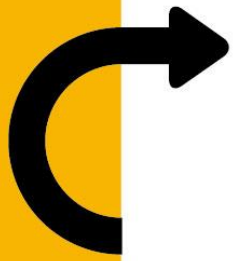
自动化测试（PDF测试岗位课程）

课程讲师：测试工具研发部



第一部分 自动化测试基础

第二部分 WEB UI功能自动化

第三部分 接口功能自动化

第四部分 手机应用功能自动化

第五部分 基于关键字的脚本开发

第一部分 自动化测试基础

- 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；

第一部分 自动化测试基础

- 自动化测试是软件测试的一个重要组成部分，是把以人为驱动的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。

第一部分 自动化测试基础

■ 涵盖各种各样的测试种类

- 功能(黑盒)自动化测试
- 功能(白盒)自动化测试
- 性能测试
- 压力测试
- GUI测试
- 安全性测试

第一部分 自动化测试基础

■ 自动化测试目的

- 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

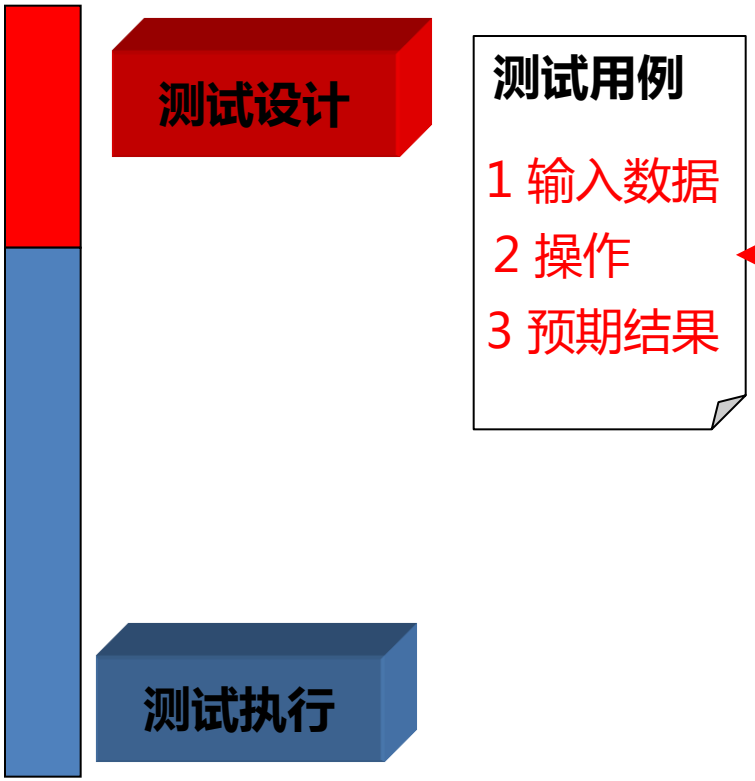
- 期望自动化测试能够完全取代手工测试
- 期望自动化测试发现大量的新缺陷
- 期望自动化测试能够智能的完成绝大多数工作
- 期望自动化测试是一劳永逸的

总结：

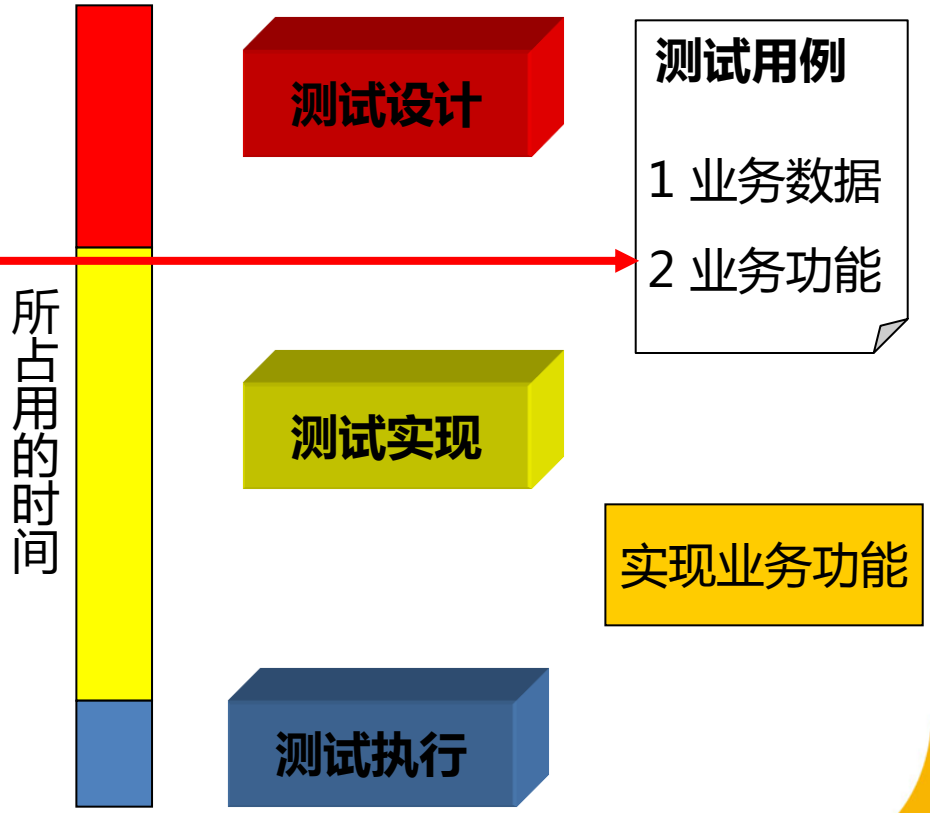
自动化测试不是万能的，它只是测试人员工具箱里的一件利器，它无法取代测试工程师的地位。

第一部分 自动化测试基础

- 手工测试



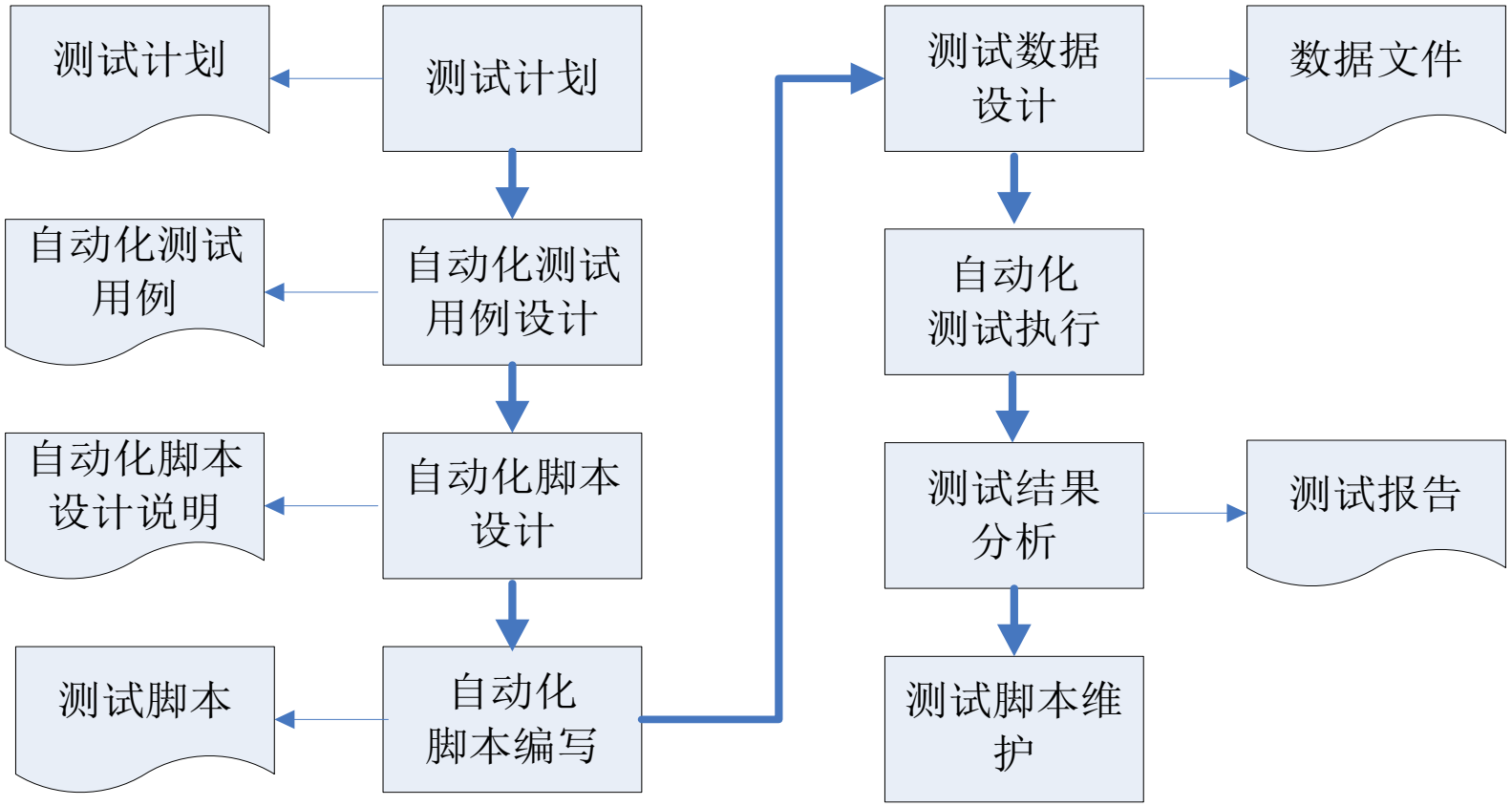
- 自动化测试



自动化测试的引入和应用

- 找准自动化测试的切入点
- 把测试开发纳入整个软件开发体系
- 测试自动化依赖测试流程和测试用例
- 软件测试自动化前期投入较大，后期收益大
- 进行资源的合理调度

流程活动图


测试计划
测试计划
自动化测试用例
自动化测试用例设计
自动化脚本设计说明
自动化脚本设计
测试脚本
自动化脚本编写
测试数据设计
数据文件
自动化测试执行
测试结果分析
测试报告
测试脚本维护

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》。

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》。

流程说明(续)

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。

- 流程说明(再续)

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。

14

第一部分 自动化测试基础

（通常）更适合手工测试，而不适合自动化测试：

- 不稳定的测试用例
- 需人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
- 后续不需或很少运行的测试用例
- 界面美观、声音体验、易用性的测试
- 定制型项目（一次性）
- 项目周期很短的项目
- 实现成本高的物理交互

