对象形式可将其表现为Item.Operation(Value)

3. 将测试逻辑按照这些关键字进行分解，形成数据文件

4. 用关键字的形式将测试逻辑封装在数据文件中，测试工具只要能够解释这些

关键字即可对其应用自动化

特点：脚本开发成本高，要求要有很强的编程能力，最初的计划和设计及管理成本很高；

测试数据在外部文件，维护成本较低



自动化测试基本流程-脚本开发4

- 高质量的自动化脚本具有的特点
 - 可重用性
 - 稳定性
 - 可扩展性
 - 可维护性





自动化测试基本流程-脚本开发5

■ 用例转为脚本时的注意事项

1. 用例之间不要有关联性，自动化测试开发同样是软件开发工程，脚本编写同样提倡高内聚低耦合的理念
2. 当前的测试用例前置配置信息要写清楚，但最终配置信息回归至原点
3. 每一个操作步骤和验证点要具体，避免脚本写一堆代码去验证
4. 尽量避免在同一个地方做重复的验证
5. 案例步骤需衔接精确，避免脚本不必要的异常，保证精确。



常用Web页面自动化工具



Selenium是Thoughtworks公司的开源项目，为Web应用程序编写的一个验收测试工具。1.x版本基于JavaScript注入方式驱动浏览器，2.0采用Web Driver方式驱动浏览器，社区活跃度较高，工具更新较快。支持Java，c#，.NET，Ruby，Python等多种语言。支持IE，FireFox，Chrome，Safari等浏览器，但仅可在firefox下录制



Wati*是一个系列工具，包含WatiR，WatiJ，WatiN。基于IE扩展dll驱动浏览器，仅支持windows系统下的IE浏览器，其对应的工具R支持Ruby，J支持Java，N支持.NET。资料比较齐全，但工具更新较慢。



Sahi是Tyto Software公司提供的一款Web界面自动化工具，其原理和Selenium1.x版本的实现一致，基于JavaScript注入方式，使用必须开启代理服务器来规避JS的同源策略，局限性较多。



常用Web页面自动化工具



QTP是Mercury公司开发的一款商用工具，专门用于界面自动化测试，提供录制能力，可以使用VBScript语言来编写脚本，提供控件查看等辅助工具。



RFT是IBM公司Rational系列产品之一，支持HTML，AWT，SWT，.NET，Siebel，Flex等应用的界面自动化测试。提供录制能力，采用Java语言编写脚本，支持控件动态查找，控件高亮显示等辅助功能。



常用Web页面自动化工具



SAT是苏宁自主研发的自动化测试工具，支持WEB页面、Http/MQ协议、移动终端等自动化。以selenium2.0为基础封装关键字，EclipseRcp为基础框架进行开发。

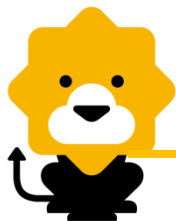


第一部分 自动化测试概述

第二部分 Selenium2.0基础介绍

第三部分 Appium基础介绍

第四部分 SAT使用介绍



Selenium2.0-所支持的浏览器

支持常用的三种浏览器：IE、Firefox、Chrome

1. 打开IE浏览器

```
Driver driver = new InternetExplorerDriver();
```

2. 打开Firefox浏览器

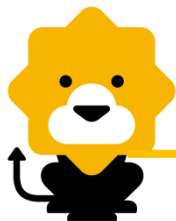
```
Driver driver = new FirefoxDriver();
```

3. 打开Chrome浏览器

```
Driver driver = new ChromeDriver();
```

4. 打开测试页面

```
driver.get("http://www.suning.com");
```



Selenium2.0-查找页面元素1

Webdriver的findElement方法可以用来找到页面的某个元素，最常用的方法是用id和name查找

假设页面写成这样：

```
<a class= "text" href= "www.abc.com" name= "passwd" id= "passwd-id" >登录</a>
```

那么可以这样找到页面的元素：

1. By ID

```
WebElement element = driver.findElement(By.id("passwd-id"));
```

2. By Name

```
WebElement element = driver.findElement(By.name("passwd"));
```

3. By XPATH

```
WebElement element =driver.findElement(By.xpath("//a[@id='passwd-id']"));
```



Selenium2.0-查找页面元素2

假设页面写成这样：

```
<a class= "text" href= "www.abc.com" name= "passwd" id= "passwd-id" >登录</a>
```

那么可以这样找到页面的元素：

4. By Class Name

```
WebElement element = driver.findElement(By.className ( "text"));
```

5. By Link Text

```
WebElement element = driver.findElement(By. linkText( "登录"));
```



Selenium2.0-操作页面元素1

找到页面元素后，对页面元素进行操作，举例说明：

输入框 (text field or textarea)

- 找到输入框元素：
`WebElement element = driver.findElement(By.id("passwd-id"));`
- 在输入框中输入内容：
`element.sendKeys("test");`
- 将输入框清空：
`element.clear();`
- 获取输入框的文本内容：
`element.getText();`



Selenium2.0-操作页面元素2

找到页面元素后，对页面元素进行操作，举例说明：

下拉选择框(Select)

- 找到下拉选择框的元素：

```
Select select = new Select(driver.findElement(By.id("select")));
```

- 选择对应的选择项：

```
select.selectByVisibleText( "mediaAgencyA" ); 或
```

```
select.selectByValue( "MA_ID_001" );
```

- 不选择对应的选择项：

```
select.deselectAll();
```

```
select.deselectByValue( "MA_ID_001" );
```

```
select.deselectByVisibleText( "mediaAgencyA" );
```

- 获取选择项的值：

```
select.getAllSelectedOptions();
```

```
select.getFirstSelectedOption();
```



Selenium2.0-操作页面元素3

找到页面元素后，对页面元素进行操作，举例说明：

单选项(Radio Button)

- 找到单选框元素：
`WebElement bookMode =driver.findElement(By.id("BookMode"));`
- 选择某个单选项：
`bookMode.click();`
- 清空某个单选项：
`bookMode.clear();`
- 判断某个单选项是否已经被选择：
`bookMode.isSelected();`



Selenium2.0-操作页面元素4

找到页面元素后，对页面元素进行操作，举例说明：

多选项(checkbox)

- 找到单选框元素：

```
WebElement checkbox =driver.findElement(By.id("myCheckbox."));
```

- 选择某个单选项：

```
checkbox.click();
```

- 清空某个单选项：

```
checkbox.clear();
```

- 判断某个单选项是否已经被选择：

```
checkbox.isSelected();
```



Selenium2.0-操作页面元素5

找到页面元素后，对页面元素进行操作，举例说明：

按钮(button)

- 找到按钮元素：
`WebElement saveButton = driver.findElement(By.id("save"));`
- 点击按钮：
`saveButton.click();`
- 判断按钮是否可用：
`saveButton.isEnabled ();`

弹出对话框(Popup dialogs)

- `Alert alert = driver.switchTo().alert();`
- 确认：`alert.accept();`
- 取消：`alert.dismiss();`
- 获取alert文本：
`alert.getText();`



Selenium2.0-操作页面元素6

找到页面元素后，对页面元素进行操作，举例说明：

windows和Frames切换

- 切换至窗口A主框架：
`driver.switchTo().defaultContent();`
- 从窗口A切至新打开窗口B：
`driver.switchTo().window("B windowtitle");`
- 在某窗口中切换frame (frame需逐层切换)
`driver.switchTo().frame("frame的控件定位符");`

导航 (Navigation and History)

- 打开一个新的页面：
`driver.navigate().to("http://www.redbaby.com");`
- 通过历史导航进行浏览器前进/后退：
`driver.navigate().forward();`
`driver.navigate().back();`



