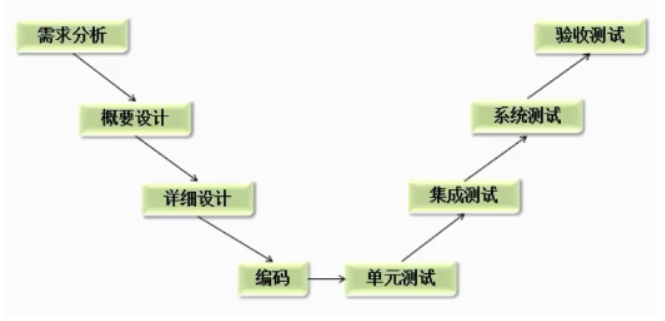
## 常用的测试建模方法：

V模型、W模型、X模型、H模型

V模型：（适用于中小企业）



优点：

包含了底层测试（单元测试）和高层测试（系统测试）；清楚的标识了开发和测试的各个阶段；自上而下逐步求精，每个阶段分工明确，便于整体项目的把控。

缺点：

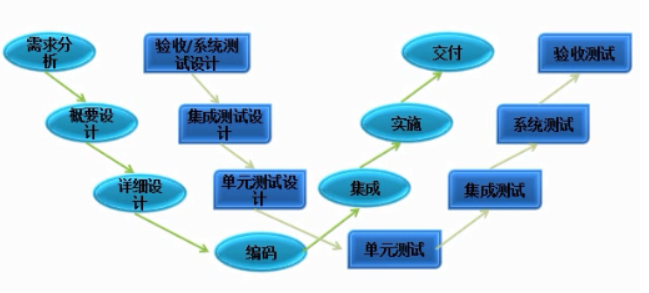
自上而下的顺序导致了，测试工作在编码之后，就导致错误不能及时的进行修改；实际工作中，需求经常变化，导致v模型步骤，反复执行，返工量很大，灵活度较低。

改良：每个步骤可以进行小的迭代工作。

适用于一些传统信息系统应用的开发

W模型：（适用于中大型企业：对人员要求高）

开发一个V，测试一个V



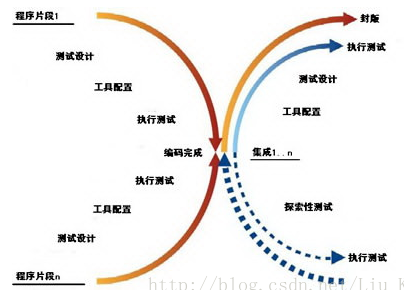
优点：

开发伴随着整个开发周期，需求和设计同样要测试；更早的介入测试，可以发现初期的缺陷，修复成本低；分阶段工作，方便项目整体管理。

缺点：

开发和测试依然是线性的关系，需求的变更和调整，依然不方便；如果没有文档，根本无法执行w模型；对于项目组成员的技术要求更高！

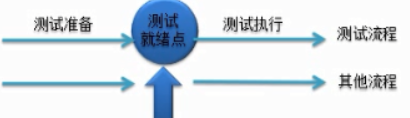
X模型：



X模型也是对V模型的改进,X模型提出针对单独的程序片段进行相互分离的编码和测试,此后通过频繁的交接,通过集成最终合成为可执行的程序。

X模型的左边描述的是针对单独程序片段所进行的相互分离的编码和测试，此后将进行频繁的交接，通过集成最终成为可执行的程序，然后再对这些可执 行程序进行测试。己通过集成测试的成品可以进行封装并提交给用户，也可以作为更大规模和范围内集成的一部分。多根并行的曲线表示变更可以在各个部分发生。

H模型：（人员要求非常高，很少人用，仅了解）



优点：

>开发的H模型揭示了软件测试除测试执行外，还有很多工作；

　　>软件测试完全独立，贯穿整个生命周期，且与其他流程并发进行；

　　>软件测试活动可以尽早准备、尽早执行，具有很强的灵活性；

>软件测试可以根据被测物的不同而分层次、分阶段、分次序的执行，同时也是可以被迭代的

缺点：

>管理型要求高：由于模型很灵活，必须要定义清晰的规则和管理制度，否则测试过程将非常难以管理和控制；

　　>技能要求高：H模型要求能够很好的定义每个迭代的规模，不能太大也不能太小；

　　>测试就绪点分析困难：测试很多时候，你并不知道测试准备到什么时候是合适的，就绪点在哪里，就绪点的标准是什么，这就对后续的测试执行的启动带来很大困难；

>对于整个项目组的人员要求非常高：在很好的规范制度下，大家都能高效的工作，否则容易混乱。例如：你分了一个小的迭代，但是因为人员技能不足，使得无法有效完成，那么整个项目就会受到很大的干扰。