第一部分 自动化测试基础

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
- 每轮都将进行回归测试的测试用例
- 高度重复、冗余的任务或场景
- 乏味且人工容易出错的工作
- 产品型项目
- 增量式开发，持续集成项目

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。

第一部分 自动化测试基础

- 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

第一部分 自动化测试基础

- 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。

第一部分 自动化测试基础

■ 自动化测试工具的原理

测试工具的优势在于可部分地替代人工的测试过程，能重复不断地执行，能精确判断数值和字符对象。自动化测试工具把测试用例用自动的方式执行，例如，自动地产生数据，自动地打开应用程序，自动地查找控件，自动地输入数据，自动地操作控件，自动地收集测试结果，自动地与预期结果进行比较等。

■ 自动化功能测试工具

实现了自动化执行测试用例，自动化地检查测试数据的测试工具，替代人工进行测试步骤的执行，从而验证应用程序是否满足了**特定功能**的测试工具。

基于代码层面的自动化测试工具

基于代码层面的功能自动化测试工具主要是一些单元测试工具，例如JUnit、NUnit、MSTest等

工具直接访问被测试的应用程序的代码，对其中的类和函数进行调用，输入各种测试数据，检查函数的返回值，通过比较返回值与期待的值是否一致来判断测试是否通过。

■ 基于浏览器和DOM对象模型的自动化测试工具

基于浏览器和DOM对象模型开发的工具，例如Selenium、Watir等

直接访问Web浏览器

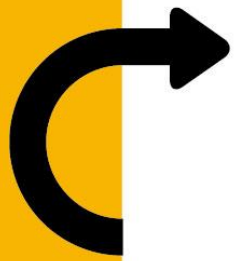
利用脚本语言操纵浏览器和Web页面中包含的DOM对象，达到模拟用户控制浏览导航、页面元素的操纵等效果

直接获取DOM对象的属性，从而获得Web页面元素的各种属性，通过这些属性可判断测试步骤的结果是否正确。

基于GUI对象识别的自动化测试工具

目前，大部分自动化功能测试工具，尤其是商业的测试工具，都是基于GUI对象识别技术来设计的，例如：QTP操作win客户端工具

利用脚本语言调用Windows的API中封装的FindWindow、GetWindowRect等函数，从而实现自动化测试编程直接获取DOM对象的属性



第一部分 自动化测试基础

第二部分 WEB UI功能自动化

第三部分 接口功能自动化

第四部分 手机应用功能自动化

第五部分 基于关键字的脚本开发

第二部分 WEB UI功能自动化

■ UI功能自动化操作原理

将操作应用程序的各种动作和输入记录下来，包括键盘操作、鼠标点击等捕捉(Record)下来，生成一个脚本文件，文件后面可以被回放(playback)。

实际测试过程中，根据测试需求对录制的脚本进行一些必要的修改或加入一些参数，如选择不同的测试数据、脚本中插入检查点(CheckPoint)进行跟踪调试等。

第二部分 WEB UI功能自动化

■ WEB-UI功能自动化测试覆盖面

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

第二部分 WEB UI功能自动化

■ 常见的WEB-UI 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。

第二部分 WEB UI功能自动化

■ 常见的WEB-UI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。

第二部分 WEB UI功能自动化

■ 苏宁的WEB-UI自动化工具

SAT是Suning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、appium等为基础封装关键字，EclipseRcp为基础框架进行开发。

第二部分 WEB UI功能自动化

■ 什么是HTML?

- HTML 是用来描述网页的一种语言。
- HTML 指的是超文本标记语言: **H**yper **T**ext **M**arkup **L**anguage
- HTML 不是一种编程语言，而是一种**标记**语言
- 标记语言是一套**标记标签** (markup tag)
- HTML 使用标记标签来**描述**网页
- HTML 文档包含了HTML **标签**及**文本**内容
- HTML文档也叫做 **web 页面**

HTML 实例

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一个标题</h1>

<p>我的第一个段落。</p>

</body>
</html>
```

第二部分 WEB UI功能自动化

■ HTML元素(标签)

- HTML 元素以**开始标签**起始
- HTML 元素以**结束标签**终止
- **元素的内容**是开始标签与结束标签之间的内容
- 某些 HTML 元素具有**空内容（empty content）**
- 空元素**在开始标签中进行关闭**（以开始标签的结束而结束）
- 大多数 HTML 元素可拥有**属性**

开始标签*	元素内容	结束标签*
	这是一个段落	
	这是一个链接	

*开始标签常被称为**起始标签（opening tag）**，结束标签常称为**闭合标签（closing tag）**。

第二部分 WEB UI功能自动化

■ HTML常见标签说明

标签	描述
基础	
<!DOCTYPE>	定义文档类型。
<html>	定义一个 HTML 文档
<title>	为文档定义一个标题
<body>	定义文档的主体
<h1> to <h6>	定义 HTML 标题
<p>	定义一个段落
 	定义简单的折行。
<hr>	定义水平线。
<!--...-->	定义一个注释

第二部分 WEB UI功能自动化

■ HTML常见标签说明

标签	描述
表单	
<form>	定义一个 HTML 表单，用于用户输入。
<input>	定义一个输入控件
<textarea>	定义多行的文本输入控件。
<button>	定义按钮。
<select>	定义选择列表（下拉列表）。
<optgroup>	定义选择列表中相关选项的组合。
<option>	定义选择列表中的选项。
<label>	定义 input 元素的标注。

第二部分 WEB UI功能自动化

■ HTML常见标签说明

标签	描述
框架	
<frame>	定义框架集的窗口或框架。
<frameset>	定义框架集。
<iframe>	定义内联框架。
图像	
<img>	定义图像。
<map>	定义图像映射。
表格	
<table>	定义一个表格
<th>	定义表格中的表头单元格。
<tr>	定义表格中的行。
<td>	定义表格中的单元。
<thead>	定义表格中的表头内容。
<tbody>	定义表格中的主体内容。

第二部分 WEB UI功能自动化

■ HTML常见标签说明

标签	描述
链接	
<a>	定义一个链接
<link>	定义文档与外部资源的关系。
<nav> New	定义导航链接
列表	
	定义一个无序列表
	定义一个有序列表
	定义一个列表项
<dl>	定义一个定义列表
<dt>	定义一个定义定义列表中的项目。
<dd>	定义定义列表中项目的描述。
<menu>	定义菜单列表。

HTML属性

- HTML 元素可以设置**属性**
- 属性可以在元素中添加**附加信息** about an element
- 属性一般描述于**开始标签**
- 属性总是以名称/值对的形式出现，**比如：name="value"**

属性实例

HTML 链接由 <a> 标签定义。链接的地址在 **href 属性**中指定：

实例

```
<a href="http://www.w3cschool.cc">This is a link</a>
```

第二部分 WEB UI功能自动化

■ HTML 常见属性说明

属性	描述
accesskey	设置访问元素的键盘快捷键。
class	规定元素的类名（classname）
contenteditable	规定是否可编辑元素的内容
contextmenu New	指定一个元素的上下文菜单。当用户右击该元素，出现上下文菜单
data-*	用于存储页面的自定义数据
dir	设置元素中内容的文本方向
draggable	指定某个元素是否可以拖动
dropzone New	指定是否将数据复制，移动，或链接，或删除
hidden	hidden 属性规定对元素进行隐藏。
id	规定元素的唯一 id
lang	设置元素中内容的语言代码。
style	规定元素的行内样式（inline style）
tabindex	设置元素的 Tab 键控制次序。
title	规定元素的额外信息（可在工具提示中显示）

第二部分 WEB UI功能自动化

■ Selenium简介

Selenium 是用于测试 Web 应用程序用户界面 (UI) 的常用框架。它是一款用于运行端到端功能测试的超强工具。您可以使用多个编程语言编写测试，且 能够在一个或多个浏览器中执行这些测试。

■ Selenium 1 + WebDriver = Selenium 2

Selenium 1 和 WebDriver 合并成一款性能更佳的产品 Selenium 2，发行于 2011

具有来自 WebDriver 的清晰面向对象API，并能以最佳的方式与浏览器进行交互

不使用 JavaScript 沙盒，它支持多种浏览器和多语言绑定。

Selenium 2 为下列程序提供驱动程序：

- Mozilla Firefox
- Google Chrome
- Microsoft Internet Explorer
- Opera
- Apple iPhone
- Android browsers

可使用 Java、C#、Ruby和 Python 编写测试

■ Selenium2（JAVA）基础

第二部分 WEB UI功能自动化

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");
```

第二部分 WEB UI功能自动化

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：
登录

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']"));

第二部分 WEB UI功能自动化

假设页面写成这样：
登录

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"));
```

第二部分 WEB UI功能自动化

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();

第二部分 WEB UI功能自动化

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();

第二部分 WEB UI功能自动化

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();

第二部分 WEB UI功能自动化

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();

第二部分 WEB UI功能自动化

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();

第二部分 WEB UI功能自动化

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 (Navigationand 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();

第二部分 WEB UI功能自动化

■ XPATH简介

- 它是一种用来确定XML（标准通用标记语言的子集）文档中某部分位置的语言(XPath 是一门在 XML 文档中查找信息的语言)
- 使用XPath的目的：为了在匹配XML文档结构时能够准确地找到某一个节点元素。可以把XPath比作文件管理路径，通过文件管理路径，可以按照一定的规则查找到所需要的文件；同样，依据XPath所制定的规则，也可以很方便地找到XML结构文档树中的任何一个节点。
- 每个XML文档都可看成是一棵树，该树与计算机中的树形文件夹非常类似

■ XPATH节点

七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档节点（或称为根节点）。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book>
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</bookstore>
```

文档节点（根节点）

属性节点

元素节点

第二部分 WEB UI功能自动化

■ 基本值

无父或无子的节点

Eg： JK. Rowling，"en“

■ 节点关系

- **父：**每个元素以及属性都有一个父

Eg：book元素是 title、author、year 以及 price 元素的父

- **同胞（Sibling）：拥有相同的父的节点**

Eg：title、author、year 以及 price 元素都是同胞

第二部分 WEB UI功能自动化

■ 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。

第二部分 WEB UI功能自动化

■ 路径表达式

➢ 使用属性

- //@ id：选择所有属性名为“id”的属性
- /class/student[@ id]：选择“class”元素下包含“id”属性的所有“student”元素
- /class/student[@ id= ‘ADR02’]：选择“class”元素下包含“id”属性且值为“ADR02”的所有“student”元素

➢ 使用函数

- //a[**text()**= ‘登录’]：选择所有文本属性为”登录”的链接a
- //a[**contains**(text(),’登录’)]：选择所有文本属性包含”登录”的链接a

