填空

1、根据项目流程阶段划分软件测试：（单元测试）、（集成测试）、（系统测试）、（验收测试）

（单元测试）对程序中的单个子程序或具有独立功能的代码段进行测试的过程。

（集成测试）是在单元测试的基础上，先通过单元模块组装成系统或子系统，再进行测试。重点是检查模块之间的接口是否正确。

（系统测试）是针对整个产品系统进行的测试，验证系统是否满足需求规格的定义，以及软件系统的正确性和性能等是否满足其需求规格的要求。

（验收测试）是部署软件之前的最后一个测试阶段。验收测试的目的是确保软件准备就绪，向软件购买者展示该软件系统能够满足用户的需求。

2、（白盒测试）与（黑盒测试），主要是根据软件测试工作中对软件代码的可见程度进行的划分。

3、性能测试是通过自动化的测试工具模拟多种（正常）、（峰值）以及（异常负载条件）来对系统的各项性能指标进行测试。软件的性能包括很多方面，主要有（时间性能）和（空间性能）两种。

5、TestNG数据提供者使用（ @DataProvider ）注解来实现，返回类型是（ Object[][] ）。

6、Selenium使用（@FindBy ）注解获取页面元素

7、Selenium通过（ isSelected() ）方法验证一个复选按钮是不是被选中或者未被选中

8、Selenium元素定位类型包含如下8种，它们都位于By类中并且都是静态方法。它们分别是：By.id()，By.name()， By.className()，By.linkText()，By.partialLinkText()，（ ）、（ ）（ ）

By.id(), By.name(), By.tagName(), By.className(), By.linkText(), By.partialLinkText(), By.xpath, By.cssSelector().

9、Selenium组件有Selenium-RC、（ Selenium WebDriver ）、（ Selenium IDE ）、（Selenium Grid ）。

10、Log4j优先级从高到低分别是：（ ）、ERROR、WARN、INFO、DEBUG， TRACE

由低到高

TRACE，DEBUG，INFO，WARN，ERROR，FATAL ALL OFF

12、findElements(arg0)返回的类型是（ ）。findElement()返回的类型是（We）

14、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。

15、自动化框架常见的模式：（ 数据驱动框架 ）、（ 关键字驱动框架 ）、（ 行为驱动框架 ）、混合型测试框架

16、TestNG基于J2SE5.0的（ 注解 ）特性的而构建的轻量级的单元测试框架结构。

17、模拟键盘鼠标事件需要哪个类？（ Actions ）（ Keys ）

18、如何去定位属性动态变化的元素？（ xpath的contains函数 ）

19、TestNG允许指定依赖关系的两种实现方式：（在[@Test](https://github.com/Test)注释中使用属性dependsOnMethods），（在[@Test](https://github.com/Test)注释中使用属性dependsOnGroups）

20、TestNG可以通过两种不同的方式将参数直接传递给测试方法：（使用testng.xml）、（使用数据提供者）

21、在Web应用中经常会遇到frame/iframe表单嵌套页面的应用，WebDriver只能在一个页面上对元素识别与定位，对于frame/iframe表单内嵌页面上的元素无法直接定位。这时就需要通过(switchTo().frame())方法将当前定位的主体切换为frame/iframe表单的内嵌页面中。

如果完成了在当前表单上的操作，则可以通过(switchTo().defaultContent())方法跳出表单。

21、在页面操作过程中有时候点击某个链接会弹出新的窗口，这时就需要主机切换到新打开的窗口上进行操作。WebDriver提供了switchTo().window()方法可以实现在不同的窗口之间切换

22、ITestResult是TestNG提供的一个接口，结合@AfterMethod，可以监听@Test方法的执行状态等信息

23、TestNG中，如果要在测试方法执行成功、失败或者跳过时指定不同后续行为，可以利用ITestListener 监听器。也可以直接继承TestListenerAdapter类来实现。

24、TestNG的用例组织结构：Test Suite由一个或者多个Test组成，Test由一个或多个测试class组成，一个测试class由一个或多个测试方法组成

25、在testng.xml中（parallel）属性，可以并行地执行测试方法

26、继承LoadableComponent类可以在页面加载的时候判断是否加载了正确的页面，只需要重写load() 和isLoaded()两个方法。此方式有助于让页面对象的页面访问操作更加健壮。

27、在分布式测试模式中，只能有一台作为Hub计算机，Hub负责管理测试脚本，并负责发送脚本给其他的Node，Node节点会打开本地的浏览器完成测试任务并返回结果给Hub

28、python中在线安装Selenium的命令(pip install selenium)，查看Selenium是否安装（pip show selenium），离线安装，单击 .tar.gz结尾的文件，并且对文件进行解压，进入到解压目录中，通过python命令进行安装。进入到目录名中>（python setup.py install）

.whl 文件本质上面是一个 .zip 包格式进入到目录名中>（pip install selenium.whl）

29、Java+Selenium上传本地文件；只要定位上传按钮，通sendKeys()添加本地文件路径就可以了。

30、unittest.main()使用TestLoader类搜索所有包含在该模块以test命名开头的测试方法，并自动执行它们，执行方法的默认顺序是：根据ASCII码的顺序加载测试用例