第一部分 自动化测试概述

第二部分 Selenium2.0基础介绍

第三部分 Appium基础介绍

第四部分 SAT使用介绍



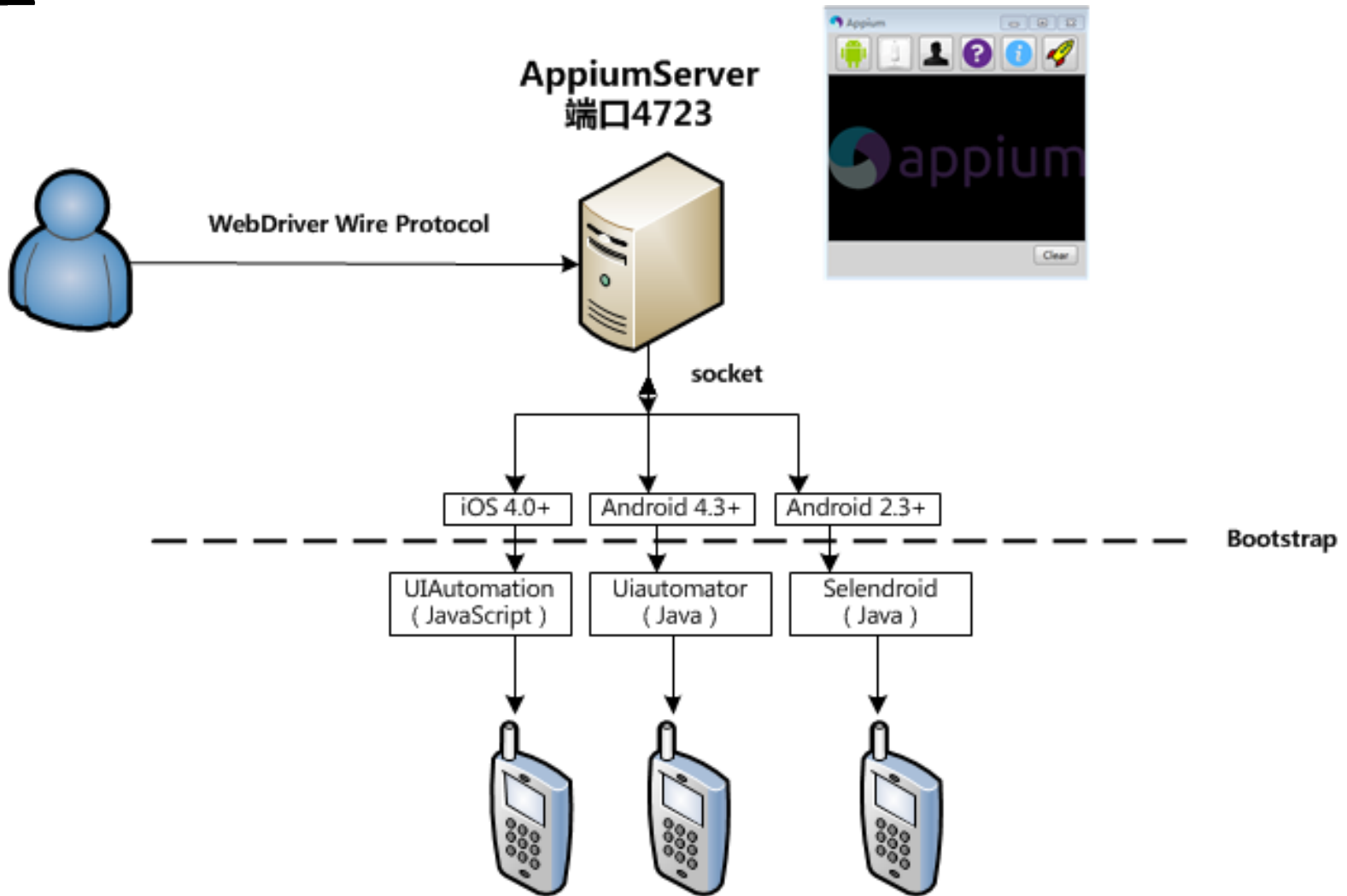
Appium框架核心原理



- **Appium is an open source test automation framework for use with native, hybrid and mobile web apps.**
 - Appium是一个开源的自动化测试框架，用于对原生应用、混合应用和 移动Web应用进行测试
- **It drives iOS and Android apps using the WebDriver protocol.**
 - 它通过WebDriver Protocol驱动iOS和Android应用

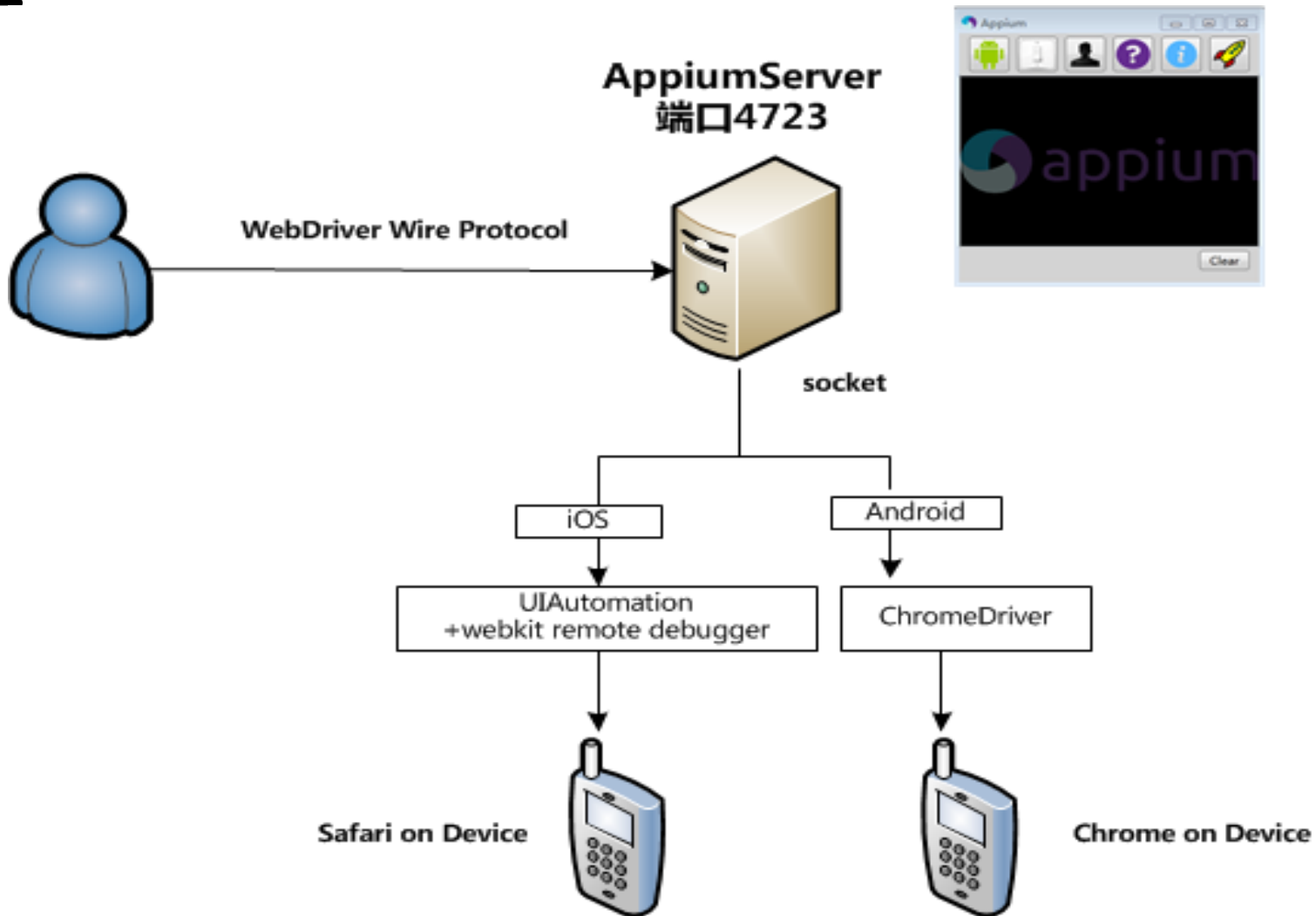


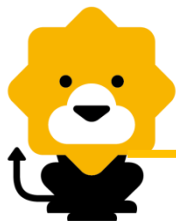
Appium之Native原理





Appium之WAP原理

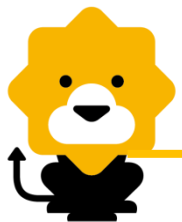




UI Automator Viewer 查找控件

The screenshot displays the UI Automator Viewer interface. On the left, a mobile application screen is shown with a login form. A red box highlights the '请输入工号' (Please enter employee ID) input field. Below it is the '请输入SOA密码' (Please enter SOA password) field, and at the bottom is a green '登录' (Login) button. On the right, the UI tree shows a hierarchy of views, with the selected 'EditText:请输入工号' node highlighted. Below the tree is the 'Node Detail' table.