第二部分 WEB UI功能自动化

■ 路径表达式

➢ 使用Xpath轴（Axes）

- preceding-sibling 选取当前节点之前的所有同级节点

Eg: /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

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

第二部分 WEB UI功能自动化

➢ Xpath轴说明

轴名称	结果
ancestor	选取当前节点的所有先辈（父、祖父等）。
ancestor-or-self	选取当前节点的所有先辈（父、祖父等）以及当前节点本身。
attribute	选取当前节点的所有属性。
child	选取当前节点的所有子元素。
descendant	选取当前节点的所有后代元素（子、孙等）。
descendant-or-self	选取当前节点的所有后代元素（子、孙等）以及当前节点本身。
following	选取文档中当前节点的结束标签之后的所有节点。
namespace	选取当前节点的所有命名空间节点。
parent	选取当前节点的父节点。
preceding	选取文档中当前节点的开始标签之前的所有节点。
preceding-sibling	选取当前节点之前的所有同级节点。
self	选取当前节点。

第二部分 WEB UI功能自动化

➢ SAT工具中Xpath使用原则

- 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')]

- 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；

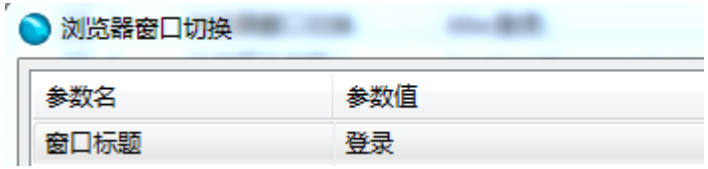
若父节点不可以，则找其父节点的父节点/兄弟节点...

第二部分 WEB UI功能自动化

■ 浏览器窗口切换

- 场景：点击某控件，打开一个新的浏览器窗口，且下个步骤需在新窗口中执行
- **示例：**SOP首页点击登录链接，打开登陆页面，切换窗口至新打开页面（案例：登录SOP）

使用浏览器窗口切换keyword：输入窗口title



- 类似的通用keyword还有：控件点击并切至新打开窗口

第二部分 WEB UI功能自动化

■ 浏览器框架切换

- IFRAME：HTML标签，会创建包含另外一个文档的内联框架（即行内框架）
- FRAME：HTML标签，定义frameset中的一个特定的窗口（框架）
- 操作包含在fram中的控件，首先需将当前识别的“主体”定位至该frame下
- iframe默认是自上向下, 自外向内逐层切换，若从内向外, 必须首先切换至默认主框架，然后再逐层切换。

第二部分 WEB UI功能自动化

- **浏览器框架切换**
- **示例**：SOP登录后首页点击商品管理-我的商品库，切换至苏宁商品库tab页（案例：切换苏宁发布商品）

  通用关键字：浏览器页面框架切换，不输入默认切至主框架下

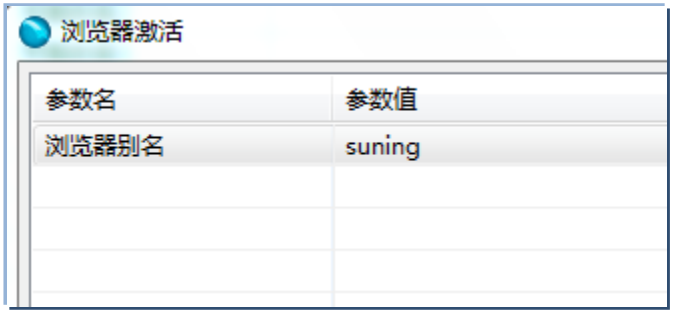
第二部分 WEB UI功能自动化

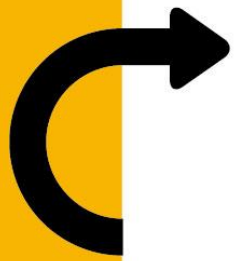
■ 浏览器激活

- 场景：打开多个系统测试，测试过程中需不同系统web页面来回切换
- 关键字：浏览器激活（根据别名激活某系统最近的浏览器窗口）

- **示例：**浏览器打开商品前台四级页面（别名：suning），浏览器打开后台配置页面（别名：acc），设置操作完成，切至前台页面查看页面变化；

  其中，切换用到浏览器激活关键字，输入参数-浏览器别名





第一部分 自动化测试基础

第二部分 WEB UI功能自动化

第三部分 接口功能自动化

第四部分 手机应用功能自动化

第五部分 基于关键字的脚本开发

第三部分 接口功能自动化

■ 接口测试定义

- 接口测试是测试系统组件间接口的一种测试。
- 接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。
- 测试的重点是要检查数据的交换，传递和控制管理过程，以及系统间的相互逻辑依赖关系等。

第三部分 接口功能自动化

■ 接口测试分类

- 系统与系统之间的调用

Eg：银行会提供接口供电子商务网站调用，或易付宝会提供接口给易购主站调用

- 上层服务对下层服务的调用，比如service层会调用DAO层的接口，而应用层又会调用服务层提供的接口，一般会通过服务之间的调用

Eg：注册用户时，会先调用用户查询的服务，查看该用户是否已经注册

PS：

接口测试先要了解是基于哪一种类型的接口，不同类型的接口测试方法可能是不一致的。

总体来说，不管是哪种类型，只要把被测接口当做服务方，而把测试手段当做客户方，最终目的是通过测试手段，去验证服务端满足了他声明提供的功能。

第三部分 接口功能自动化

■ 接口测试数据准备

➢ 使用对应数据库表中已有的数据

优点：方便且数据仿真性比较高

缺点：数据变动影响其他人员，且时效性不稳定，下次可能即不存在

有保障的做法：不动用原有数据前提下，每次测试前在对应表中增删改匹配的数据，但需要有对应数据库的增删改权限

➢ 直接通过调用其他API的方式准备测试数据

eg：测试删除收藏夹商品功能，准备待删除的收藏夹商品数据，可以调用增加收藏夹商品的接口直接生成测试数据

第三部分 接口功能自动化

■ 接口测试用例设计

➢ 正常情况，也即合法情况测试，主要为接口逻辑测试

- 保证接口测试的顺利进行，开发人员JavaDoc的输写或接口的详细说明文档定不可少。
- 一般情况，JavaDoc或接口详细说明文档需包含前提条件，业务逻辑，输入参数，输出参数的描述。
- 接口逻辑，一般可理解为接口的业务场景逻辑功能，根据文档所描述的业务逻辑，进行用例的设计，主要目标是测试在正常输入的情况下能得出正确的结果，测试用例的设计方法同黑盒测试。
- 测试的各个方面，包括数据的各个入口，路径，出口都应考虑周全，覆盖全部的业务逻辑场景。

第三部分 接口功能自动化

■ 接口测试用例设计

➢ 异常情况：非法参数

参数非法输入，意为制造异常的测试场景，查看测试的异常描述是否清晰

主要包括以下几方面：

- 参数必填项校验：参数输入空值
- 参数取值范围、类型输入非法

第三部分 接口功能自动化

■ 接口测试校验点

➢ 根据入参判断输出结果参数是否与接口说明文档一致

以HTTP/MQ协议响应消息为例

需判断xml/json格式的响应消息中参数是否正确，比如错误描述信息、错误码、成功与否的标识等

功能参数返回的值是否正确，一般和数据库表中某些字段进行比较。

➢ 根据入参判断涉及到的数据库表中数据变化是否与说明文档一致

比如删除收藏夹中某商品A的接口，需要校验收藏夹表中某用户下不存在商品A的数据

增加/更新操作同理

第三部分 接口功能自动化

■ 接口测试需了解

➢ **接口测试类型**：是具体的某协议接口，比如HTTP协议、MQ协议等；还是java接口

➢ **接口本身的功能需求说明**

- 实现了什么业务场景功能；
- 涉及到哪些系统-单系统或是依赖其他系统；
- 涉及到哪些具体的数据库及表；
- 输入参数有哪些，具体意义，入参数据的取值范围及来源，入参与数据库表中字段的对应关系；
- 输出参数有哪些，具体意义，参数结果与数据库表中字段的对应关系，或者参数结果与输入参数的对应关系，或者除输出结果外，其他隐形的结果，比如数据库中数据变化；

➢ **根据不同的接口类型，还需要针对性的了解**

第三部分 接口功能自动化

HTTP简介

- HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。
- HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP工作原理

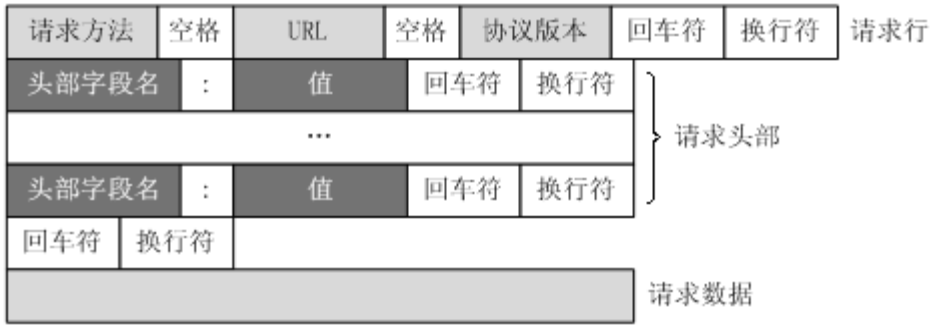
- HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。

  Web服务器有：Apache服务器，IIS服务器（Internet Information Services）等。
- Web服务器根据接收到的请求后，向客户端发送响应信息。
- HTTP默认端口号为80，但也可改为8080或者其他端口。

■ HTTP消息结构

➢ 客户端请求消息

请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据4个部分组成



第三部分 接口功能自动化

■ HTTP消息结构

➢ 服务器响应消息

HTTP响应也由三个部分组成，分别是：状态行、消息报头、响应正文

```
HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 122

<html>
<head>
<title>Wrox Homepage</title>
</head>
<body>
<!-- body goes here -->
</body>
</html>
```

状态行

消息报头

空行

下面的就是响应正文了

第三部分 接口功能自动化

HTTP请求方法

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

第三部分 接口功能自动化

■ HTTP请求方法

GET请求：

login.action?name=hyddd&password=idontknow&verify=%E4%BD%E5%A5%BD

- 以?来分隔URL和数据；
- 以& 来分隔参数；
- 如果数据是英文或数字，原样发送；
- 如果数据是中文或其它字符，则进行BASE64编码

操作方式	数据位置	明文密文	数据安全	长度限制	应用场景
GET	HTTP包头	明文	不安全	长度较小	查询数据
POST	HTTP正文	密文	安全	支持较大数据传输	修改数据