31、HTMLTestRunner是Python的标准库unittest单元测试框架的一个扩展，用于生成HTML测试报告。

32、在python中，xlrd模块实现对excel文件内容读取，xlwt模块实现对excel文件的写入

33、HTMLTestRunner是Python的标准库unittest单元测试框架的一个扩展，用于生成HTML测试报告。

34、Python对SMTP支持有smtplib和email两个模块，email负责构造邮件，smtplib负责发送邮件

# 选择

## Selenium

1、下面哪个方法是验证复选框是不是被选中（ ）

A．isEnabled() B. isSelected()

C. isDisplayed() D. clear()

答案：B

2、下面哪个方法可以获获取当前页面的URL（ ）

A．getClass() B. getCurrentUrl()

C. getPageSource() D. getTitle()

答案：B

3、如何模拟浏览器的前进操作（ ）

A．driver.navigate().back(); B. driver.navigate().toString()

C. driver.navigate().refresh(); D. driver.navigate().forward();

答案：D

4、在显示等待中，判断元素是否存在的方法是（ ）

[A. elementToBeClickable(locator)](mailto:A.@Test(enabled=true))

B. textToBePresentInElement(locator, text)

C. presenceOfElementLocated(locator)

D. titleContains(title)

答案：C

5、在对话框的操作，下列代码实现的功能是（ ）

Alert alert =driver.switchTo().alert();

alert.accept();）

A. 取消对话框 B. 在对话框中输入文本

C. 确定对话框 D. 获得对话框的文本内容

答案：C

6、选出在Java中有效的注释声明（ ）

A．//这是注释 B. \*/这是注释\*/

C. /这是注释 D. \*\*/这是注释\*/

答案：A

7、下列实例化浏览器对象错误的方式是（ ）

A. WebDriver driver = new FirefoxDriver();

B. WebDriver driver = new InternetExplorerDriver();

C. WebDriver driver = new WebDriver();

D. WebDriver driver = new ChromeDriver();

答案：C

8、什么样的项目适合自动化测试（ ）

A. 软件维护周期长。

B. 任务测试不明确，频繁变动

C. 测试人员不具备较强的编程能力

D. 软件系统界面不稳定，变动多

答案是：A

9、Webdriver的协议是什么？（ ）

[A．Http](mailto:A.@Test(enabled=true)) B．The WebDriver Wire Protocol

C．The WebDriver Protocol D．Tcp/IP

答案是B

10、怎样去选择一个下拉框中的value＝apple 下拉列表框？

A．selectByIndex(1) B．selectByValue(“apple”)

C．selectByVisibleText(“apple”) D．select(“apple”)

11、一个登录按钮，除了用click方法点击以外还有其他方式吗？

A．hashCode() B．wait()

C．notify() D. submit()

答案是：D

12、在显示等待中，元素是否可用和被单击是（ ）

[A．titleContains(title)](mailto:A.@Test(enabled=true))

B．textToBePresentInElement(locator,text)

C．presenceOfElementLocated(locator)

D．elementToBeClickable(locator)

答案是：D

13、（ ）是嵌入到Firefox浏览器中的一个插件，实现简单的浏览器操作的录制与回放功能，也可以将录制的脚本导出成java、Python、Ruby、C#

A．Selenium-RC B．Selenium-IDE

C．Selenium-WebDriver D．Selenium-Grid

答案是：B

14、下列哪项属于单元测试工具是（ ）

A．TestNG B．UFT

C．Selenium D．HttpClient

答案是：A

15、WebDriver在Selenium3中是（ ）

A．常量 B．类

C．接口 D．抽象类

答案是：C

16、下列哪项不属于Page Object 设计模式的优点（）

A．减少代码的重复 B．测试数据和测试行为进行了完全的分离

C．提高测试用例的可维护性 D．提高测试用例的可读性

答案是：B

## TestNG

1、TestNG如何设定测试用例的优先级（ ）

A．@Test(priority=0) B．@Priority=0

[C．@Test(dependsOnMethods="car](mailto:C.%20%20%20@Ignore)") D．@Test(parameters=1)

2、TestNG如何对测试用例进行分组（ ）

[A](mailto:A%20.%20%20@Test(enabled=true))．@Test(name=0) B．@Test(enabled=false)

[C．@Ignore](mailto:C.%20%20%20@Ignore) D．@Test(groups="car")