Node Detail	
index	0
text	请输入工号
resource-id	com.suning.snmessenger:id/username_edit
class	android.widget.EditText
package	com.suning.snmessenger
content-desc	
checkable	false
checked	false
clickable	true
enabled	true
focusable	true
focused	false



WAP应用控件查找（H5控件定位）

- Chrome debugger
- Firebug

38



Appium代码举例

```
package cn.appium.test;
```

```
import io.appium.java_client.android.AndroidDriver;  
import io.appium.java_client.remote.MobileCapabilityType;
```

```
import java.io.File;  
import java.net.MalformedURLException;  
import java.net.URL;
```

```
import org.openqa.selenium.By;  
import org.openqa.selenium.remote.DesiredCapabilities;
```

```
@SuppressWarnings("rawtypes")
```

```
public class MyAppium {
```

```
    public static void main(String[] args) throws MalformedURLException {
```

```
        File appDir = new File("D:/test");
```

```
        File app = new File(appDir, "SnMessenger.apk");
```

```
        DesiredCapabilities capabilities = new DesiredCapabilities();
```

```
        capabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "");
```

```
        capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "SuningDevice");
```

```
        capabilities.setCapability(MobileCapabilityType.APP, app.getAbsolutePath());
```

```
        capabilities.setCapability(MobileCapabilityType.NEW_COMMAND_TIMEOUT, 120);
```

```
        AndroidDriver driver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
```

```
        driver.findElement(By.id("com.suning.snmessage:id/username_edit")).sendKeys("hellworld");//定位工号输入框输入工号
```

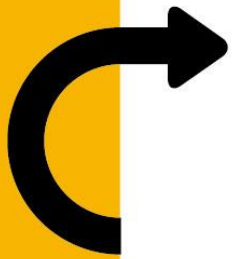
```
        driver.findElement(By.id("com.suning.snmessage:id/password_edit")).sendKeys("hellworld");
```

```
        driver.findElement(By.id("com.suning.snmessage:id/login_button")).click();
```

```
        driver.quit();
```

```
    }
```

```
}
```

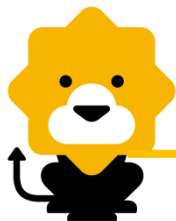



第一部分 自动化测试概述

第二部分 Selenium2.0基础介绍

第三部分 Appium基础介绍

第四部分 SAT使用介绍



SAT简介

什么是SAT？



苏宁自动化测试工具（Suning Automation Tester），简称“SAT”，SAT是苏宁测试工具开发部自主研发的集WEB页面、HTTP协议、桌面终端、手机终端、数据库操作、LINUX操作等方面的自动化测试工具；该工具基于RCP平台上实现自动化用例设计和自动化执行的工具。它采用自动化测试架构中的关键字驱动(Keyword)思想，使测试设计和测试实现分离，将实现不同公共组件类、业务类Keyword集成到一个用例中运行，也最大限度地实现Keyword共享，降低测试组重复开发的工作量，使测试人员可以更关注业务本身的测试；另外工具提供了较完善的日志和测试报告功能，方便用户查看用例执行日志和批量执行测试报告；后面我们将逐步更好的完善工具的功能，最大限度的降低工具使用门槛，提高其自动化开发、测试效率。



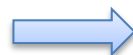
SAT所以支持的终端类型

WEB-PC端



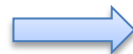
支持浏览器有：IE、
Firefox、chrome

WAP端



移动终端浏览器、H5

移动终端



支持系统有：Android、
IOS



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



获取控件的常用方式

不区分大小写

- id
- name
- className
- linkText
- partialLinkText
- tagName
- xpath



获取控件常用方式-ID

- * ID是常用的控件定位方式之一
- * 某控件元素存在ID属性，且ID的属性值在本页面是唯一的，则可使用ID作为该控件的定位方式。
- * **举例**：SOP登录页的用户名输入框

苏宁云台商家登录

用户名:

使用Firefox的插件Firebug或者IE8自带的开发人员工具可以查看易购首页的Html控件元素结构。

```
<li>  
  <label>用户名: </label>  
  <input id="account" class="uiText" type="text" onblur="snTxtBlur(this,'请输入用户名');userN  
' );" value="testa@163.com" name="username">  
  <span></span>  
</li>
```

控件存在ID属性，且account是SOP登录页唯一值，可选择ID作为控件定位方式。

● 文本框文本输入

参数名	参数值
文本框控件定位符	id::account
待输入的文字	testa@163.com



获取控件常用方式-NAME

- * NAME是常用的控件定位方式之一
- * 某控件元素存在NAME属性，且NAME的属性值在本页面是唯一的，则可使用NAME作为该控件的定位方式。
- * **举例**：SOP登录页的密码输入框，存在NAME属性，且取值是唯一的。

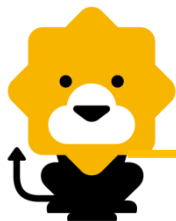
苏宁云台商家登录

用户名:

密码:

```
<label>密码: </label>  
<input id="pwd" class="uiText" type="password" value="" maxlength="20" name="password" autocomplete="off">  
<span></span>
```



参数名	参数值
文本框控件定位符	name::password
待输入的文字	\$VAR_Password



获取控件常用方式-ClassName

- * 某控件元素存在Class属性，且class的属性值在本页面是唯一的，则可使用classname作为该控件的定位方式。
- * 若class值包含空格，如abc cd，则不可以使用className定位
- * **举例：**SOP校验登录成功 [testa@163.com](#) 欢迎来苏宁云台 [退出](#)

```
<p class="loginInfo">  
  <span id="userName">testa@163.com</span>  
  欢迎来苏宁云台  
  <a class="exit" href="https://passportsit.cnsuning.com/ids/1">  
</p>
```

参数名	参数值
控件定位符	classname::loginInfo
预期文本	欢迎来苏宁云台
预期结果	 true
匹配模式	 模糊匹配



获取控件常用方式-(partial)linkText

- * 某控件元素是一个以a为标签的链接，且链接的文本在本页面是唯一的，则可使用linktext和partialLinkText作为该控件的定位方式。
- * 不同：partialLinkText支持模糊匹配

* 举例：SOP首页登录链接



```
<p id="loginInfo" class="loginInfo">  
  欢迎来苏宁云台  
  <a class="exit" target="_blank" href="/sel/login/logon.action" >登录</a>  
  <a class="exit" target="_blank" href="https://sopsit.cnsuning.com/sel/memb  
</p>
```

控件点击

参数名	参数值
控件定位符	linktext::登录

控件点击

参数名	参数值
控件定位符	partialLinkText::登录



获取控件常用方式-tagName

- * 某控件元素的节点在本页是唯一的，则可使用tagName作为该控件的定位方式。
- * tagName作为控件定位较不常用，原因：控件节点在本页一般不是唯一存在
- * 书写格式：tagName::xxxxx

* 举例：

```
<p id="loginInfo" class="loginInfo">  
    欢迎来苏宁云台  
    <a class="exit" target="_blank" href="/sel/login/logon.action" 登录 </a>  
    <a class="exit" target="_blank" href="https://sopsit.cnsuning.com/sel/memb  
</p>
```

上述红框中a即为控件的tagName



获取控件常用方式-定位原则

定位原则：

- * 首先：存在id/name，且具有唯一性，优先使用id/name作为控件定位方式
- * 其次：首先不满足时，
 - 对于非a节点，考虑使用className作为定位方式；
 - 对于a节点，优先考虑使用linkText/partialLinkText作为定位方式，其次考虑使用className定位
- * 最后：前面2个不满足时，考虑使用xpath作为定位方式