第三部分 接口功能自动化

■ HTTP状态码

浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

常见的HTTP状态码(HTTP Status Code)：

- 200 - 请求成功
- 301 - 资源（网页等）被永久转移到其它URL
- 404 - 请求的资源（网页等）不存在
- 500 - 内部服务器错误

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

■ HTTP消息体格式

➢ XML格式

XML被设计用来传输和存储数据，标签没有预定义，需要自行定义，可扩展，具备自我描述性。

树形结构，格式参考XPATH介绍

```
<?xml version="1.0" encoding="utf-8"?>
 <MbfService xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
  <input1>
    <MbfHeader>
    </MbfHeader>
    <MbfBody>
      <sysHeader>
      </sysHeader>
      <appHeader>
      </appHeader>
      <body>
        <ecoType>140000000010</ecoType>
        <userName>11111252cc1</userName>
        <password></password>
        <mobileNum>N/A</mobileNum>
        <mobileNumStat>N/A</mobileNumStat>
        <bindEmail>oop10010@127.com</bindEmail>
        <bindEmailVerifyStat></bindEmailVerifyStat>
        <roleType></roleType>
```

第三部分 接口功能自动化

■ HTTP消息体格式

➢ JSON格式

JSON 指的是 JavaScript 对象表示法（**J**ava**S**cript **O**bject **N**otation），轻量级的文本数据交换格式，独立于语言 *，具有自我描述性，更易理解；

JSON 数据的书写格式是：名称/值对。

JSON 值可以是：

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（true 或 false）
- 数组（在方括号中）
- 对象（在花括号中）
- null

```
{
    "promotions": [
        {
            "promotion": {
                "ordNo": "D50062580",
                "storeNo": "7611",
                "billType": "DHWL",
                "eCoupon": {
                    "eCouponAmount": "0",
                    "ruleCode": "",
                    "activityCode": "",
                    "startDate": "",
                    "endDate": ""
                },
                "zeroCouponAmount": "0"
            }
        }
    ]
}
```

第三部分 接口功能自动化

■ SAT的HTTP协议自动化

➢ 基于HTTPClient封装HTTP协议测试的关键字

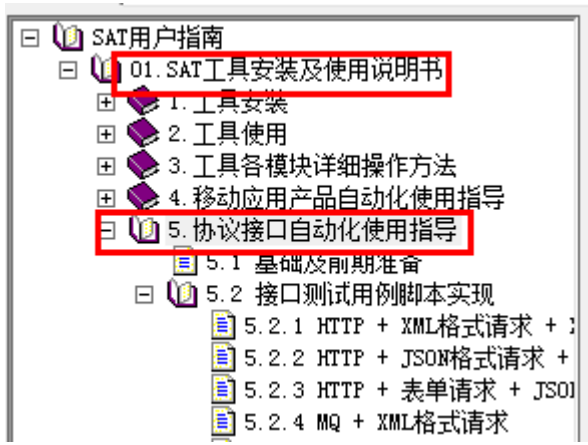
HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

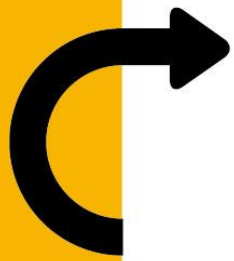
主要功能：

- 实现了所有 HTTP 的方法（GET,POST,PUT,HEAD 等）
- 支持自动转向
- 支持 HTTPS 协议
- 支持代理服务器等

➢ SAT协议关键字使用方法

请参考SAT帮助文档





第一部分 自动化测试基础

第二部分 WEB UI功能自动化

第三部分 接口功能自动化

第四部分 手机应用功能自动化

第五部分 基于关键字的脚本开发

第四部分 手机应用功能自动化

■ 手机APP分类

➢ Native App

- 原生应用，也即客户端，是针对不同手机系统单独开发的本地应用，使用需要先下载到手机并安装
- 在技术实现上一般采用针对操作系统的特定语言进行编写

  如：使用Objective-c开发IOS应用，使用Java+Android开发android应用。

➢ Web App

- Web应用，简单的说就是一个触屏版的网站。
- Web应用完全用HTML、JavaScript和CSS等Web技术开发，通过移动设备的浏览器来访问。

➢ Hybrid App

- 混合应用，是一种介于Native App、Web App之间的App，虽看上去是一个Native App，但只是一个UI WebView，访问的是一个Web App。
- Hybrid App实质是伪造一个浏览器的apk/ipa原生程序，并运行了一个Web APP。
- Hybrid App兼具“Native App良好用户交互体验的优势”和“Web App跨平台开发的优势”。

第四部分 手机应用功能自动化

■ Native App、Web App和Hybrid App的比较

	Native App	Web App	Hybrid App
开发语言	原生语言 ObjectC、Java、.net等	网页语言 HTML5+JS	网页或原生语言
跨平台性	低	高	高
设别能力	高	低	高
开发难度	高	低	低
应用体验	好	差	较好
向后兼容	差	好	好

第四部分 手机应用功能自动化

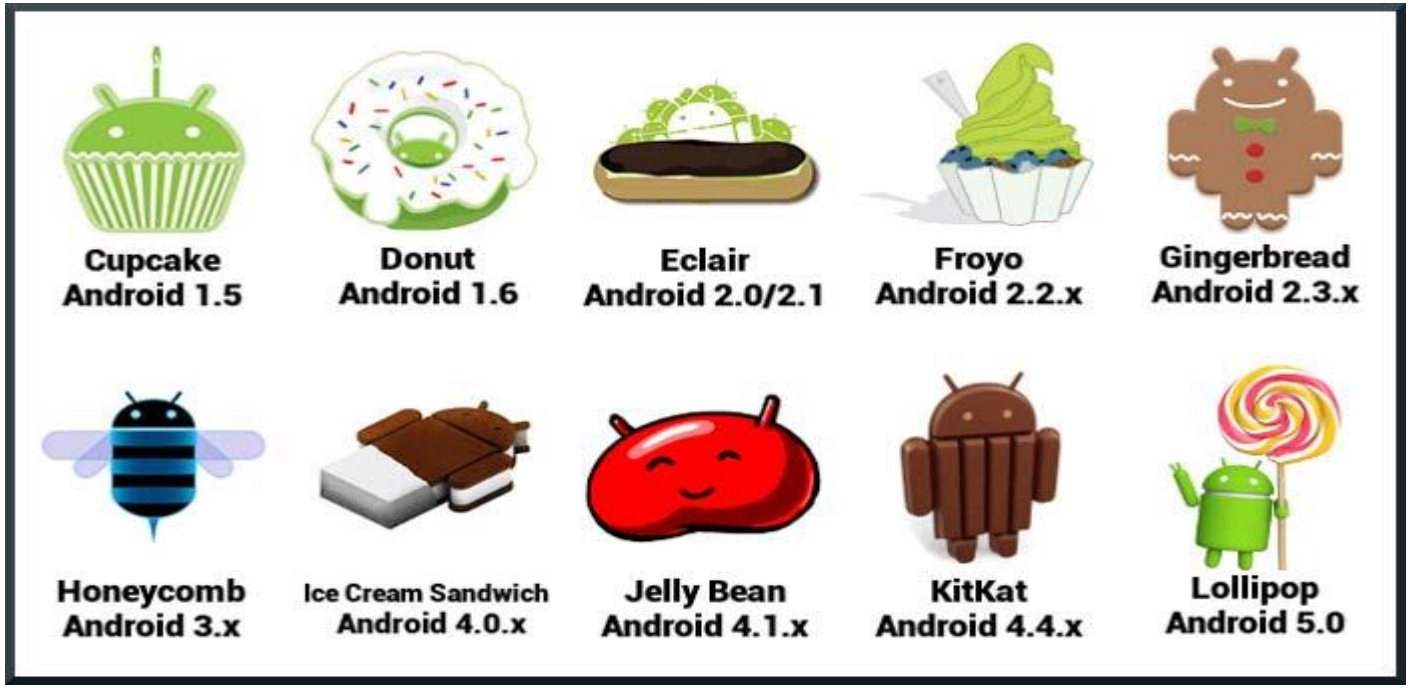
■ 手机APP自动化测试

- **功能方面：**主要指app是否完成了设计的所有功能
- **兼容性方面：**考虑手机的版本、型号、分辨率、网络等差异性
- **系统交互：**电话短信干扰，低电量/push/充电提醒等易构造的中断场景
- **性能、稳定性方面：**靠工具来实现应用启动以及运行时CPU占用、内存占用、耗电量、耗流量、安装卸载，异常捕获crash等
- **深度遍历：**自定义遍历引擎探测深度，自动分析界面可操作控件信息，并触发系统事件使得应用界面切换，自动抓取菜单树及界面上的信息，和预期结果进行比对，进而判断界面是否存在异常。