等价类设计思路：

1.根据输入条件，确定等价类，包括有效等价类和无效等价类，建立等价类列表

2.为每个等价类规定一个唯一的编号

3.设计一个测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类被覆盖完为止

4.设计一个新的测试用例，使其覆盖且只覆盖一个尚未覆盖的无效等价类。重复这一步，直到测试用例覆盖了所有的无效等价类。

等价类的划分：

1. 按区间划分
2. 按数值划分
3. 按数值集合划分
4. 按限制条件或规则划分
5. 细分等价类

弱一般等价类测试：基于单缺陷假设，对一个测试用例中的每个等价类的一个变量实现

强一般等价类测试：基于多缺陷假设，覆盖每个等价类进行测试。

弱健壮等价类测试：基于单缺陷假设的健壮性测试，对于无效输入等价类也将拥有一个测试用例.

强健壮等价类测试：基于多缺陷假设的健壮性测试。

举例来说，学生成绩，范围是0～100：

弱一般等价类：0~100中任意一个数为测试数据，只设计1个用例即可

强一般等价类：分为0、 1~99、 100三个等价类，需要设计3个用例

弱健壮等价类：考虑到60分在实际情况中为特殊数字，划分0、 1~99（除60外）、 100、 60四个等价类

强健壮等价类：考虑各种非法输入，如负数、其他字符等等

等价类划分是为了测试的完备性和无冗余性。

健壮性有两层含义：容错能力和恢复能力

容错性测试通常构造一些不合理的输入来引诱软件错误，如：输入错误的数据类型；输入定义域之外的数值

恢复能力测试如网络切断后再连接时，测试系统能否重新运行；有无重要的数据丢失；是否毁坏了其他相关的软件硬件。

---------------------------------------------------------------------

初识Selenium2.0

Selenium2.0由Selenium1.0+WebDrive组成；Selenium1.0由Selenium Grid+Selenium IDE+Selenium RC组成；

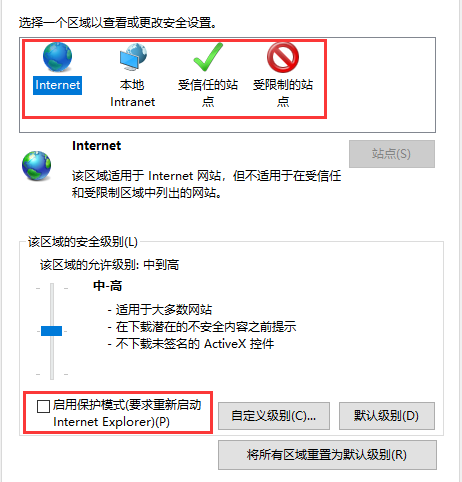
Selenium3.0相比较于Selenium2.0移除了Selenium RC；Firefox和Safari推出了自己的driver；Selenium3支持IE9+,Selenium2.0支持7-11；全面拥抱java8

Selenium3.0可以说完全等同于WebDriver，因为WebDrive里面也包含了Selenium Grid和Selenium IDE。

测试类中，测试方法以方法名的ASCLL码顺序执行

firefox56以下不需要安装webdriver驱动，firefox57（firefox47及以上版本）需要安装geckodriver-v0.19.1-win32/64；如果安装webdriver驱动，则地址填写驱动所在地址，不安装驱动，则地址填写运行文件地址

IEDriverServer的版本号和Selenium的版本号一定要一致；如果报错，需要改变Internet选项安全栏中四个保护模式状态一致；



chromedriver驱动下载地址：

<http://chromedriver.storage.googleapis.com/index.html>

geckodriver火狐驱动下载地址：

<https://github.com/mozilla/geckodriver/releases>

iedriverserver驱动下载地址：

<http://selenium-release.storage.googleapis.com/index.html>

MicrosoftWebDriver驱动下载地址：

<https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

driver.close(); 关闭当前窗口

driver.quit(); 完全退出浏览器

打开某网页：

driver.get(“http://www.baidu.com”); // http头不能少

driver.navigate().to(“http://www.baidu.com”);和get()方法等价，get()方法会等圈子转完才会执行下一步

driver.navigate().back(); // 浏览器后退

driver.navigate().forward(); // 浏览器前进

driver.navigate().refresh(); // 浏览器刷新

driver.manage().window().maximize(); // 浏览器最大化

Dimension dimension = new Dimension(900,800);