3、一个类里面可能存在多个测试方法(被@Test注解的方法)，默认测试方法的执行顺序是（ ）

A．按照方法在类中的先后顺序 B．按照方法名的字典序(ascii)升序排序执行

C．随机执行的 D．按照方法名的字典序(ascii)降序排序执行

4、下面哪个选项（ ）的作用是，被注释的方法将在每一个测试用例调用前运行

[A](mailto:A%20.%20%20@Test(enabled=true))．@BeforeMethod B．@BeforeClass

C．@BeforeTest D．@BeforeSuite

5、在testng.xml中，preserve-order=“true”的作用是（ ）

A．运行所有的测试用例

B．会在不同的线程中运行测试方法

C．可以保证节点下面的方法是按顺序执行的

D．运行失败的测试用例

6、使用断言判断a是否等于b的方法是（ ）

[A. assertEquals(a,b)](mailto:A.@Test(enabled=true))  B. assertSame(a, b)

C. assertTrue(a) D. assertNotEquals(a, b)

7、使用TestNG断言，测试a是否不等于b方法是（ ）

A．assertEquals(a,b) B．assertSame(a, b)

C．assertTrue(a) D．assertNotEquals(a, b)

8、用TestNG跳过某个测试方法的实现（）

[A. @Test(enabled=true)](mailto:A%20.%20%20@Test(enabled=true)) B. @Test(enabled=false)

[C. @Ignore](mailto:C.%20%20%20@Ignore) D. @Test(parameters=1)

9、TestNG中，（ ）可以实现测试方法的参数化

[A.](mailto:A%20.%20%20@Test(enabled=true)) @Test(dataProvider = "aaa") B. @Parameters

C．@Parameter D. @DataProvider

# 简答

## 1、手工测试的局限性？

参考答案：

1. 通过手工测试无法做到覆盖所有代码路径。

2）简单的功能性测试用例在每一轮测试中都不能少，而且具有一定的机械性、重复性，工作量往往较大。

3）如果有大量（几千）的测试用例，需要在短时间内（1天）完成，手工测试几乎不可能做到。

## 2、自动化测试的优势？

参考答案：

1. 缩短软件开发测试周期，可以让产品更快投放市场。
2. 测试效率高，充分利用硬件资源。
3. 节省人力资源，降低测试成本。
4. 增强测试的稳定性和可靠性。
5. 提高软件测试的准确度和精确度，增加软件信任度。
6. 软件测试工具使测试工作相对比较容易，但能产生更高质量的测试结果。

7）软件测试实行自动化进程，绝不是因为厌烦了重复的测试工作，而是因为测试工作的需要，更准确地说是回归测试和系统测试的需要。

## 3、什么项目适合做自动化测试？

参考答案：

1）任务测试明确，需求不会频繁变动。

2）每日构建后的测试验证。  
3）比较频繁的回归测试。  
4）软件系统界面稳定，变动少。  
5）需要在多平台上运行的相同测试案例、组合遍历型的测试，大量的重复任务。  
6）软件维护周期长，资源丰富。  
7）项目进度压力不太大。  
8）被测软件系统开发比较规范，能够保证系统的可测试性。  
9）具备大量的自动化测试平台。  
10）脚本重复利用率高。

## 4、自动化测试框架的步骤是什么？

参考答案：

1）根据测试业务的手工测试用例，选出需要自动化的用例（例如：冒烟）

2）根据可自动化执行的测试用例，分析出测试框架需要模拟的手工操作和重复高的测试流程或逻辑

3）将重复高的测试流程在代码中实现，并进行封装

4）根据业务的类型和本身的技术能力选择数据驱动测试、关键字驱动测试框架、混合型框架还是行为测试框架

5）确定框架类型后，将框架中的常用的浏览器选择、测试数据处理、文件操作、数据库操作、页面元素的原始操作、日志和报告等功能进行方法的封装实现

6）对框架代码进行集成测试和系统测试，采用PO模式和TestNG框架编写测试脚本，使用框架进行自动化测试，验证框架的功能是否可以满足自动化测试的需求。

7）编写自动化测试框架的常用API，以供他人参阅

8）在测试组中内部进行培训和推广

9）不断收集测试过程中的框架使用问题和反馈意见，不断增加和优化自动化框架的功能，不断增强框架中复杂操作的封装效果，尽量降低测试脚本的编写复杂性

10）定期评估测试框架的使用效果，评估自动化测试的投入和产出比，再逐步推广自动化框架的应用范围

## 5、请对Martin Fowler大师在测试金字塔模型的基础上提出分层自动化测试的概念，请进行描述。

1）单元测试自动化：在面向过程编程中，单元也可以是整个模块，单常见的是单个函数或过程。

在面向对象编程中，单元通常是整个接口，例如类，但可以是单独的方法。需要规范的来做单元测试同样需要相应的单元测试框架，如java的Junit、testNG，C#的NUint，Python的unittest、pytest等，几乎所有的主流语言，都会有其对应的单元测试框架。

2）接口测试自动化：Web应用的接口测试大体分为两类：模块接口测试和Web接口测试。

模块接口测试，主要测试模块之间的调用与返回。它主要强调对一个类方法或函数的调用，并对返回结果的验证，所用到的测试工具与单元测试相同。

Web接口测试又可分为两类：服务器接口测试和外部接口测试。

3）UI层的自动化测试： UI层的功能进行测试。例如，不断重复的对一个表单提交，结果查询等功能进行测试，我们可以通过相应的自动化测试工具来模拟这些操作，从而解放重复的劳动。