Android概述

- Android 是一个开源的，基于 Linux 的移动设备操作系统，如智能手机和平板电脑
- Android 是由谷歌及其他公司带领的开放手机联盟开发的

Android版本发展历史



第四部分 手机应用功能自动化

■ Android概述

➢ **Android API级别：**API 级别是一个用于唯一标识 API 框架版本的整数，由某个版本的 Android 平台提供

平台版本	API 等级	VERSION_CODE
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD

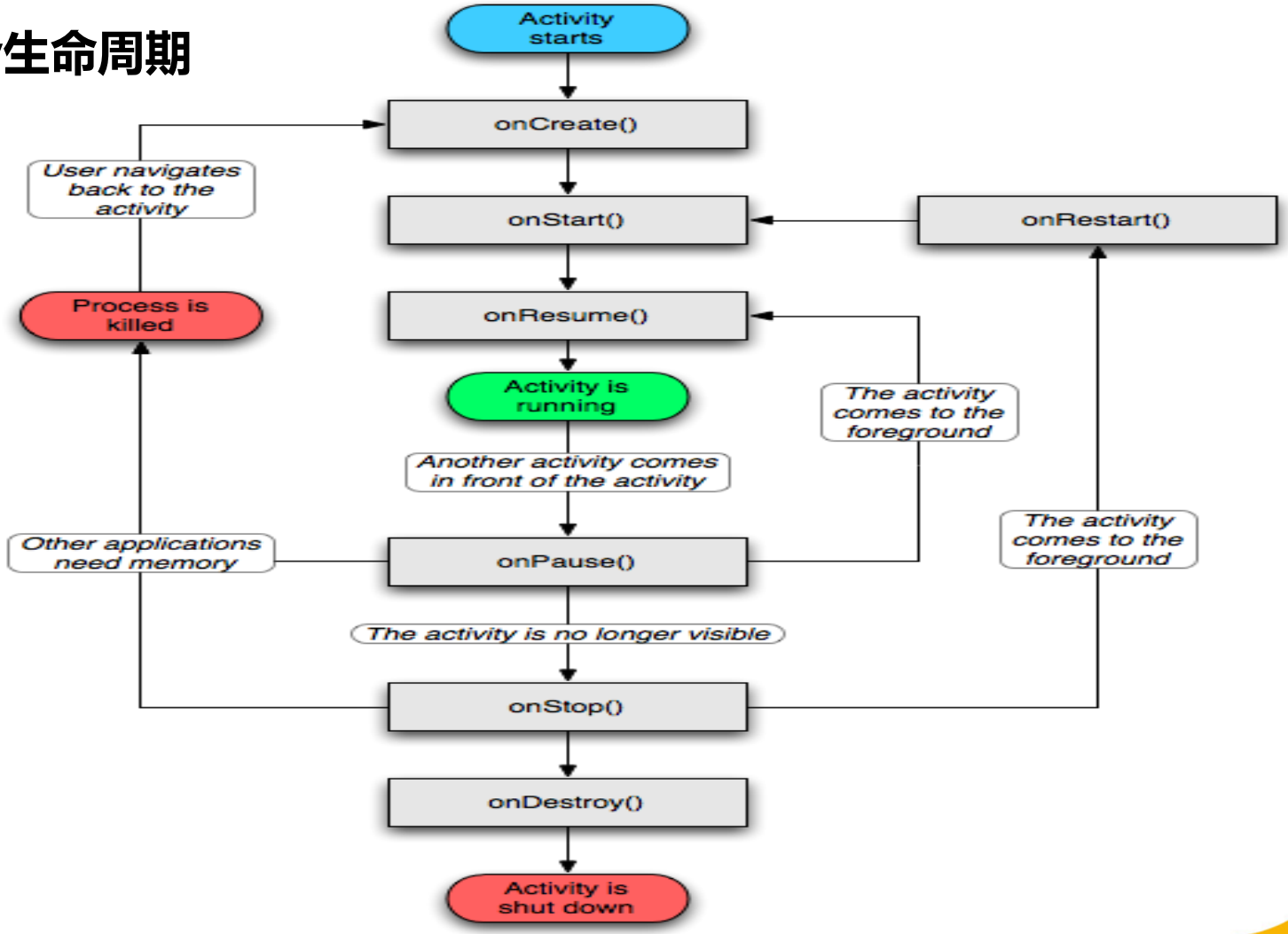
第四部分 手机应用功能自动化

■ Android应用程序组件

组件	描述
Activities（活动）	• 描述UI，并且处理用户与机器屏幕的交互 • 一个活动标识一个具有用户界面的单一屏幕 • 当应用程序拥有多余一个活动，其中的一个会被标记为当应用程序启动的时候显示 eg：一个邮件应用程序可以包含一个活动用于显示新邮件列表，另一个活动用来编写邮件。
Services(服务)	• 运行在后台，执行长时间操作的组件，是处理与应用程序关联的后台操作 eg：服务可以是用户在使用不同的程序时在后台播放音乐，或者在活动中通过网络获取数据但不阻塞用户交互。
Broadcast Receivers(广播接收器)	• 处理Android操作系统和应用程序之间的通信
Content Providers(内容提供者)	• 处理数据和数据库管理方面的问题

第四部分 手机应用功能自动化

- **Activity生命周期**



■ Activity生命周期

三个循环：

- **整个的生命周期，从onCreate(Bundle)开始到onDestroy()结束**

Activity在onCreate()设置所有的“全局”状态，在onDestory()释放所有的资源。

例如：某个Activity有一个在后台运行的线程，用于从网络下载数据，则该Activity可在onCreate()中创建线程,在onDestory()中停止线程。

- **可见的生命周期，从onStart()开始到onStop()结束**

Activity在屏幕上，有可能不在前台，不能和用户交互。在两个接口之间，需要保持显示给用户的UI数据和资源等。

例如：可以在onStart中注册一个IntentReceiver来监听数据变化导致UI的变动，当不再需要显示时候，可以在onStop()中注销它。onStart()，onStop()都可以被多次调用，因为Activity随时可以在可见和隐藏之间转换。

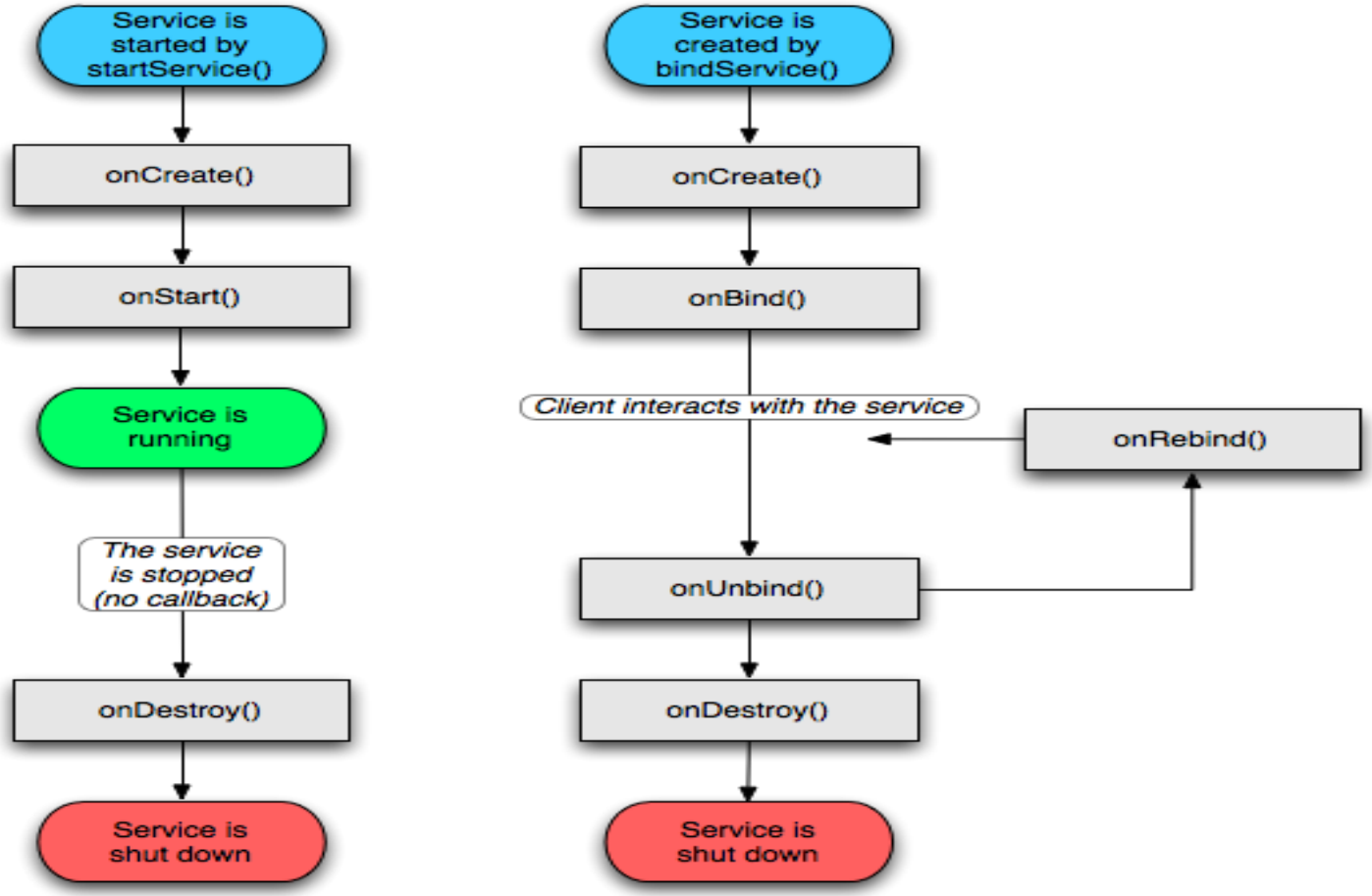
- **前台的生命周期，从onResume()开始到onPause()结束**

该Activity处于所有Activity的最前面，和用户进行交互。Activity可以经常性地在resumed和paused状态之间切换

例如：当设备准备休眠时，当一个Activity处理结果被分发时，当一个新的Intent被分发时

第四部分 手机应用功能自动化

- Service生命周期



Service生命周期

- 通过start启动一个service，会触发onCreate()和onStart()操作，但如果该Service已在系统中存在，则onCreate()不会被再次调用，它只在Service第一次启动时触发。
- 通过start启动的Service会一直运行，直到通过stop停止它。
- 多次通过start启动某个服务会导致服务的onStart()被多次调用，但仍只有一个实例，因此无论启动多少次，停止它只需调用一次stop就可以了。
- 通过bind来获得一个服务的链接，如果系统中还没有该服务，则会触发onCreate函数新创建一个服务，连接建立时onBinder会被触发。
- 通过bind获得服务的链接一直会保持到通过unbind断掉它
- 通过bind启动的链接，会一直存在到没有任何客户端与它保持连接为止，如果某个链接被客户端主动断掉只会是Service的链接数减1，当减至0的时候这个Service就会被销毁。

第四部分 手机应用功能自动化

■ Android附件组件

组件	描述
Fragments（碎片）	代表活动中的一个行为或者一部分用户界面。
Views （视图）	绘制在屏幕上的UI元素，包括按钮，列表等。
Layouts（布局）	控制屏幕格式，展示视图外观的View的继承。
Intents	组件间的消息连线。
Resources	外部元素，例如字符串资源、常量资源及图片资源等。
Manifest	应用程序的配置文件。