driver.manage().window().setSize(dimension); // 设置浏览器大小

driver.getCurrentUrl(); // 获取当前页面URL

driver.getTitle(); // 获取当前页面title

---------------------------------------------------------------------

自动化测试的根本：

准确的操作测试对象（元素）:查找元素、操作元素

查找元素：

findElement(); 返回一个WebElement元素 若返回的是多个元素，则只操作第一个或报错

findElements(); 返回一个List，多个WebElement元素

八种定位方式：

By.id(id); 通过ID属性查找

By.name(name); 通过name属性查找

By.className(className); 通过classname属性查找

Classname中有空格时要用.代替，返回多个元素时，使用findElement()会报错

By.linkText(链接文本); 通过链接文本查找 -- <a></a>标签里面的文本才可以使用

By.partialLinkText(部分链接文本); 通过部分链接文本查找

By.cssSelector(Css路径); 通过CSS路径查找 – 和xpath差不多

By.tagName(tagName); 通过tagname查找

By.xpath(XPath路径); 通过XPath路径查找 – xpath也可以定位多个元素 xpath下标从 1 开始

优先使用ID属性查询：ID查询最有效、最快；并且ID属性一般都是唯一的；但是ID有时是随机数

没有ID，考虑name；name不行，最后再用xpath

特殊需求再考虑其他定位方式

如果定位方式错误，元素找不到将抛出NoSuchElementException错误。

XPath：

在 XPath 中，有七种类型的节点：元素节点、属性节点、文本节点、命名空间节点、处理指令节点、注释节点、文档（根）节点。

XML 文档是被作为节点树来对待的。树的根被称为文档节点或者根节点。

基本值（或称原子值，Atomic value）：无父或无子的节点

项目（Item）：基本值或者节点

节点关系：

父（Parent）：每个元素以及属性都有一个父。

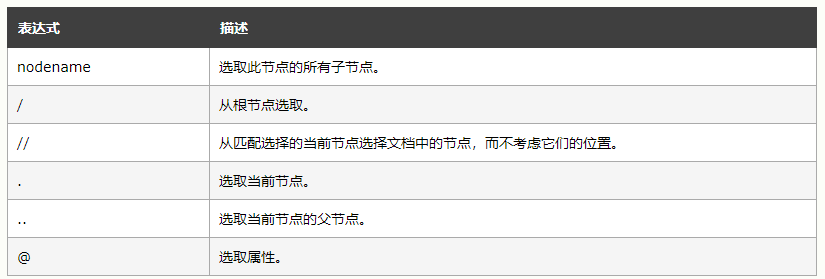
子（Children）：元素节点可有零个、一个或多个子

同胞（Sibling）：拥有相同的父的节点

先辈（Ancestor）：某节点的父、父的父，等等

后代（Descendant）：某节点的子，子的子，等等

选取节点：



实例：



谓语（Predicates）

谓语用来查找某个特定的节点或者包含某个指定的值的节点。谓语被嵌在方括号中。



选取若干路径：

通过在路径表达式中使用”|”运算符，选取若干个路径

步（step）：

轴（axis）：定义所选节点与当前节点之间的树关系

节点测试（node-test）：识别某个轴内部的节点

零个或者更多谓语（predicate）：更深入地提炼所选的节点集

步的语法：轴名称：：节点测试[谓语]