## 6、请描述Selenium Webdriver的工作原理。

参考答案：

1）启动浏览器后，selenium-webdriver会将目标浏览器绑定到特定的端口，启动后的浏览器则作为webdriver的remote server，接受测试脚本的命令。

2）客户端(测试脚本)，借助ComandExecutor发送HTTP请求给sever端（通信协议：The WebDriver Wire Protocol，在HTTP request的body中，会以WebDriver Wire协议规定的JSON格式的字符串来告诉Selenium希望浏览器接下来做什么事情）。

3）Sever端需要依赖原生的浏览器组件（如geckodriver.exe），来转化浏览器native的调用来完成操作

## 7、现在需要你设计一个自动化框架，简述自动化框架的构成。

1）utils包：实现测试过程中调用的工具类方法，例如：文件操作、mapObject、页面对象的操作方法

2）appModules包主要用于实现复用的业务逻辑封装方法

3）pageObjects包：用于实现被测试的页面对象

4）testCases 包 ：具体的测试方法

5）dataprovider包：提供数据驱动的类，txt驱动，excel驱动，数据库的驱动

## 8、Webdriver的三种等待方式是哪些，并简述它们的作用。

参考答案

1）强制等待 Thread.sleep（3000）

2）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。

WebDriverWait wait =new WebDriverWait(driver, 10);

wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("")));

3）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。

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

## 9、Selenium 有几种定位方式，请详细列出？

参考答案：

一共有8种类型的定位方式，它们都位于By类中并且都是静态方法。它们分别是：By.id(), By.name(), By.tagName(), By.className(), By.linkText(), By.partialLinkText(), By.xpath, By.cssSelector().

## 10、close()和quit()方法的区别是什么？

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。

## 11、Selenium 包含哪些组件，简述各组件的作用?

参考答案：

Selenium RC ，Selenium Grid，Selenium IDE ，Selenium WebDriver

webdriver是按照server–client的经典设计模式设计的:

•server端就是remote server，可以是任意的浏览器：测试脚本启动浏览器后，该浏览器就是remote server，它的职责就是等待client发送请求并做出响应

•client端（测试代码）：测试代码中的一些行为，比如打开浏览器，转跳到特定的url等操作是以http请求的方式发送给被server端（被测浏览器）server接受请求，并执行相应操作，并在response中返回执行状态、返回值等信息。

Selenium IDE 是嵌入到Firefox浏览器中的一个插件，实现简单的浏览器操作的录制与回放功能，也可以将录制的脚本导出成java、Python、Ruby、C#。

Selenium Grid是一种自动化的测试辅助工具，Grid通过利用现有的计算机基础设施，能加快Web应用的功能测试。利用Grid可以很方便地实现在多台机器上和异构环境中运行测试用例

## 12、如何处理一个alert弹窗？

参考答案

处理alert弹窗，需要先跳转到alert上，然后在点击确定或者取消按钮，最后返回到原始的页面，例如：

String mainPage = driver.getWindowHandle();

Alert alt = driver.switchTo().alert(); //跳转到alert上面

alt.accept(); //点击确定

alt.dismiss(); // 点击取消

//回到原始的页面

driver.switchTo().window(mainPage);

## 13、什么是page object设计模式？

使用面向对象的设计模式，页面对象模式将测试代码和被测试页面的页面元素及其操作方法进行分离，以此降低页面元素变化对测试代码的影响。

每一个测试页面都会被单独定义一个类，类中会定位所有需要参与测试的页面元素对象，并且定义操作每一个页面元素对象的方法。

## 14、Selenium Grid作用是什么?

专门用于远程分布式测试或并发测试的组件，可以在不同机器上测试不同浏览器

## 15、怎么判断元素是否存在？

判断元素是否存在和是否出现不同， 判断是否存在意味着如果这个元素压根就不存在， 就会抛出NoSuchElementException

这样就可以使用try catch，如果catch到NoSuchElementException 就返回false

## 16、如何判断元素是否出现？

判断元素是否出现，存在两种情况，一种是该元素压根就没有，自然不会出现；另外一种是有这样的元素，但是是hidden状态

可以通过先判断是否存在，如果不存在返回false；如果存在再去判断是否displayed

## 17、怎样选择下拉菜单的元素

如果这个下拉菜单是通过select标签创建的，可以通过以下代码实现

WebElement selector = driver.findElement(By.id("Selector"));

Select select = new Select(selector);

选择select的option有以下三种方法

selectByIndex(int index) 通过index

selectByVisibleText(String text) 通过匹配到的可见字符

selectByValue(String value) 通过匹配到标签里的value

如果这个下拉菜单不是通过select标签创建的，我们可以通过xpath定位元素，然后进行点击

## 18、如何使用webdriver执行一个鼠标右键点击操作

鼠标的操作使用Actions类

Actions actions=**new** Actions(driver);

actions.moveToElement(webElement).perform();

actions.contextClick().perform();