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



xPath使用基础-xpath简介

- **同胞 (Sibling)**
- 拥有相同的父的节点
- 在下面的例子中，title、author、year 以及 price 元素都是同胞：
- `<book>`
- `<title>Harry Potter</title>`
- `<author>J K. Rowling</author>`
- `<year>2005</year>`
- `<price>29.99</price>`
- `</book>`



xPath使用基础-xpath简介

路径表达式：

➤ /bookstore：取根元素 bookstore。

注释：假如路径起始于正斜杠(/)，则此路径始终代表到某元素的绝对路径

➤ bookstore/book：选取属于 bookstore 的子元素的所有 book 元素。

➤ //book：选取所有 book 子元素，而不管它们在文档中的位置。

➤ bookstore//book：选择属于 bookstore 元素的后代的所有 book 元素

➤ /bookstore/book[1]：选取属于 bookstore 子元素的第一个 book 元素。

➤ /bookstore/book[last()]：选取属于 bookstore 子元素的最后一个 book 元素。


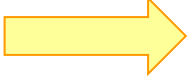

➤ //title[@lang= 'en']：选取所有 title 元素，且这些元素拥有值为 en 的 lang 属性。

➤ /bookstore/* :选取 bookstore 元素的所有子元素。

➤ //*:选取文档中的所有元素。



xPath使用基础-选择属性

- 在XPath语法中，要获得属性信息必须以前缀“@”来指定，如下：
- `//@ id`  表示选择所有属性名为“id”的属性
- `/class/student[@ id]`  表示选择“class”元素下包含有“id”属性的所有“student”元素
- `/class/student[@ id=' ADR02']`  表示选择“class”元素下包含有“id”属性的值为“ADR02”的所有“student”元素



xPath使用基础-使用举例

在SOP首页左侧菜单商品管理项



```
<div id="test">  
  <div id="leftMenu" class="leftMenu">  
    <ul>  
      <li>  
      <li>  
      <li>  
      <li>  
      <li>  
      <li>  
    </ul>  
    <dl>  
      <dt class="menu6 on">  
        <span>/</span>  
        <em>/</em>  
        商品管理  
      </dt>  
      <dd style="display: block;">  
        <a href="javascript:menuOnClick('http://sopsit.cnsuning.com/sel/cm/mylibrary.action','SOPC103C">  
        </dd>  
    </dl>  
  </div>  
</div>
```

Xpath:://*[@id='leftMenu']//*[contains(text(),'商品管理')]



xPath使用基础-定位原则

- XPath使用定位原则：
 - XPath是万用的定位方式，基本上控件都可用xpath来定位，但不是优先考虑的定位
- id::xxx 等同于 xpath:://*[@id='xxx']
- name::xxx 等同于 xpath:://*[@name='xxx']
- className::xxx 等同于 xpath:://*[@class='xxx']
- linkText::xxx 等同于 xpath:://a[text()='xxx']
- partialLinkText::xxx 等同于 xpath:://a[contains(text(),'xxx')]
- xpath定位优先使用待定位节点本身的属性及节点组合来唯一确定该节点；
若无法唯一确定该节点，则向上考虑找到其可以唯一确定的父节点，通过定位父节点来定位；
若父节点不可以，则找其父节点的父节点/兄弟节点...



xPath使用基础-定位原则

preceding-sibling 选取当前节点之前的所有同级节点

/AAA/XXX/preceding-sibling::* /AAA/XXX节点的所有之前同级节点

```
<BBB>  
  <CCC/>  
  <DDD/>  
</BBB>  
<XXX>
```

following-sibling 选取当前节点之后的所有同级节点

/AAA/BBB/following-sibling::* 取/AAA/BBB节点的之后的所有同级节点



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



SAT-Web端 用例设计原则

- 保证用例的独立性、完整性

如：避免用例间的干扰，降低耦合；

要保证数据的准确性，才能保证测试结果的有效性；

端到端完整流程的保证；

备注：用例单条循环执行无脚本错误；多条执行错误对其上下用例无影响

- 用例的重用性

如：基于业务特定的功能，实现一组封装，降低接口、页面变更对用例的影响；

分析哪些数据需要做变量设置，避免因环境和业务功能的改变而影响用例

- 用例的可读性

如：必要的注释；

前置条件、步骤、预期结果放在正确的位置；



SAT-Web端 用例设计原则

- 用例的健壮性

如：避免由于机器速度、测试环境导致自动化执行的不稳定性；

不能写死等待，而是用智能校验的方式

- 用例的结果检查全面

跟手工用例的检查点一样完整，缺一不可，不允许对一些重要功能做弱校验

如：数据库检查、响应消息、成功、失败返回结果校验等。

以上都是一些通用的规范，结合自身的业务特点，会有一些其他规范要求



SAT-Web端 用例设计步骤

- 举例：web自动化
- 在每条案例的前置步骤加入“打开浏览器-用户登录”操作，后置步骤加入“浏览器退出”，容错步骤—“浏览器进程清除/浏览器退出”
- 如上设计在每条案例执行得第一步会先执行“打开浏览器-用户登录”，案例执行完成之后便执行后置步骤“浏览器退出”，接下来又是同样的循环
- 如在案例执行过程中有任何一步出错的话，会直接跳到容错步骤，执行“浏览器进程清除”，接下来会执行下一条案例的前置步骤，即“打开浏览器-用户登录”



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



SAT支持应用类型

- SAT支持3种主流app设计类型
 - **原生应用 (Native)** : 使用开发特定平台专用app
如：豆芽、苏宁易购移动客户端、易付宝移动客户端等
 - **混合应用 (hybrid)** : 兼具Native App和Web App两种模式应用特点，用WebView装载H5页面
如：苏宁阅读、百度新闻等轻应用
 - **Mobile Web** : 使用H5开发技术，通过手机浏览器装载页面。
如：<http://m.suning.com/>
- SAT支持2中设备类型
 - **真机**
 - **模拟器**



SAT移动终端应用-技术和架构

使用技术

- 1、集成了Selenium、Appium开源的测试框架
- 2、跨设备，支持Android、IOS及FirefoxOS平台
- 3、跨语言，java、python、ruby、nodejs、php
- 4、实现了对测试脚本的自动化录制和回放
- 5、优点：能快速自动生成测试脚本



SAT移动终端应用-技术和架构

Appium设计理念



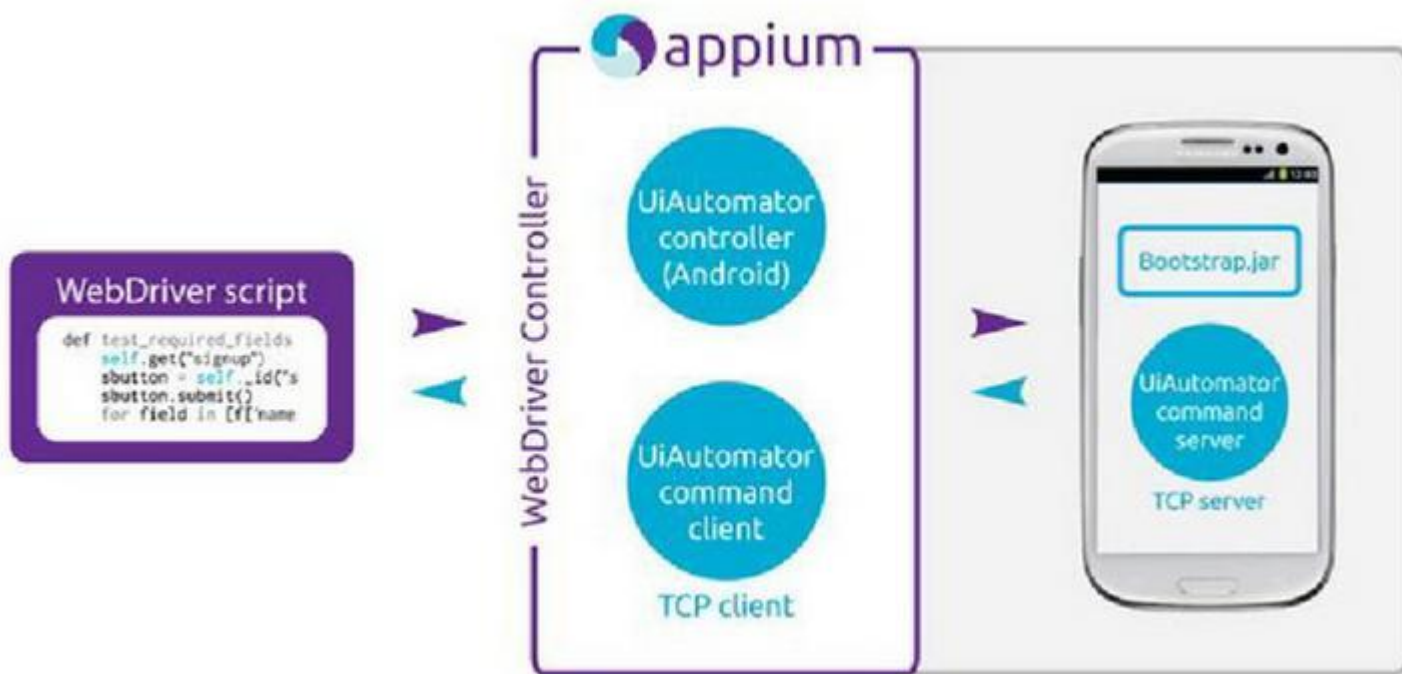
- 使用webdriver协议与用户交互
- Xpath抽象定位
- 底层调用各自平台的自动化技术，类似selenium的架构

