# 一、测试用例规范

1. **测试用例逻辑：**

尽量只做功能中正向逻辑的验证，不要考虑太多逆向逻辑的验证，逆向逻辑的情况很多（例如手机号输错有很多种情况），验证一方面比较复杂，需要编写大量的脚本，另一方面自动化脚本本身比较脆弱，很多非正常的逻辑的验证能力不强。（我们尽量遵循用户正常使用原则编写脚本即可）

1. **修改功能：**

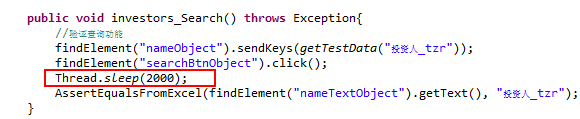
如果对数据进行了修改，需要对数据进行还原

1. **测试数据：**

添加新数据时，新数据命名需加上“自动化”字眼

1. **等待时间：**

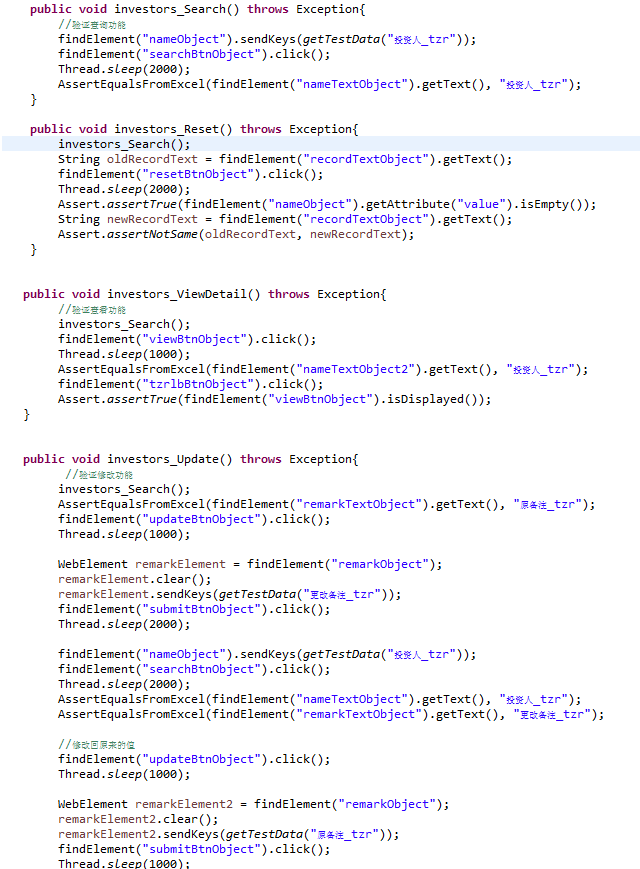
一般是在点击查询、查看、修改等按钮后加入等待时间，如果是强制等待，时间最好是不超过2秒



1. **尽量避免代码重复以提高效率**

**PageAction：**

**定义好查询方法，其他功能方法去调用**



修改：

调用查询方法，更改数据后，要对数据进行还原

查看详情：调用查询方法

定义查询方法

重置：调用查询方法

**TestCase：**

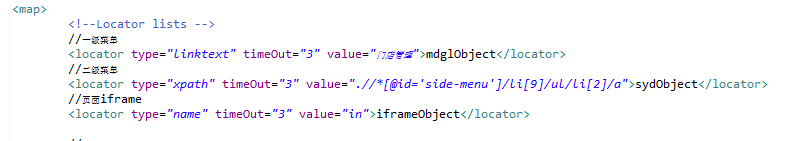
**无须重复写上查询功能，因为其他功能有调用**



**6、Object元素：**

6.1 在元素备注中必须用中文注明元素名称

6.2 元素命名按照开发代码命名，对于开发没有命名的必须使用有意义的英文命名



1. **元素定位的准确性（规范前端开发人员对元素添加id/name属性）：firebug 插件**

7.1 最常用id 、name定位；其次link text、partial link text、xpath；再考虑class name、tag name、css selector