补充：

doubleClick(WebElement target)双击

clickAndHold(WebElement target)长按

dragAndDrop(WebElement source, WebElement target)拖动

perform() 执行所有Actions中存储的行为

release() 释放鼠标

//按住左键向右移动600，向下移动200，释放左键

action.clickAndHold().moveByOffset(600, 200).release().perform();

## 19、如果在测试过程中，发现click();发现没有响应，该怎么办？

driver.findElement(By.xpath("//input[@type='submit']")).click();发现没有响应，该怎么办？

参考答案：

JavascriptExecutor jse=(JavascriptExecutor) driver;

jse.executeScript("arguments[0].click();", element);

## 20、如何模拟滚动条向下滑动？

参考答案：

JavascriptExecutor jse=(JavascriptExecutor) driver;

jse.executeScript("window.scrollBy(0,800);”);

## 21、java+Selenium实现截屏的操作

**public** **void** screenShot(String name) **throws** IOException {

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyyMMdd-HHmmss");

String nowDateTime = sdf.format(**new** Date());

File s\_file = ((TakesScreenshot) driver).getScreenshotAs(OutputType.***FILE***);

Files.*copy*(s\_file, **new** File("D:\\demo\\" + name + "-" + nowDateTime+ ".jpg"));

}

## 22、怎么验证一个复选框是不是被选中或者未被选中

参考答案：

使用isSelected()方法，例如

driver.findElement

(By.id("xpath of checkbox")).isSelected()，如果返回true说明被选中，如果返回false，说明未选中

## 23、如何使用Selenium执行一个拖拽的操作？

Actions类

Actions actions=new Actions(driver);

actions.dragAndDrop(source, target).perform();

## 24、findElements(By by); findElement(By by)的区别是什么？

两者都是查找元素的方法，findElement是查找一个页面元素，它只会返回一个页面元素类型WebElement。findElements会查找页面上所有符合匹配的元素，它返回的是一个元素集合List<WebElement>。

键盘事件

## 28、如何使用键盘事件，点击向下键和Tap键

Keys()类提供了键盘上几乎所有按键的方法。

Actions action = new Actions(driver);

//键盘向下键

action.sendKeys(Keys.DOWN).build().perform();

sendKeys(Keys.BACK\_SPACE) 删除键（BackSpace）

sendKeys(Keys.SPACE) 空格键(Space)

sendKeys(Keys.TAB) 制表键(Tab)

sendKeys(Keys.ESCAPE) 回退键（Esc）

sendKeys(Keys.ENTER) 回车键（Enter）

## 29、在一个页面上，点击一个链接之后不会在当前页面上打开此链接而是新开一个窗口打开此页面。

Selenium也提供了对应的API来处理。简述这个查找新窗口的过程。

参考答案：

1）首先需要了解下，每个窗口都是有个句柄的，可以理解为浏览器窗口的标识符。可以根据这些标识符来确定新开的窗口，具体思路如下：

首先获得当前句柄：

String current\_handles = driver. getWindowHandle();

2）出现新窗口之后，获取所有窗口句柄

Set<String> all\_handles = driver.getWindowHandles();

3）循环判断，把当前句柄从所有句柄中移除，剩下的就是想要的新窗口

Iterator<String> it = all\_handles.iterator();

while(it.hasNext()){

if(current\_handles == it.next()) continue;

//跳入新窗口

WebDriver window = dr.switchTo().window(it.next());

}

## 30、Selenium Grid的使用过程

1. 找到两台计算机A作为Hub（代码），B作为Node（jdk,selenium.jar,浏览器）

2. 在计算机A中打开cmd窗口，在D:/demo输入语句：

java –jar selenium.jar –role hub

使用java命令把jar包文件作为程序，并且将role参数传递给jar文件的参数，以此来启动管理中心

3. 在计算机B打开窗口，打开cmd窗口，在D:/demo下输入语句：java–jar selenium.jar -role node -port 6655 -hub

<http://10.7.90.240:4444/grid/register>

4、在计算机A中访问<http://localhost:4444/grid/console>，验证Node是否注册成功

5、在A 机器上编写测试分布式代码

## 31、HTMLTestRunner的使用步骤？

参考答案：

使用步骤：

1、导入HTMLTestRunner

2、通过open()方法以二进制写方式打开当前目录的result.html，如果没有，则自行创建该文件

3、调用HTMLTestRunner下的HTMLTestRunner类，stream指定测试报告文件，title定义测试报告的标题，description用于定义测试报告的副标题