Appium架构

- Android SDK 4.0以上使用uiautomator,4.0以下使用selendroid
- Android上使用instrumentation和uiautomator两套技术
- ios使用instruments
- 同时还支持firefoxOS，并可扩展其他平台



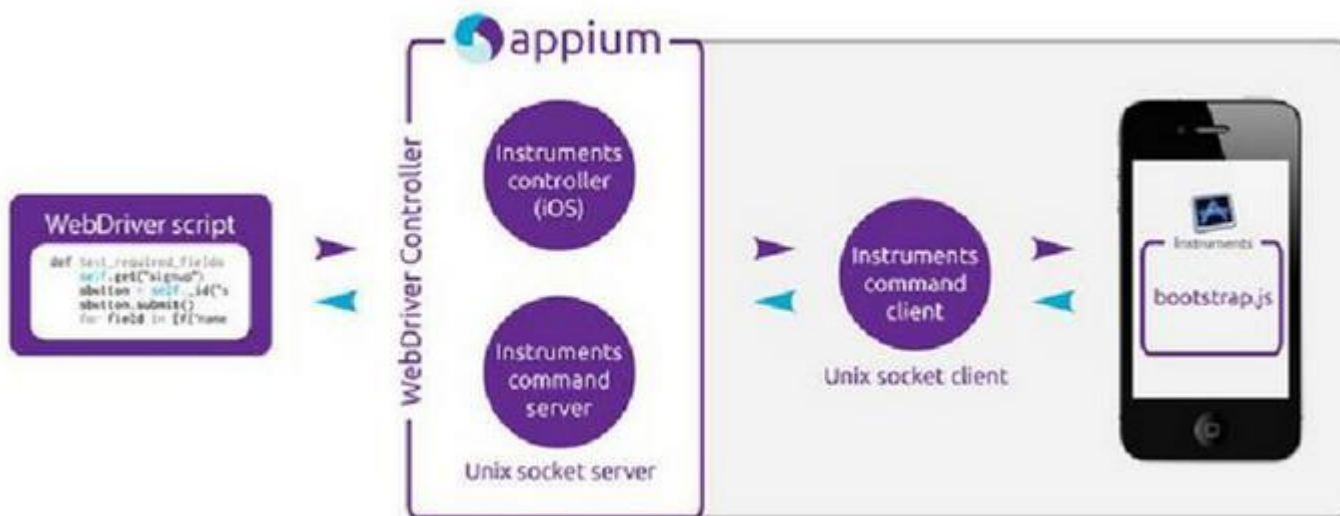
Appium在android上的架构

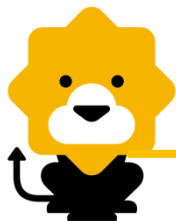




SAT移动终端应用-技术和架构

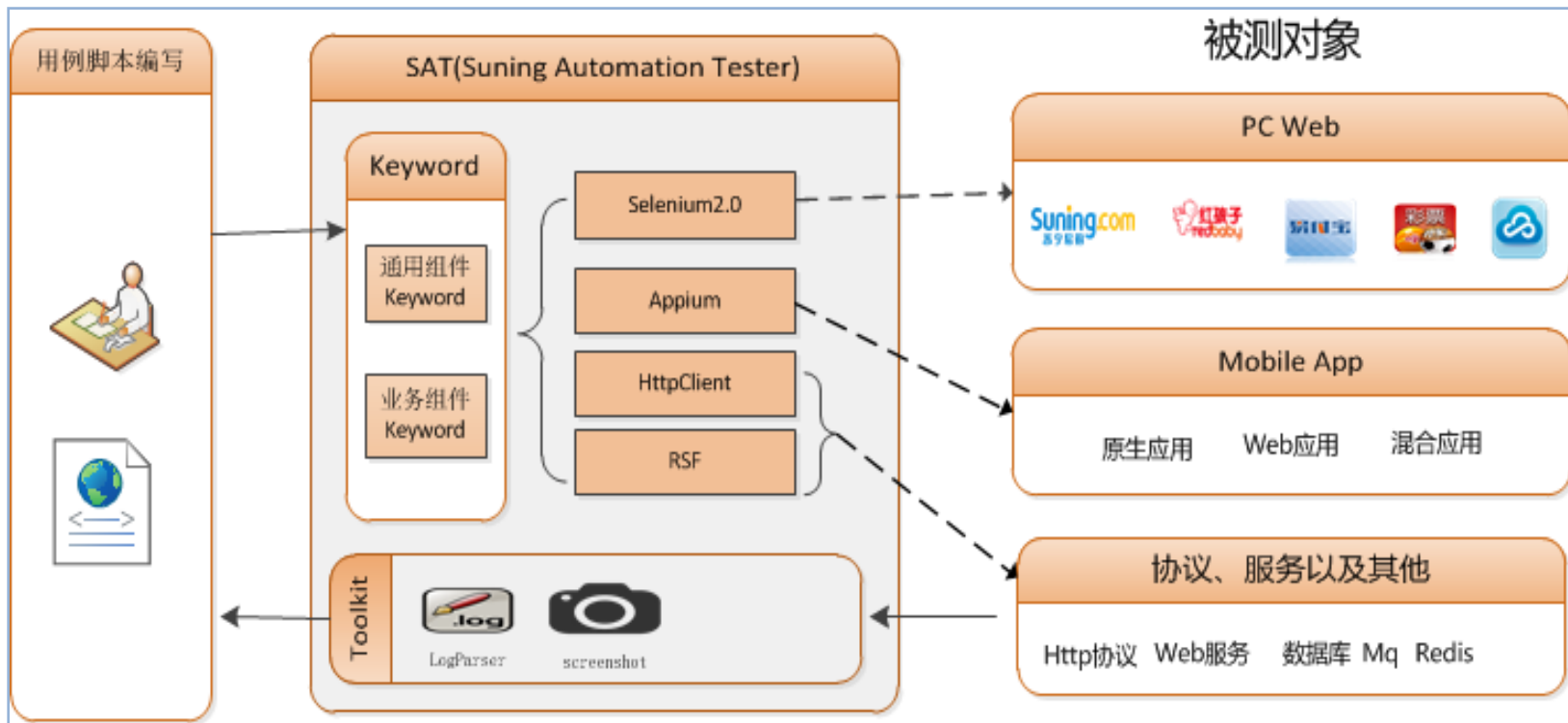
Appium在iOS上的架构





SAT移动终端应用-技术和架构

Appium在SAT上的架构





Appium的加载流程

- 调用android adb完成基本的系统操作
- 向android上部署bootstrap.jar包并启动
- Forward android的端口到pc机器上

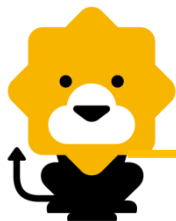
- Pc上监听端口接受请求，使用webdriver协议
- 分析命令并通过forward的端口发给bootstrap.jar
- Bootstrap.jar接受请求并把命令发给uiautomator



SAT移动终端应用-技术和架构

流程举例

- Webdriver脚本执行`element.send_keys`
- 发送到remote webdriver
 - `/wd/hub/session/xxx/keys`
- pc处理请求并转发给android和iphone
- Bootstrap.jar调用uiautomator
- Uiautomator调用`setText`方法



基于SAT框架编写Java代码脚本

```
public class MobileTest {  
  
    public static void main(String[] args) {  
        Logger.openLog(false);  
        //创建豆芽测试Driver  
        Mobiles.create(MobileOS.android, "com.suning.snmessage", "com.suning.snmessage.login.LoginActivity");  
        //启动豆芽应用  
        Mobiles.current().launchDefaultApp();  
        //输入用户名  
        Mobiles.current().findElement(ElementLocator.id("com.suning.snmessage:id/username_edit")).sendKeys("13050526");  
        //输入密码  
        Mobiles.current().findElement(ElementLocator.id("com.suning.snmessage:id/password_edit")).sendKeys("13050526");  
        //点击登录  
        Mobiles.current().findElement(ElementLocator.id("com.suning.snmessage:id/login_button")).click();  
        //校验  
        Mobiles.current().verifyElementExisted(ElementLocator.id("com.suning.snmessage:id/login_button"), false);  
    }  
}
```