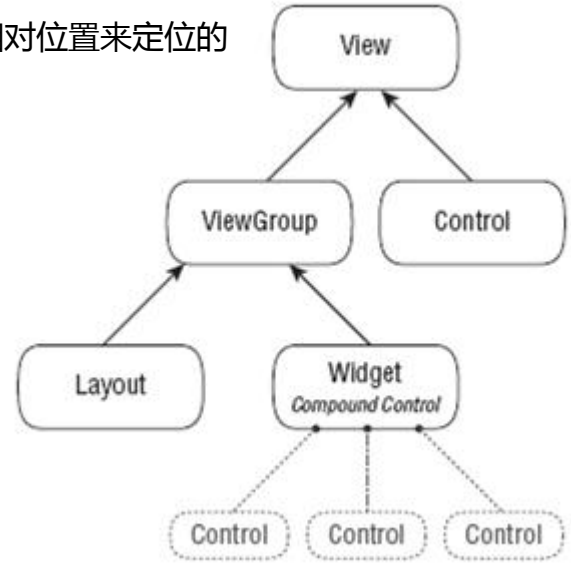
第四部分 手机应用功能自动化

■ Android APP UI构成

- ➢ Android中，所有的可视化组件都是继承自View(视图)类，用户通过View和ViewGroup或者扩展自他们的类来构建用户界面
- ➢ Control用来实现相对比较简单的功能
- ➢ **Layout介绍**

- LinearLayout 线性布局，包含在LinearLayout里面的控件按顺序排列成一行或者一列。
- RelativeLayout 相对布局，它是依靠与父容器或同在一容器中其它控件的相对位置来定位的
- TableLayout 表格布局，类似于HTML的Table。通过TableRow来定义一行
- AbsoluteLayout 绝对布局，就是Android不提供任何布局控制，

而是由我们自己通过X坐标，Y坐标来控制组件的位置



■ Android APP UI构成

➢ **Widget 用来组合控件和构建更加复杂的控件（如用户自定义组件）**

- **TextView：** 标准的只读文本label。它支持多行显示、字符串格式化和文本自动换行。
- **EditText：** 可编辑的文本输入框。它支持多行输入和文字换行。
- **ListView：**

一个ViewGroup，以列表的方式创建和管理一组显示项。

标准的ListView使用TextView来显示每一个字符串数组内的值。

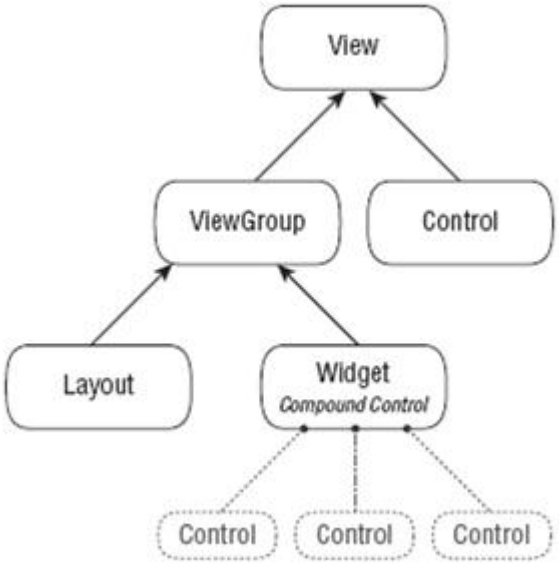
- **Spinner：**

组合控件，显示一个TextView和一个关联的ListView，

用来从一个列表中选择一项并显示选择项在TextView中。

它还有一个button，当按下时显示一个选择框。

- **Button：**标准的按钮。
- **CheckBox：**两种状态的button，代表checked或unchecked。
- **RadioButton：**单选按钮。



■ Android APP UI构成

➢ **Widget 用来组合控件和构建更加复杂的控件（如用户自定义组件）**

- **TextView：** 标准的只读文本label。它支持多行显示、字符串格式化和文本自动换行。
- **EditText：** 可编辑的文本输入框。它支持多行输入和文字换行。
- **ListView：**

一个ViewGroup，以列表的方式创建和管理一组显示项。

标准的ListView使用TextView来显示每一个字符串数组内的值。

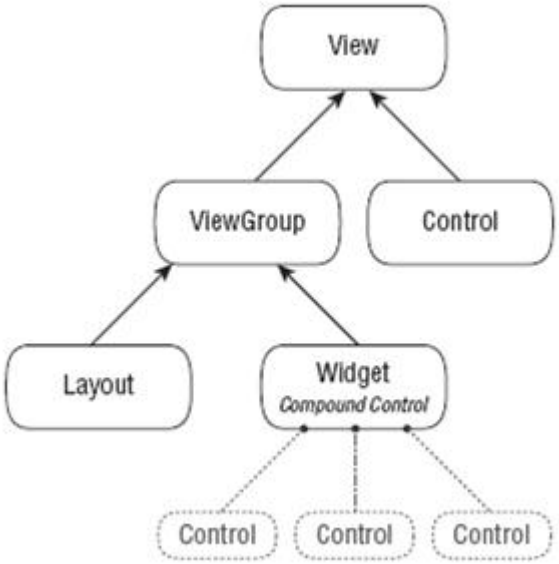
- **Spinner：**

组合控件，显示一个TextView和一个关联的ListView，

用来从一个列表中选择一项并显示选择项在TextView中。

它还有一个button，当按下时显示一个选择框。

- **Button：**标准的按钮。
- **CheckBox：**两种状态的button，代表checked或unchecked。
- **RadioButton：**单选按钮。



第四部分 手机应用功能自动化

■ Android Native APP UI控件定位

➢ **android自带的原生APP控件定位工具有两种：**uiautomatorviewer 和 hierarchyviewer

DUMP	hierarchyviewer	uiautomatorviewer
root	需要	
API		16以上
android版本		4.2以上
获取方式	通过socket连接手机端的viewserver获取数据，获得数据后，逐个遍历每一行数据处理每一行数据，然后存放在ViewNode对象中，每个view的数据都保存在内存中。	通过adb执行shell命令，存放在手机端的/system/bin/uiautomator脚本会被执行，在/data/local/tmp/下生成uidump.xml文件，然后adb pull将xml文件复制到本地来，然后程序读取xml生成树形结构
优点	1.无版本限制。2.可获得动态数据	1.数据的分析过程交给了手机端，减少了PC端的处理过程，客户端直接读取XML文件，所提供的信息短小精悍。2.无需root。3.可获得package名。4.可直接判断控件是否可点击。对话框的坐标获得正确
缺点	1.无法获得package和activity名。2.需要root。3.属性多而无用。4.对话框的坐标有缺陷	1.API限制。2.无法获得activity名。3.无法获得动态界面的数据
效率	根据软件的实际测试，获得一个界面的所有节点和图片所有时间和为：10s左右	由于读取xml的数据快，所以略有优势。Time:4左右.获得图片和控件信息
总结：在api为16以上的设备推荐使用uiautomator的dump方式获取数据。		