4、通过HTMLTestRunner的run()方法来运行测试套件中所组装的测试用例

5、通过close()关闭测试报告文件

## 1、TestNG中注释有哪些？

@BeforeSuite，@AfterSuite，@BeforeTest， @AfterTest

@BeforeGroups，@AfterGroups，@BeforeClass，@AfterClass

@BeforeMethod，@AfterMethod ，@Test

## 2、TestNG中什么是断言（Assert）？

Assert用于验证结果是不是达到预期的结果，如果在执行测试的时候测试用例失败了，它会停止执行当前的测试用例并提供断言日志。

## 3、什么是数据驱动模式，TestNG如何从Excel读数据的参数化？

参考答案：

1）相同的测试脚本使用不同的测试数据来执行，测试数据和测试行为进行了完全的分离。这样的测试脚本设计模式称为数据驱动。

2）

引入apache poi的jar包，使用这个包的类和方法可以从Excel读取数据

使用testNG 的注解，下定义一个方法返回类型Object[][]的方法用来接收从excel读取的数据

定义一个需要参数化的@Test方法，指定数据来自上面第二步的DataProvider

## 4、如何使用TestNG+Ant创建无人值守的测试？

参考答案：

1）编写测试代码，运行通过

2）编写testng.xml

3）编写build.xml(执行testng.xml)

1. 编写bat文件（dos下的批处理文件）

5）使用Windows中任务创建定时任务

## 5、使用TestNG的优势是什么？

（1）TestNG可以基于group的方式去执行测试

（2）Tesng注释很好理解

（3）Testng允许并行测试

（4）报告展示多样化，可自行开发测试报告插件以及测试监听器等

（5）执行测试的顺序可通过testng修改

1. 错误的用例可以被再次被执行

## 6、什么是持续集成？

持续集成源于极限编程(XP)，是一种软件实践，软件开发过程中集成步骤是一个漫长并且无法预测的过程。集成过程中可能会爆发大量的问题，因此集成过程需要尽可能小而多，实际上持续集成讲的是不断的去做软件的集成工作。持续集成，最简单的形式是包括一个监控版本控制（Github）变化的工具。当变化被发觉时，这个工具可以自动的编译并测试应用。

## 7、SoftAssert的特点，并对软断言进行举例说明

1） 如果一个断言失败，会继续执行这个断言下的其他语句或者断言  
3） 调用assertAll()在该用例的最后一个断言后面  
4） 软断言的类，叫SoftAssert.java，这个类是需要创建实例对象，才能调用相关实例方法进行软断言

举例

@Test

public void test2() {

System.out.println("Test start");

SoftAssert assertion = new SoftAssert();

assertion.assertEquals(12, 13,"两者不相等");

System.out.println("Test complete");

System.out.println(3+8);

assertion.assertAll();

}

## 8、使用TestNG编写测试用例的步骤。

参考答案：

1. 编写测试代码逻辑
2. 插入TestNG注解标签
3. 配置TestNG.xml文件，设定测试类、测试方法、测试分组的执行信息。
4. 执行TestNG查看测试报告

# 判断

1、由于自动化测试的诸多好处，自动化终将完全取代手工测试（错）

1. Selenium是用于白盒测试的工具（ 错 ）
2. Xpath是一门在 XML 文档中查找信息的语言（ 对 ）
3. Selenium是用于多浏览器支持（ 对 ）
4. WebElement是个类，不是接口（ 错 ）
5. Selenium可以操作页面的滚动条（ 对 ）
6. Selenium不支持JS脚本的执行（ 错）
7. Selenium本身就可以支持无人值守的测试（错 ）
8. Selenium支持HTML5新元素（ 对 ）
9. Selenium中hidden或者是display ＝ none的元素无法定位（ 错 ）

10、Selenium可以修改页面元素的属性，例如背景颜色（ 对 ）

11、XPath是一种在XML文档中定位元素的语言。因为HTML可以看作XML的一种实现，所以Selenium用户可以使用这种强大的语言在Web应用中定位元素（对）

12、在 @Listeners 中添加监听器跟在 testng.xml 添加监听器的不同之处在于，它不能添加 IAnnotationTransformer 和 IAnnotationTransformer2 监听器。原因是因为这两种监听器必须在更早的阶段添加到 TestNG 中才能实施修改注释的操作，所以它们只能在 testng.xml 添加。（对）

13、TestNG执行测试方法之前，会重新实例化测试类，即执行构造方法。（ 对 ）

14、如果继承父类，会先执行父类的构造方法。（ 对 ）

15、一个类里面可能存在多个测试方法(被**@Test**注解的方法)，默认测试方法的执行顺序是按照方法名的字典序(ascii)升序排序执行的。

15、硬断言如果一个断言失败，会继续执行这个断言下的其他语句或者断言。（错）

16、在分布式测试模式中，只能有一台作为Hub计算机，一台Node。（错）

17、Hub分发的测试用例会在多个Node中执行。（ 对 ）

# 编程题

## javamail+监听器的使用

使用TestNG+JavaMail,编写监听器类TestListener1 ,实现把测试结果以文本的方式发送至test333@126.com。发件箱是test111@126.com。授权码123456qwe。

**package** ch14.javamail;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

**public** **class** Demo1 {

@Test

**public** **void** test1() {

Assert.*assertEquals*(1, 2);

}

@Test

**public** **void** test3() {

Assert.*assertEquals*(1, 4);

}

@Test

**public** **void** test2() {

Assert.*assertEquals*(1, 1);