绝对位置路径起始于正斜杠（/）

运算符：|、+、-、\*、div、=、!=、<、>、<=、>=、or、and、mod

---------------------------------------------------------------------

webElement.click(); 点击方法

webElement.sendKeys(); input文本框输入文本

webElement.clear(); input清空文本框

webElement.getText(); 获取标签中间的值

webElement.getTagName() 获取标签的名字

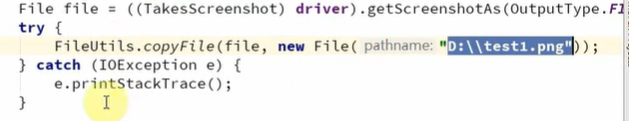
webElement.getAttribute() 获取属性值

webElement.isDisplayed() 判断元素是否展示

webElement.isSelected() 判断元素是否被选中

webElement.Enabled() 判断元素是否被激活

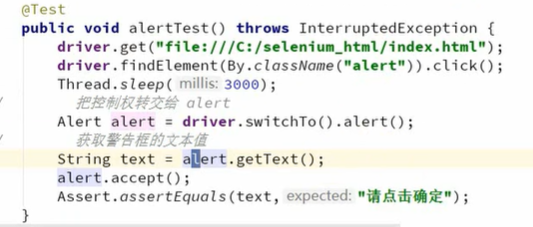
截图操作：只截取当前窗口

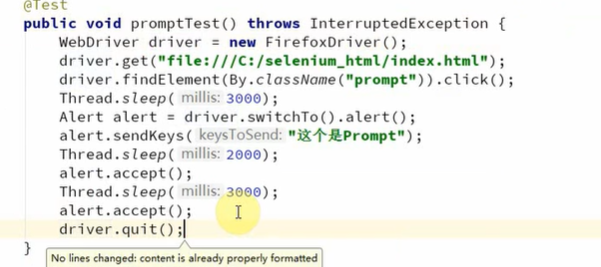


Alter弹窗处理：

把控制权交给Alert，此时alert相当于一个webElement：

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







Frame处理：

webDriver.switchTo().frame(name or id or webElement or int); 控制权转交给iframe

控制权转交给原来界面：

webDriver.switchTo().defaultContent();

多级iframe要一级一级传递，返回时也要一级一级返回；



下拉框的处理：

1. select.selectByIndex(); 根据索引来选取，从0开始
2. select.selectByValue(); 根据属性value的属性值来选取
3. select.selectByVisibleText(); 根据标签之间的Text值，页面显示的值选取

select类还有其他对下拉框操作的方法



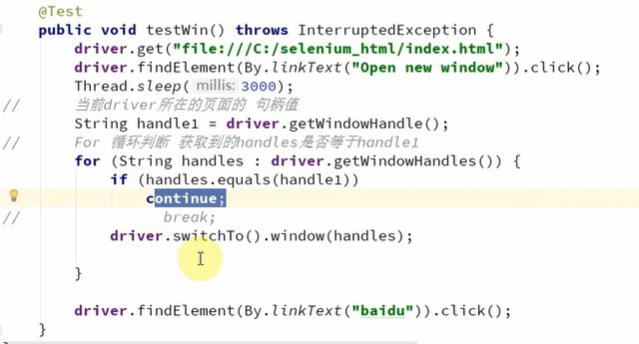
多窗口的处理：

String handle1 = webDriver.getWindowHandle(); 获取当前window的句柄值，即webDriver控制权所在的窗口的handle

webDriver.getWindowHandles(); 获取所有window的句柄值

webDriver.switchTo().window(handle); 通过句柄值转交控制权

句柄值每次随机生成，每当窗口打开，则句柄值固定，不会改变



元素等待的处理：

1. 线程等待：Thread.sleep(xxx);
2. 全局等待：（隐式等待）

webDriver.manage().timeouts().implicitlyWait(30,TimeUnit.SECONDS)

1. 显示等待： (推荐使用)

WebDriverWait wait = new WebDriverWait(webDriver,10);

wait.until(ExpectedConditions.presenceOfElementLocated(By.id));

Actions类(鼠标操作-selenium包)：

实例化Actions类：Actions actions = new Actions(driver);

右键鼠标：actions.contextClick(webElement).perform();

双击：actions.doubleClick(webElement).perform();

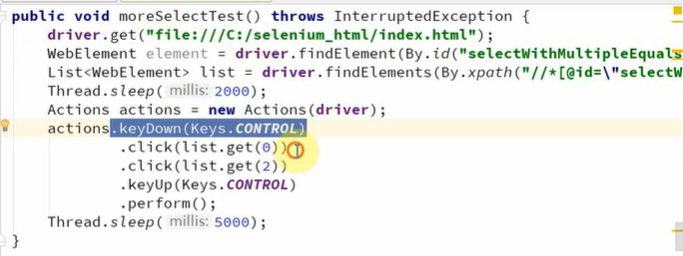
鼠标移到某个元素上：actions.moveToElement(webElement).perform();

元素拖动：actions.dragAndDropBy(webElement,500,500).perform();

把元素拖到另一个元素上：

actions.clickAndHold(el1).moveToElement(el2).release(el1).perform();

多选下拉框：



Java-Robot类：（鼠标-键盘组合键-java.awt）

按住某个按键：keyPress();

松开某个按键：keyRelease();