第四部分 手机应用功能自动化

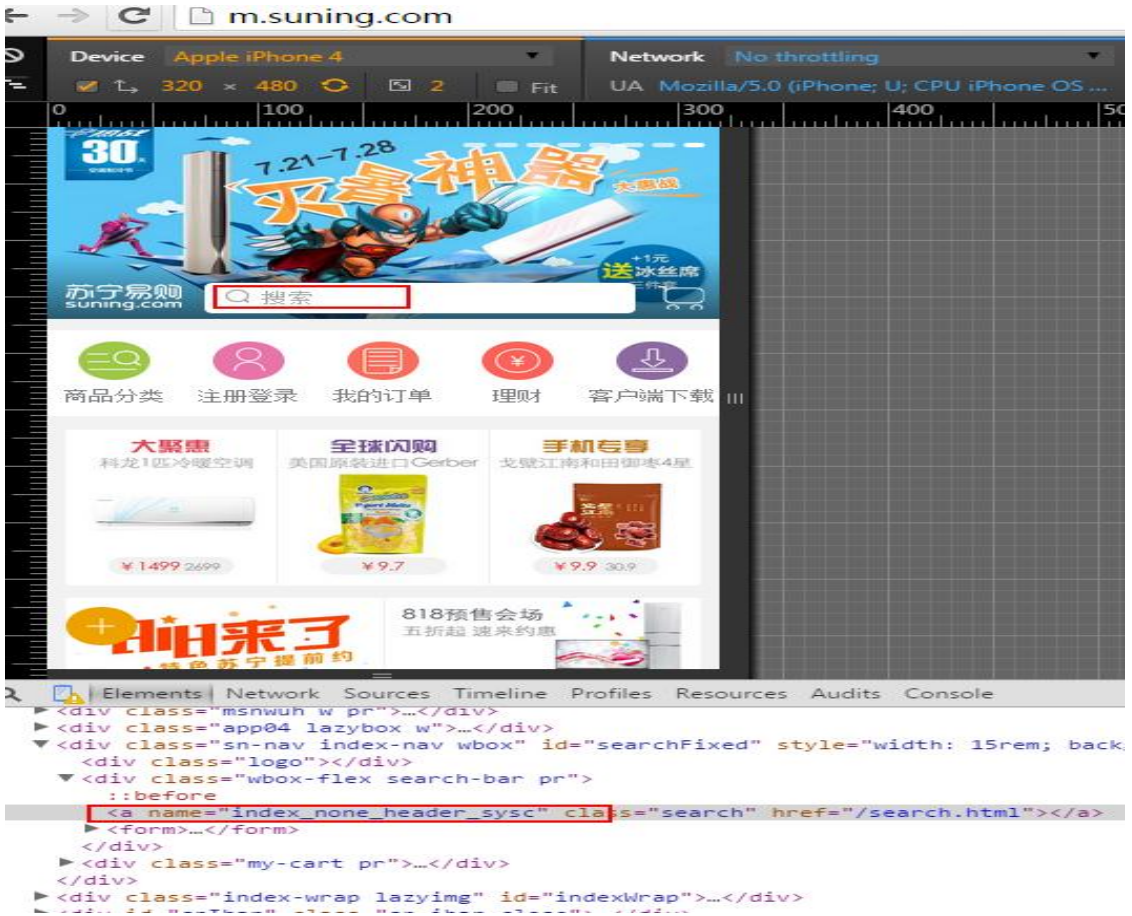
■ Android Native App UI控件定位

- ➢ uiautomatorviewer

第四部分 手机应用功能自动化

■ Android Wap App UI控件定位

使用chrome浏览器打开页面
F12- chrome debugger定位
同pc-web



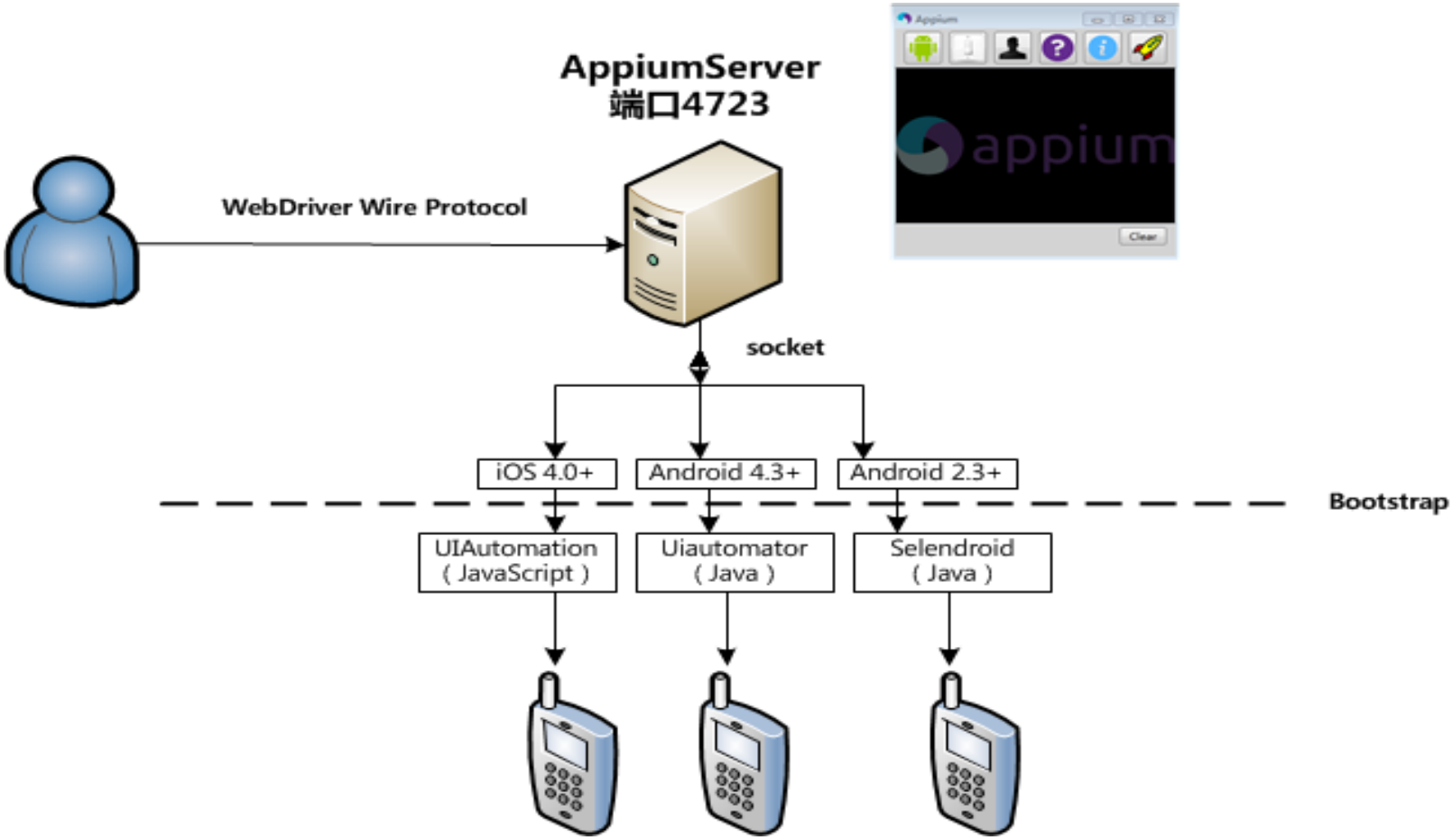
第四部分 手机应用功能自动化

■ SAT基于Appium开源框架进行应用功能自动化测试

➢ 什么是Appium?

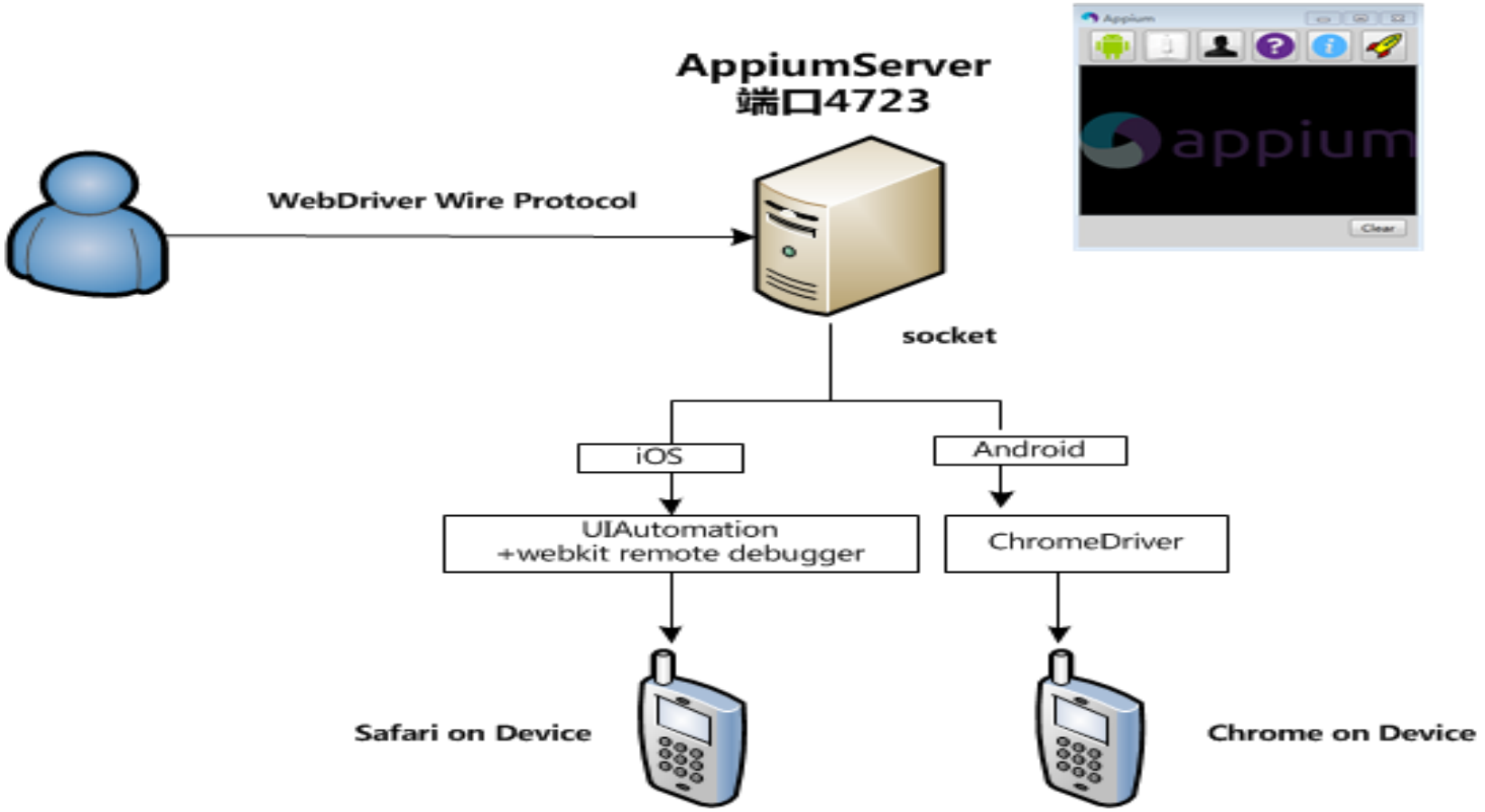
- Appium是一个开源的自动化测试框架，支持原生应用、混合应用和移动Web应用自动化测试;
- 通过WebDriver Protocol驱动iOS和Android、Firefox OS应用;
- 用Appium自动化测试不需要重新编译App;
- 支持很多语言来编写测试脚本，Java、Javascript、PHP、Python、C#、Ruby等主流语言;
- Server也是跨平台的，你可以使用Mac OS X、Windows或者Linux

- Appium之Native原理



第四部分 手机应用功能自动化

- **Appium之WAP原理**



第四部分 手机应用功能自动化

■ Appium常用接口-1

接口	说明
currentActivity()	获取当前activity,比如（.ApiDemos）
isAppInstalled(String bundleId)	根据bundleId来判断该应用是否已经安装
installApp(String appPath)	安装app，appPath为应用的本地路径
removeApp(String bundleId)	卸载app.bundleId在android中代表的是报名，而在ios中有专门的bundleId号。
closeApp()	关闭应用，其实就是按home键把应用置于后台
launchApp()	启动应用
resetApp()	先closeApp然后在launchAPP
pullFile(String remotePath)	将设备上的文件pull到本地硬盘上
pushFile(String remotePath, byte[] base64Data)	将字符数组用64位格式写到远程目录的某个文件中。也可以理解为把本地文件push到设备上。
setNetworkConnection(NetworkConnectionSetting connection)	设置手机的网络连接状态，可以开关蓝牙、wifi、数据流量。通过NetworkConnectionSetting中的属性来设置各个网络连接的状态。
sendKeyEvent(int key)	按下某个键，具体哪个键由key值决定，key值定义在AndroidKeyCode类中
sendKeyEvent(int key, Integer metastate)	按下某个键的同时按下附加键（Ctrl/Alt/Shift等），具体是哪些键，由key值（AndroidKeyCode类中定义）和metastate（AndroidKeyMetastate类中定义）决定。

第四部分 手机应用功能自动化

■ Appium常用接口-2

接口	说明
tap(int fingers, WebElement element, int duration)	点击element控件中心点按下，duration*5毫秒秒后松开，如此重复fingers次。
tap(int fingers, int x, int y, int duration)	点击(x,y)点按下，duration*5毫秒后松开，如此重复fingers次。
swipe(int startx, int starty, int endx, int endy, int duration)	从(startx,starty)滑到（endx,endy），分duration步滑，每一步用时是5毫秒。
press(WebElement el)	在控件上执行press操作。
press(int x, int y)	在坐标为（x,y）的点执行press操作
moveTo(WebElement el)	以el为目标，从另一个点移动到该目标上
waitAction(int ms)	等待ms秒
longPress(WebElement el)	控件长按
rotate(ScreenOrientation orientation)	设置屏幕横屏或者竖屏
context(String name)	设置上下文
scrollTo(String text)	滚动到某个text属性为指定的字符串的控件
pinch(WebElement el)	2个手指操作控件，从对角线向中心点滑动。
zoom(WebElement el)	与pinch(el)的动作刚好相反。两个手指由控件的中心点慢慢向控件的左顶点后右底点滑动。