}

}

参考答案：

**public** **class** TestListenerWeb **extends** TestListenerAdapter {

@Override

**public** **void** onFinish(ITestContext testContext) {

StringBuilder sb = **new** StringBuilder();

ITestNGMethod method[] = **this**.getAllTestMethods();

sb.append("此次一共运行测试用例"+method.length+"\n");

sb.append("失败的测试用例如下："+"\n");

List<ITestResult> failedList = **this**.getFailedTests();

**for**(ITestResult result:failedList) {

String f\_method=result.getInstanceName()+"."+result.getName();

sb.append(f\_method+"\n");

}

MailUtil1.*sendMail*(sb.toString());

}

}

**public** **class** MailUtil1 {

**static** String *from*="test2014123@126.com";

**static** String *auth\_code*="123456abcd";//授权码，不是密码

**static** String *to*="626231936@qq.com";

**public** **static** **void** sendMail(String content) {

Properties prop=**new** Properties();

prop.put("mail.transport.protocol", "smtp");

prop.put("mail.smtp.host", "smtp.126.com");

prop.put("mail.smtp.auth", **true**);

Session session=Session.*getInstance*(prop, **new** Authenticator() {

@Override

**protected** PasswordAuthentication getPasswordAuthentication() {

**return** **new** PasswordAuthentication(*from*,*auth\_code* );

}

});

Message message=**new** MimeMessage(session);

**try** {

message.setFrom(**new** InternetAddress(*from*));

message.addRecipient(Message.RecipientType.***TO***, **new** InternetAddress(*to*));

message.setSubject("Hello，测试");

message.setText(content);

Transport.*send*(message);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

System.***out***.println(e.toString());

}

}

}

## txt参数化的实现

**public** **class** FileDataProvider {

**public** Object[][] getTestDataByTxt(String fileName) {

File file = **new** File(fileName);

Object[][] data = **null**;

FileInputStream fis = **null**;

InputStreamReader isreader = **null**;

BufferedReader breader = **null**;

String record;

**try** {

fis = **new** FileInputStream(file);

isreader = **new** InputStreamReader(fis, "UTF-8");

// 设定utf-8,使用带缓冲区的字符输入流BufferedReader读取文件内容

breader = **new** BufferedReader(isreader);

**int** cols = 0;

// 忽略第一行

record = breader.readLine();

**if** (**null** == record) {

**return** **null**;

} **else** {

cols = record.split("\t").length;

}

List<String> datalist = **new** ArrayList<String>();

// 遍历读取文件除第一行外的所有内容，储存在名为datalist的ArrayList中

**while** ((record = breader.readLine()) != **null**) {

datalist.add(record);

}

data = **new** Object[datalist.size()][cols];

String[] strs;

**for** (**int** i = 0; i < datalist.size(); i++) {

strs = datalist.get(i).split("\t");

**for** (**int** j = 0; j < cols; j++) {

data[i][j] = strs[j];

}

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** data;

}

}

## excel参数化的实现

**public** **class** ExcelDataProvider {

@DataProvider(name = "aaa")

**public** Object[][] getData() **throws** IOException {

**return** getTestDataByExcel("data/user.xlsx","Sheet1");

}

@Test(dataProvider = "aaa")

**public** **void** test1222(String a, String b) {

System.***out***.println(a);

}

**public** Object[][] getTestDataByExcel(String fileName, String sheetName)

**throws** IOException {

File file = **new** File(fileName);

FileInputStream inputstream = **new** FileInputStream(file);

Workbook wbook = **null**;

String fileExtensionName = fileName.substring(fileName.indexOf("."));

**if** (fileExtensionName.equals(".xlsx")) {

wbook = **new** XSSFWorkbook(inputstream);

} **else** **if** (fileExtensionName.equals(".xls")) {

wbook = **new** HSSFWorkbook(inputstream);

}

Sheet sheet = wbook.getSheet(sheetName);

// 通过sheetName生成Sheet对象

**int** rowCount = sheet.getLastRowNum() - sheet.getFirstRowNum();

// 获取当前sheet行数，行号和列号都是从０开始

List<Object[]> records = **new** ArrayList<Object[]>();

// 使用双循环获取excel文件的所有数据（第一行除外）

**for** (**int** i = 1; i < rowCount + 1; i++) {

Row row = sheet.getRow(i);

String fields[] = **new** String[row.getLastCellNum()];

**for** (**int** j = 0; j < row.getLastCellNum(); j++) {

// 获取单元格数据

fields[j] = row.getCell(j).getStringCellValue();

}

records.add(fields);

}

Object[][] results = **new** Object[records.size()][];

**for** (**int** i = 0; i < records.size(); i++) {

results[i] = records.get(i);

}

**return** results;

}

}

## Selenium的使用

表格的处理

1）从页面表格中找到第二行第二列的输入框，并输入“hello”（10分）

提示：驱动路径为D:\\demo\\geckodriver.exe，要求使用firefox浏览器，安装路径为默认路径，测试路径：<http://localhost:8032/test/table.html>

注意：代码不唯一，可以使用不同的方式识别对象，能够正确识别对象均可得分

参考答案：

@Test

public void test1(){

//遍历所有的表格内的文本

System.setProperty("webdriver.gecko.driver", "D:\\demo\\geckodriver.exe");

wd = new FirefoxDriver();

wd.get("http://localhost:8032/test/table.html");

WebElement t1 = wd.findElement(By.id("table1"));

List<WebElement> rows=t1.findElements(By.tagName("tr"));

List<WebElement> cols1= rows.get(1).findElements(By.tagName("td"));

WebElement cellInput=cols.get(1).findElement(By.tagName("input"));

element1.sendKeys("hello");

wd.quit();.