SAT移动终端应用预置配置

•手机配置

- 1) USB调试功能打开 ;
- 2) 设置屏幕常亮和永不锁屏
- 3) 输入法设置为手机默认输入法 (一般为英文键盘)
- 4) 已正确安装手机驱动

•其它条件：有待测应用的apk安装包或者应用已安装



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

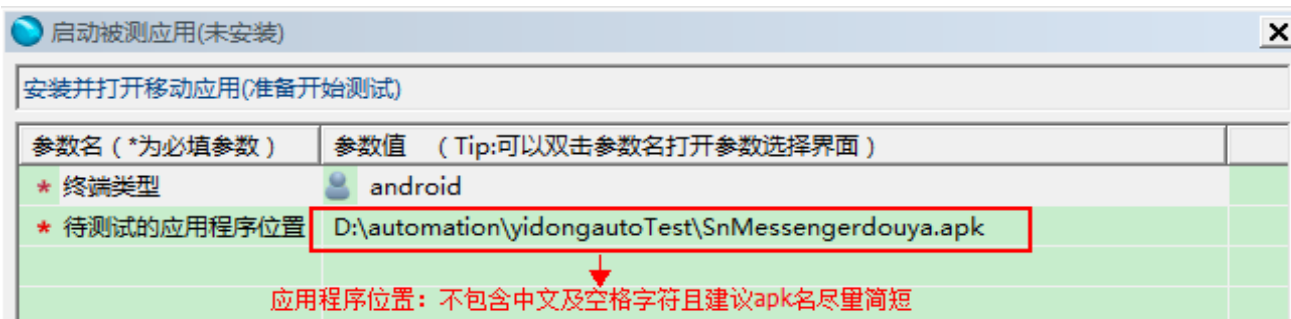
移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



SAT移动终端应用-APK端

- 编写用例（eg：使用安卓系统真机进行豆芽apk登录功能演示）步骤
 - 拖拽移动设备关键字，并调试运行，通过Appium实现PC端与移动终端通信，完成app安装
 - 1) 启动appium应用
 - 2) 启动被测应用(未安装)/启动被测应用(已安装)



PS：上述步骤亦可放在测试套中作为所有用例的前置步骤，执行一次使用调试当前脚本 或者 调试选中步骤，完成app安装/启动（调试运行步骤必不可少）注：

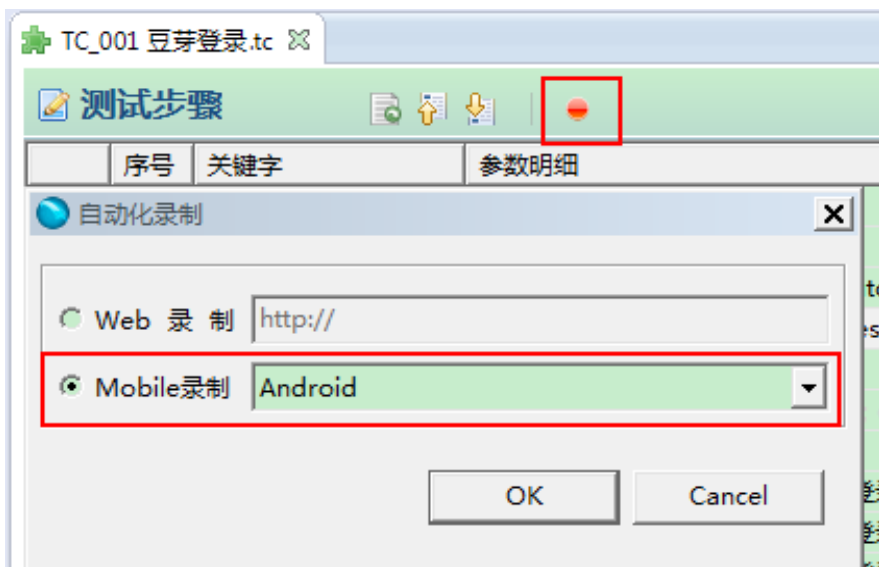
- 1) 执行启动被测应用(未安装)或启动被测应用(已安装)才能使PC端与mobile端建立连接
- 2) 启动被测应用(已安装)参数可通过关键字：【获取app包名和入口Activity】获取