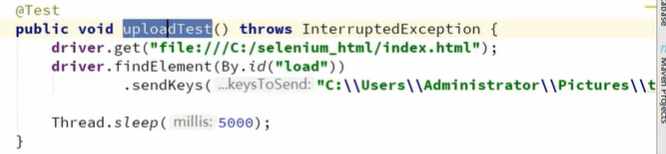
实例化Robot：Robot robot = new Robot();

按住ctrl键：robot.keyPress(KeyEven.VK\_CONTROL);

按住s键：root.keyPress(KeyEven.VK\_S);

上传文件：

获取input输入框元素，然后往里面加入所要上传文件的路径



---------------------------------------------------------------------

使用Js操作元素

把driver转成JavascriptExecutor类型：

JavascriptExecutor js = (JavascriptExecutor)webDriver;

执行js：js.executeScript(js语句);

PhantomJS（已经暂停开发，且新版selenium已不支持）

PhantomJS是一个可编程的无头浏览器.

无头浏览器：一个完整的浏览器内核,包括js解析引擎,渲染引擎,请求处理等,但是不包括显示和用户交互页面的浏览器。

无头浏览器使用场景：

1.页面自动化测试：希望自动的登陆网站并做一些操作然后检查结果是否正常。

2.网页监控：希望定期打开页面，检查网站是否能正常加载，加载结果是否符合预期。加载速度如何等。

3.网络爬虫：获取页面中使用js来下载和渲染信息，或者是获取链接处使用js来跳转后的真实地址。

使用chrome的无头浏览器：

创建无Chrome无头参数

ChromeOptions chromeOptions=new ChromeOptions();

chromeOptions.addArguments("-headless");

webDriver = new ChromeDriver(chromeOptions);

Selenium IDE:

Selenium IDE不需要任何编程逻辑来编写其测试脚本，而只需记录与浏览器的交互以创建测试用例。 之后，可以使用播放选项重新运行测试用例。

Selenium IDE仅作为Mozilla Firefox和Chrome插件提供，它无法在Firefox和Chrome以外的浏览器上记录测试用例。记录的测试脚本也可以导出到C#，Java，Ruby或Python等编程语言。

Selenium grid:

selenium grid是selenium家族中的三大组件的一员。selenium grid有两个版本，grid1和grid2,。目前grid1已经基本被废弃了。grid2的出版要晚于selenium2，所以grid2支持selenium2的所有的功能。

作用：

1.使webdriver写的脚本可以在不同的系统里，不同的浏览器下去运行。

2.可以批量执行用例，减少测试执行时间。

内容：

1.selenium grid不是用来写脚本代码的，只负责运行。所以我们的脚本还是需要利用webdriver，在我们的开发环境中去写好。

2.selenium grid给我们提供了两个东西。一个叫hub，一个叫node。

3.hub被称为总控节点。他是你加载所有的测试机器，一个grid里面只有一个hub。打个比方：hub就像我们的交换机，他负责接通网线，把网络分发给后面的各种设备，不管我们是用手机上网，还是电脑上网，都是连着我们的交换机。grid中的hub，总管着的是脚本代码,不能运行代码。

4.node称之为节点。他就好比我们的连接网络的各个终端设备，比如电脑，比如手机，iPad之类等等。所以grid中，可以有很多个node，node用于接收代码，且在不同的浏览器中运行代码。

启动hub

java -jar selenium-server-standalone-3.141.59.jar -role hub

-port：xxx 指定hub监听的端口

修改Hub配置：

1. 直接通过命令修改：

java -jar selenium-server-standalone-3.141.59.jar -role hub -port 4445

1. 通过Json文件修改：

java -jar selenium-server-standalone-3.141.59.jar -role hub -hubConfig hub.json

json格式：

{

"host": null,

"port": 4444,

"newSessionWaitTimeout": -1,

"servlets" : [],

"prioritizer": null,

"capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",

"throwOnCapabilityNotPresent": true,

"nodePolling": 5000,

"cleanUpCycle": 5000,

"timeout": 300000,

"browserTimeout": 0,

"maxSession": 5

}

Node节点环境要求：

1. node节点必须要有java环境
2. node节点跟hub节点机器间可以互相ping通
3. node节点负责执行Selenium脚本，所以必须要有Selenium环境（脚本语言对应的环境如java，各个浏览器及其对应的driver）

启动node，并注册到hub：

java -jar selenium-server-standalone-3.141.59.jar -role node -hub http://169.254.161.199:4444/grid/register/