第四部分 手机应用功能自动化

■ Android 调试桥(adb)常用命令-1

说明	命令
显示系统中全部Android平台	android list targets
显示系统中全部AVD（模拟器）	android list avd
创建AVD（模拟器）	android create avd --name 名称 --target 平台编号
启动模拟器	emulator -avd 名称 -sdcard ~/名称.img (-skin 1280x800)
删除AVD（模拟器）	android delete avd --name 名称
创建SDCard	mksdcard 1024M ~/名称.img
AVD(模拟器)所在位置	Linux(~/.android/avd) Windows(C:\Documents and Settings\Administrator\.android\avd)
启动DDMS	ddms
显示当前运行的全部模拟器	adb devices
对某一模拟器执行命令	Adb -s 模拟器编号 命令
安装应用程序	adb install -r 应用程序.apk
卸载apk包	adb uninstall apk包的主包名
获取模拟器中的文件	adb pull <remote> <local>
向模拟器中写文件	adb push <local> <remote>

第四部分 手机应用功能自动化

■ Android 调试桥(adb)常用命令-2

说明	命令
进入模拟器的shell模式	adb shell
查看adb命令帮助信息	adb help
在命令行中查看LOG信息	adb logcat -s 标签名
删除系统应用	adb remount （重新挂载系统分区，使系统分区重新可写）。 adb shell cd system/app rm *.apk
获取管理员权限	adb root
启动activity	adb shell am start -n 包名/包名＋类名（-n 类名,-a action,-d date,-m MIME-TYPE,-c category,-e 扩展数据等）。
发布端口	adb forward tcp:5555 tcp:8000
复制一个文件或目录到设备或模拟器上	adb push <source> <destination></destination></source>
从设备或模拟器上复制一个文件或目录	adb pull <source> <destination></destination></source>
查看bug报告	adb bugreport
记录无线通讯日志	adb shell logcat -b radio
获取设备的ID和序列号	adb get-product adb get-serialno

第四部分 手机应用功能自动化

■ Android 自带的命令行工具(Monkey)常用命令 -1

- Monkey可以运行在模拟器或真实设备上，并产生一系列随机的用户事件（点击、触摸、手势以及系统级别的事件），一般用来进行压力测试或稳定性测试。

monkey [-p ALLOWED_PACKAGE [-p ALLOWED_PACKAGE] ...]
[-c MAIN_CATEGORY [-c MAIN_CATEGORY] ...]
[--ignore-crashes] [--ignore-timeouts]
[--ignore-security-exceptions]
[--monitor-native-crashes] [--ignore-native-crashes]
[--kill-process-after-error] [--hprof]
[--pct-touch PERCENT] [--pct-motion PERCENT]
[--pct-trackball PERCENT] [--pct-syskeys PERCENT]
[--pct-nav PERCENT] [--pct-majornav PERCENT]
[--pct-appswitch PERCENT] [--pct-flip PERCENT]
[--pct-anyevent PERCENT]
[--pkg-blacklist-file PACKAGE_BLACKLIST_FILE]
[--pkg-whitelist-file PACKAGE_WHITELIST_FILE]
[--wait-dbg] [--dbg-no-events]
[--setup scriptfile] [-f scriptfile [-f scriptfile] ...]
[--port port]
[-s SEED] [-v [-v] ...]
[--throttle MILLISEC] [--randomize-throttle]
COUNT

第四部分 手机应用功能自动化

■ Android 自带的命令行工具(Monkey)常用命令 -2

四大类：

➢ 常用选项

- --help：打印帮助信息
- -v：指定打印信息的详细级别，一个 -v增加一个级别 ， 默认级别为 0

➢ 事件选项

- -s：指定产生随机事件种子值，相同的种子值产生相同的事件序列。如： -s 200
- --throttle：每个事件结束后的间隔时间——降低系统的压力（如不指定，系统会尽快的发送事件序列）。如：--throttle 100
- --pct-touch：指定触摸事件的百分比，如：--pct-touch 5%
- --pct-motion <percent> （滑动事件）
- --pct-trackball <percent> （轨迹球事件）
- --pct-nav <percent> （导航事件 up/down/left/right）
- --pct-majornav <percent>（主要导航事件 back key 、 menu key）
- --pct-syskeys <percent>（系统按键事件 Home 、Back 、startCall 、 endCall 、 volumeControl）
- --pct-appswitch <percent> （activity之间的切换）
- --pct-anyevent <percent>（任意事件）

第四部分 手机应用功能自动化

■ Android 自带的命令行工具(Monkey)常用命令 -3

四大类：

➢ 约束选项

- -p：指定有效的package（如不指定则对系统中所有package有效），一个-p 对应一个有效package， 如：-p com.ckt -p com.ckt.asura；
- -c：activity必须至少包含一个指定的category，才能被启动，否则启动不了；

➢ 调试选项

- --dbg-no-events：初始化启动的activity，但是不产生任何事件。
- --hprof：指定该项后在事件序列发送前后会立即生成分析报告 —— 一般建议指定该项。
- --ignore-crashes：忽略崩溃
- --ignore-timeouts：忽略超时
- --ignore-security-exceptions：忽略安全异常
- --kill-process-after-error：发生错误后直接杀掉进程
- --monitor-native-crashes：跟踪本地方法的崩溃问题
- --wait-dbg：知道连接了调试器才执行monkey测试。

第四部分 手机应用功能自动化

■ Android 自带的命令行工具(Monkey)常用命令 -4

举例：

➢ 一个简单的命令

- adb shell monkey -p com.xy.android.junit -s 500 -v 10000

➢ 一般工作中为了保证测试数量的完整进行，我们一般不会在发生错误时立刻退出压力测试

- adb shell monkey -p com.xy.android.junit -s 500 --ignore-crashes --ignore-timeouts --monitor-native-crashes -v -v 10000 > E:\monkey_log\java_monkey_log.txt

PS：monkey作用的包：com.ckt.android.junit

产生时间序列的种子值：500

忽略程序崩溃 、 忽略超时 、 监视本地程序崩溃 、 详细信息级别为2 ， 产生 10000个事件 。

第四部分 手机应用功能自动化

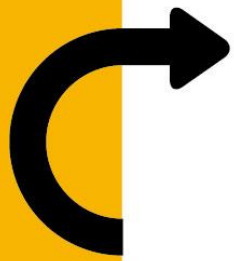
■ Android 自带的命令行工具(Monkey)常用命令 -4

➢ Monkey监控并特殊处理的3个事件：

- 如果指定测试包时，限制测试在指定的包中；
- 如果应用crash或存在未捕获的异常，monkey停止并报告错误；
- 如果应用产生ANR（*application not responding*）错误，monkey停止并报告错误。

➢ 产生ANR的两个条件：

- 线程响应超过5s；
- HandleMessage回调函数超过10s



第一部分 自动化测试基础

第二部分 WEB UI功能自动化

第三部分 接口功能自动化

第四部分 手机应用功能自动化

第五部分 基于关键字的脚本开发

第五部分 基于关键字的脚本开发

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。

第五部分 基于关键字的脚本开发

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。

第五部分 基于关键字的脚本开发

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**(SAT工具使用的脚本方式)**

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。

第五部分 基于关键字的脚本开发

■ 关键字驱动的思想

➢ 界面元素名与测试工具定义对象名的分离

被测程序和生成的测试脚本之间增加模型曾，将界面上的所有元素映射成对应的逻辑对象，测试针对逻辑对象进行。界面元素的改变只会影响映射表，而不影响测试

➢ 执行动作与具体实现细节的分离

把测试执行的动作和测试具体实现细节分离，用关键字描述测试执行动作，只说明该步测试执行什么动作而不管测试工具具体怎样执行。这种分离使得关键字对于实现细节不敏感，有利于测试在不同工具间的移植

➢ 测试脚本于测试数据的分离

把测试执行过程中所需的测试数据从脚本中提取出来，在运行时由测试控制模块从数据库中读取预先定制好的数据，这样测试脚本和测试数据可以独立维护

第五部分 基于关键字的脚本开发

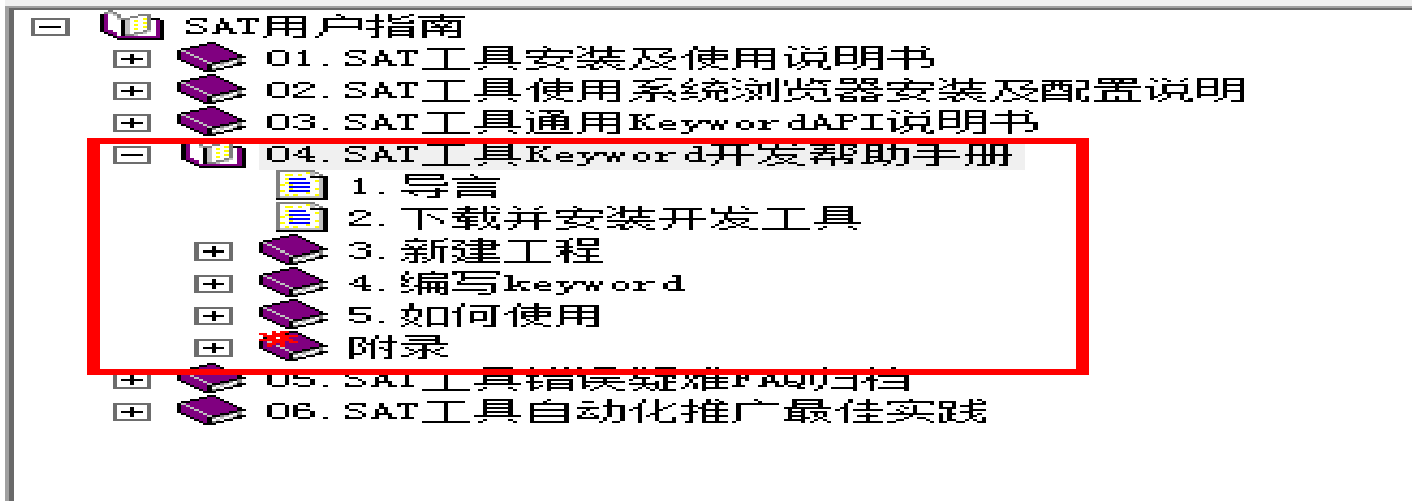
■ 何时需要编写业务关键字？

- 工具暂不提供的关键字，比如业务类的上传下载插件功能
- 需要逻辑判断的业务，例如不同的选择产生不同的分支
- 动态的页面定位符，随机的数据场景
- 多次被使用的操作步骤集，比如用户登录，加入购物车等

第五部分 基于关键字的脚本开发

■ SAT如何编写编写业务关键字

请下载SAT工具，参考SAT帮助文档《04.SAT工具keyword开发帮助手册》



Thanks!