SAT移动终端应用-APK端

➤启动Mobile录制界面，通过Device Screenshot按钮实现PC与移动终端屏幕同步

- 1)启动Mobile录制界面：点击录制按钮，选择Mobile录制-Android（Iphone暂不支持），点击【OK】。

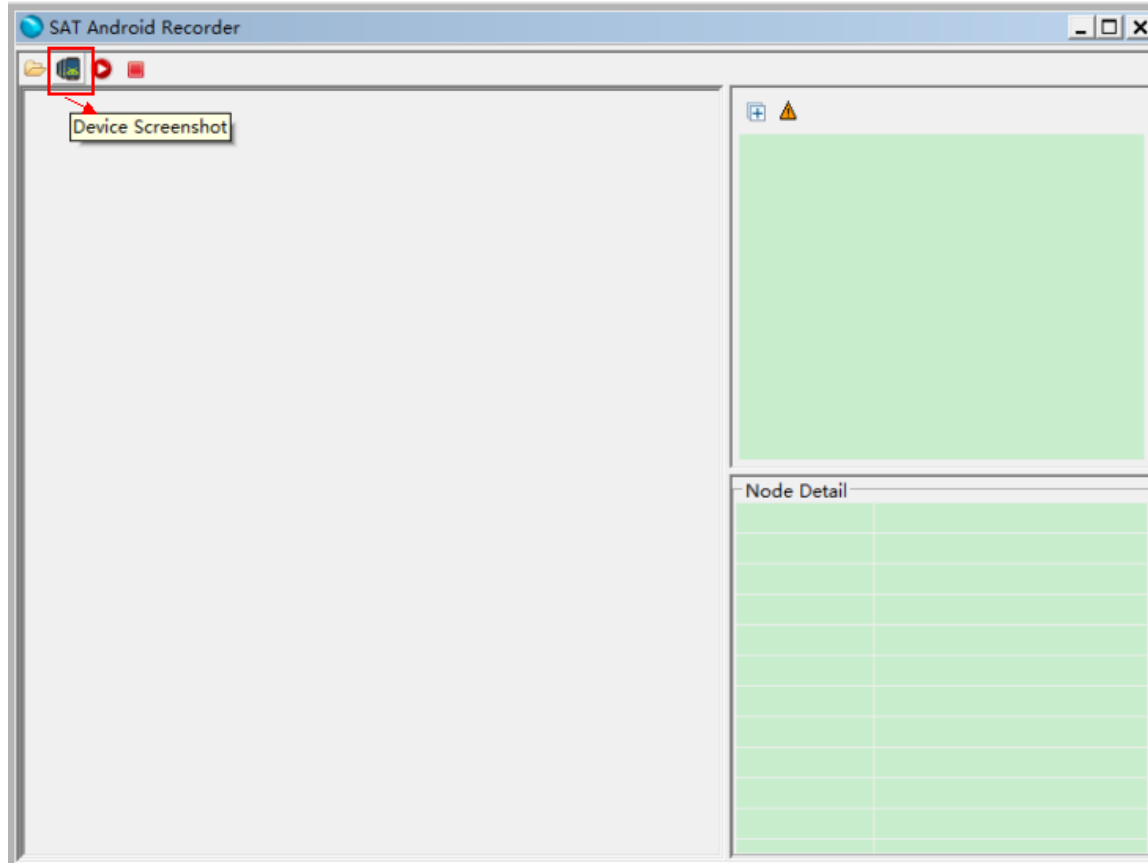


上述步骤会打开 SAT Android Recorder界面。



SAT移动终端应用-APK端

2) 点击Device Screenshot按钮，实现PC与手机屏幕同

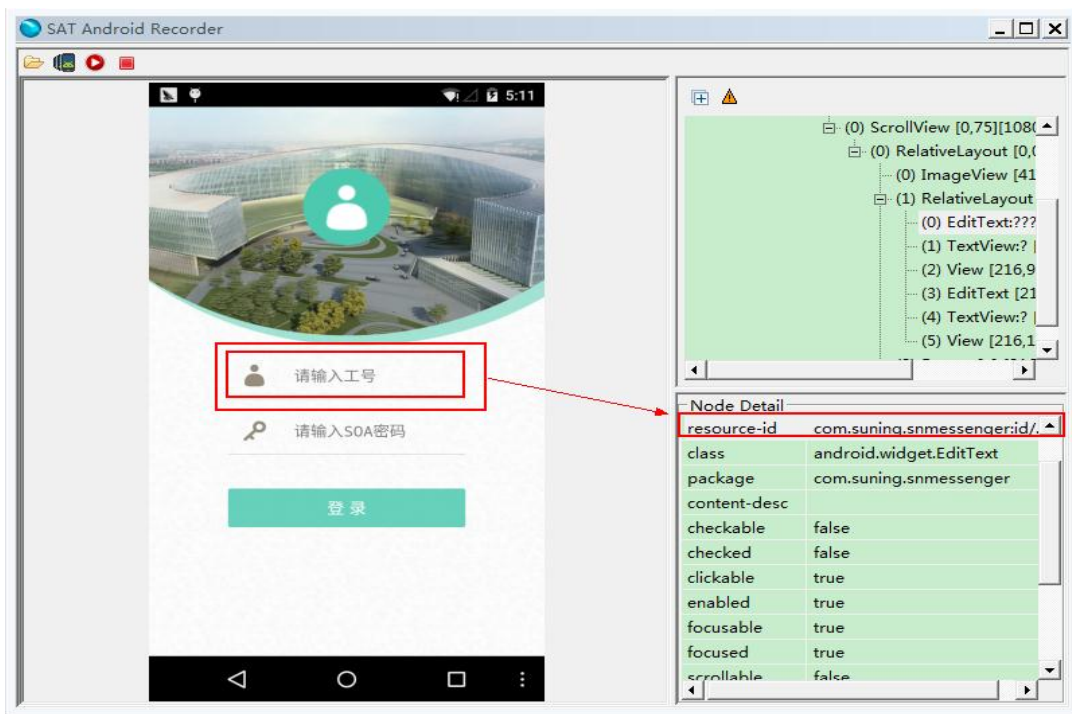




SAT移动终端应用-APK端

编写用例脚本 方式一:

- 1)根据用例手工操作步骤，在SAT Android Recorder界面鼠标点击待操作的控件，比如输入用户名，则点击用户名的文本框。
- 2)在右侧下方出现控件的属性信息，包括id、name、xpath；选择可唯一标识该控件的属性，优先级顺序id>name>xpath





SAT移动终端应用-APK端

3)选择对应操作的通用关键字编写测试步骤，比如输入用户名步骤，选择文本框文本输入关键字。

文本框文本输入(mobile)	
根据文本框控件定位符查找文本框位置，清空文本框中原文字，同时将期望文字模拟输入至文本框中	
参数名 (*为必填参数)	参数值 (Tip:可以双击参数名打开参数选择界面)
* 文本框控件定位符	id::com.suning.snmessage:id/username_edit
待输入的文字	13076762



SAT移动终端应用-APK端

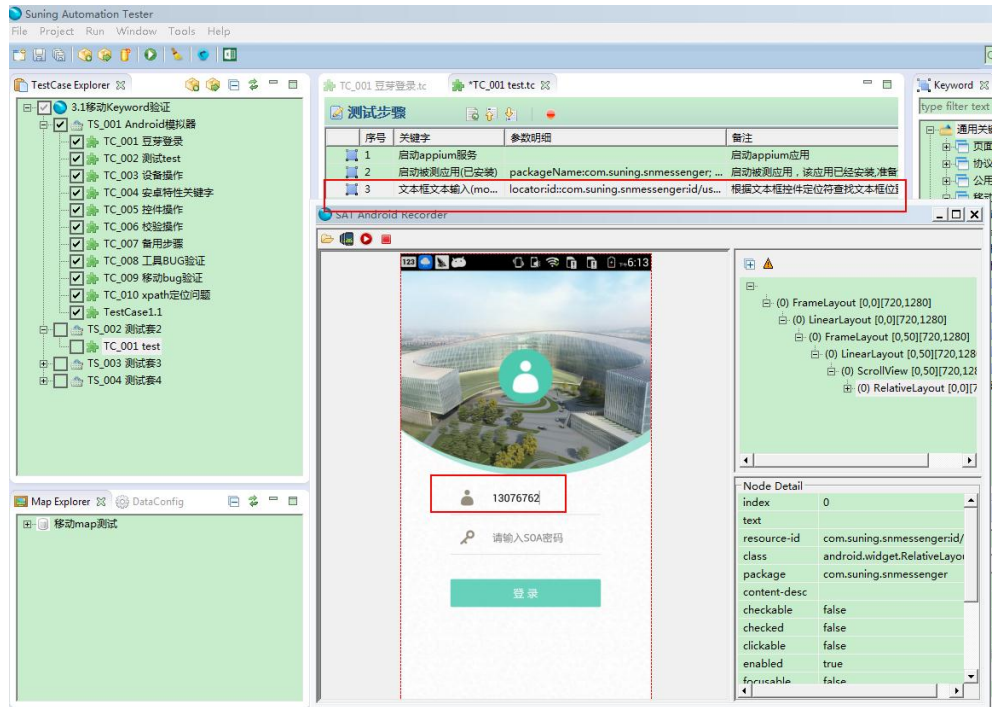
➤编写用例脚本 方式二：

在SAT Android Recorder界面点击start启动录制功能

根据用例手工操作步骤，鼠标右击待操作的控件，选择相应的操作菜单，包括点击、文本框输入、校验等，自动生成脚本步骤

页面步骤及校验点录制完毕，手工操作手机界面进行应用界面跳转；SAT Android Recorder界面刷新同方法一描述。

点击stop按钮停止录制





SAT移动终端应用-APK端

步骤：

- 拖拽移动设备关键字，关闭appium及应用
- 关闭移动应用(结束测试)
- 停止appium应用

序号	关键字	参数明细	备注
1	启动appium服务		启动appium应用
2	启动被测应用(已安装)	packageName:com.suning.snessenger; ...	启动被测应用，该应用已经安装,准备测试
3	文本框文本输入(mo...	locator:id::com.suning.snessenger/id/us...	根据文本框控件定位符查找文本框位置，清空...
4	文本框文本输入(mo...	locator:map::移动map测试/豆芽登录.map:...;	根据文本框控件定位符查找文本框位置，清空...
5	控件点击(mobile)	locator:map::移动map测试/豆芽登录.map:...;	根据定位符查找控件，并模拟鼠标点击该控件
6	关闭应用		关闭应用，结束测试，关闭后可再打开应用，...
7	停止appium服务		用于停止appium服务