}

2）课上练习，Mymovie的例子

## Selenium Grid的代码实现

WebDriver driver = **null**;

String nodeUrl="http://192.168.154.121:6655/wd/hub";

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

@Test

**public** **void** testFF() **throws** MalformedURLException, InterruptedException{

DesiredCapabilities capability = DesiredCapabilities.*firefox*();

**this**.driver = **new** RemoteWebDriver(**new** URL(nodeUrl),capability);

**this**.driver.get(baseUrl);

Thread.*sleep*(5000);

**this**.driver.quit();

}

## TestNG注解的使用。程序分析题，写出下列代码的运行结果

提示：<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name=*"Suite1"* verbose=*"1"*>

**package** edu.demo;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeSuite;

**public** **class** BaseTest {

**public** BaseTest() {

System.***out***.println("执行父类的构造方法");

}

@BeforeSuite

**public** **void** bs() {

System.***out***.println("BaseTest-BeforeSuite");

}

@BeforeClass

**public** **void** bc() {

System.***out***.println("BaseTest-BeforeClass");

}

}

**package** edu.demo;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.Test;

**public** **class** Demo2 **extends** BaseTest {

@BeforeMethod

**public** **void** bm() {

System.***out***.println("Demo2-BeforeMethod");

}

@Test

**public** **void** test2() {

System.***out***.println("test2");

}

@Test

**public** **void** atest2() {

System.***out***.println("atest2");

}

}

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name=*"Suite1"* verbose=*"1"*>

<test name=*"test1"*>

<classes>

<class name=*"edu.demo.Demo2"* />

</classes>

</test>

</suite>

## 属性文件的使用

**public** **class** ReadProperties {

**public** **static** **final** String ***filePath***="conf/config.properties";

**public** **static** String getPropertyValue(String key) **throws** IOException {

Properties prop = **new** Properties();

FileInputStream fis = **new** FileInputStream(***filePath***);

prop.load(fis);

fis.close();

**return** prop.getProperty(key);

}

}

## 页面对象模式的实现

**public** **class** LoginPage {

// 1、通过@FindBy的方式找出所有涉及到操作的页面元素

// 2、在构造方法中实例化WebDriver对象,并初始化所有的页面元素

// 3、实现所有的操作方法

**private** WebDriver driver =**null**;

@FindBy(name="username")

**public** WebElement username;

@FindBy(name="password")

**public** WebElement password;

@FindBy(linkText="退出")

**public** WebElement quit;

@FindBy(xpath="//input[@type='submit']")

**public** WebElement submitBtn;

// public Select Fruit;

**public** LoginPage2(WebDriver driver) {

PageFactory.*initElements*(driver, **this**);

**this**.driver=driver;

}

**public** **void** openUrl() {

**this**.driver.get("http://localhost:8032/mymovie/admin.php/");

}

**public** WebDriver getDriver() {

**return** driver;

}

**public** **void** login(String u\_name,String u\_password) **throws** InterruptedException {

**this**.username.sendKeys(u\_name);

**this**.password.sendKeys(u\_password);

**this**.submitBtn.click();

Thread.*sleep*(3000);

}

}

**public** **class** BaseTest {

**public** WebDriver wd = **null**;

**public** JavascriptExecutor jse = **null**;

**public** Actions action = **null**;

@BeforeClass

**public** **void** setUpBrowser() **throws** InterruptedException {

System.*setProperty*("webdriver.gecko.driver", "D:\\demo\\geckodriver.exe");

System.*setProperty*("webdriver.firefox.bin", "D:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe");

wd = **new** FirefoxDriver();

jse = (JavascriptExecutor) wd;

wd.manage().timeouts().implicitlyWait(5, TimeUnit.***SECONDS***);

action = **new** Actions(wd);

}

@AfterClass

**public** **void** quitBrower() {

**if** (wd != **null**) {

wd.quit();

}

}

**public** **class** LoginPage2Test **extends** BaseTest{

LoginPage2 lp;

@BeforeClass

**public** **void** init(){

lp =**new** LoginPage2(wd);

}

@BeforeMethod

**public** **void** Open() {

lp.openUrl();

}

@Test

**public** **void** testLoginSuccess() **throws** InterruptedException {

lp.login("admin", "admin");

*assertTrue*(lp.getDriver().getPageSource().contains("退出"));

}