SAT自动化测试工具基本使用介绍

WEB-PC端

- 获取控件的常用方式
- xpath使用基础
- SAT-Web端用例设计

移动终端

- SAT移动终端应用技术介绍
- SAT移动终端应用-APK端
- SAT移动终端应用-WAP端



◆浏览器支持：

Browser、Chrome、Safari

说明：浏览器需要预先安装在手机中



SAT移动终端应用-WAP端

•编写用例（eg：使用安卓系统真机m.suning.com演示）

步骤：

➤拖拽移动设备关键字，并调试运行，打开易购首页

- 1) 启动appium应用
- 2) 打开浏览器（mobile）
- 3) 浏览器地址输入（mobile）

使用调试当前脚本 或者 调试选中步骤，完成易购首页打开

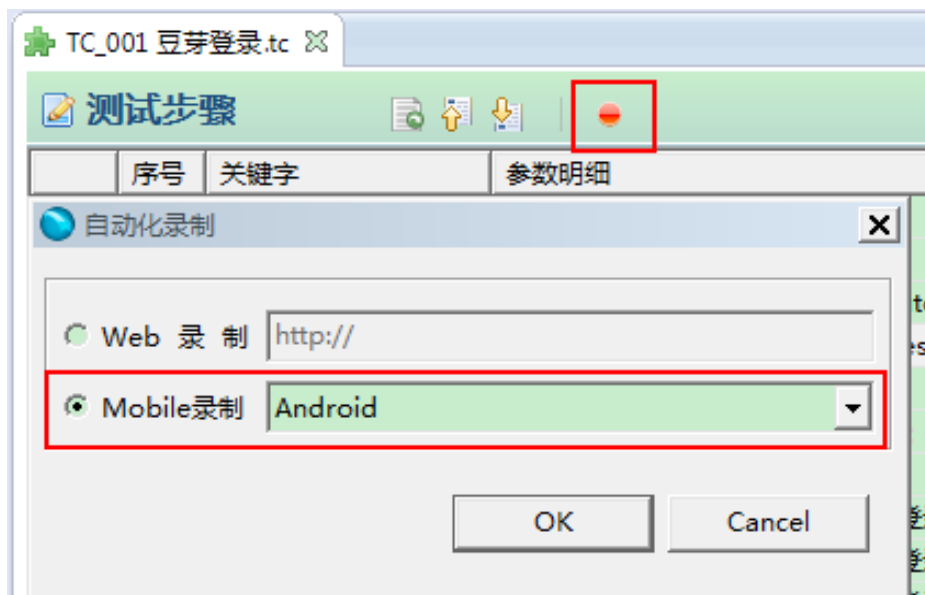
注：执行打开浏览器才能使PC端与mobile端建立连接。



SAT移动终端应用-WAP端

步骤：

- 启动Mobile录制界面，通过Device Screenshot按钮实现PC与移动终端屏幕同步
- 1)启动Mobile录制界面：点击录制按钮，选择Mobile录制-Android（Iphone暂不支持），点击【OK】。

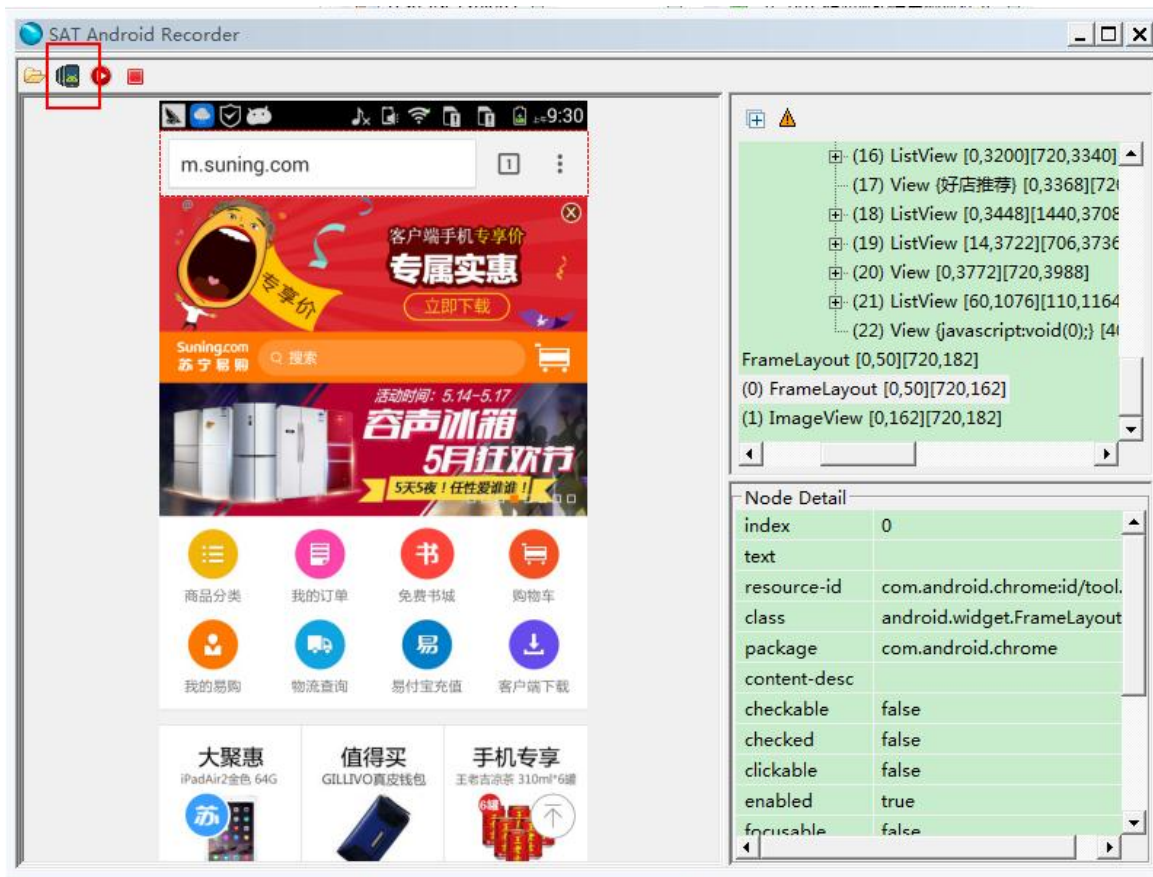


上述步骤会打开 SAT Android Recorder界面。



SAT移动终端应用-WAP端

2) 点击Device Screenshot按钮，实现PC与手机屏幕同步





SAT移动终端应用-WAP端

➤编写用例脚本：

选择对应操作的通用关键字编写测试步骤，比如搜索框搜索步骤，需选择控件点击、文本框文本输入两个关键字。拖拽相应关键字，填写所需参数值。

eg: wap端苏宁易购首页搜索框控件定位: name::index_none_header_sysc

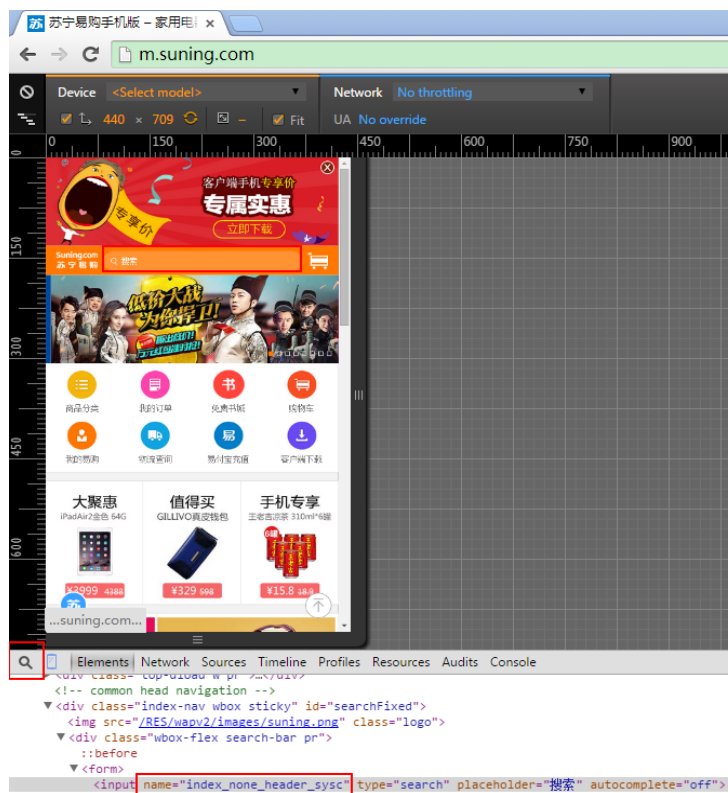
控件点击(mobile)	
根据定位符查找控件，并模拟鼠标点击该控件	
参数名 (*为必填参数)	参数值 (Tip:可以双击参数名打开参数选择界面)
* 控件定位符	name::index_none_header_sysc



SAT移动终端应用-WAP端

➤控件定位

- 1) native:通过录制页面获取控件定位，同APK端
- 2) H5：同PC端WEB定位，建议使用chrome浏览器打开并定位
如，首页搜索文本框





SAT移动终端应用-WAP端

➤重点：H5与native混合页面：

混合型应用存在不同端控件，需使用【切换移动上下文】关键字
预期控件为NATIVE上下文，选择NATIVE; 预期控件为WEB上下文，选择WEB。

Eg: wap端苏宁易购首页滚屏操作

说明：Wap端打开网页后，默认端为WEB端。滚屏前需使用切换上下文关键字切换至NATIVE端，滚屏完毕后，操作H5页面控件前需切换回WEB端。

9	切换移动上下文	swithoption;	切换移动上下文至NATIVE端	
10	按方向滚屏	direction:下; size:全屏; count;	向下滚屏	
11	切换移动上下文	swithoption:WEB;	切换移动上下文至WEB端	
12	控件点击(mobile)	locator:id::footerLogin;	点击登录	

Thanks!