7.2 xpath尽量使用相对地址定位，一般不推荐使用绝对路径（这样能减少元素定位受页面变化的影响）

# 二、Java命名规范

**类的命名 （单词首字母大写）**

根据约定，Java类名通常以大写字母开头，如果类名称由多个单词组成，则每个单词的首字母均应为大 写例如TestPage；如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如：XMLExample,还有一点命名技巧就是由于类是设计用来 代表对象的，所以在命名类时应尽量选择名词。 　　   
例如： Graphics

**方法的命名 （首字母小写，字母开头大写）**

方法的名字的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头。   
例如：drawImage

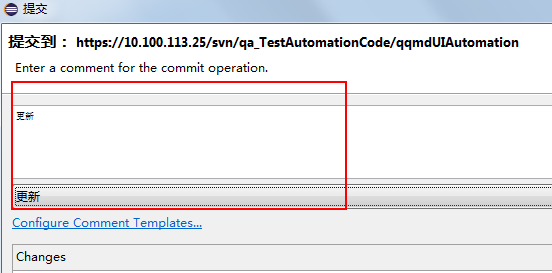
**常量的命名 （全部大写 ，常加下划线）**常量的名字应该都使用大写字母，并且指出该常量完整含义。如果一个常量名称由多个单词组成，则应该用下划线来分割这些单词。   
例如：MAX\_VALUE

**参数的命名**

参数的命名规范和方法的命名规范相同，而且为了避免阅读程序时造成迷惑，请在尽量保证参数名称为一个单词的情况下使参数的命名尽可能明确。

# 二、SVN操作规范

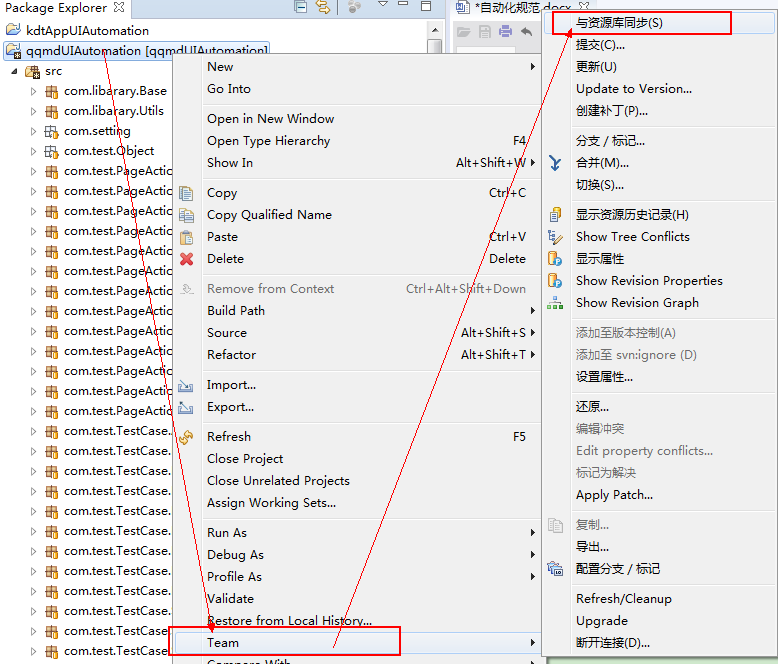
1. 对提交时采用写注释，若是更新文件选择【更新】，若是新增文件选择【新增】，也可以点击“Configure Comment Templates”来自定义注释模板



1. 不要提交、更新 setting包里的config.xml文件，testData里的excel文件
2. 不要提交本地自动生成的文件，例如自动生成的.classpath文件等，提交了这样的文件后，别人在更新后就与本地的环境冲突从而影响大家的工作。
3. 不用理会screenshots包里的文件，这是跑测试用例自动生成的截图，不需要提交和更新
4. 不用理会test-ouput包，这是跑测试用例自动生成的测试结果，不需要提交和更新

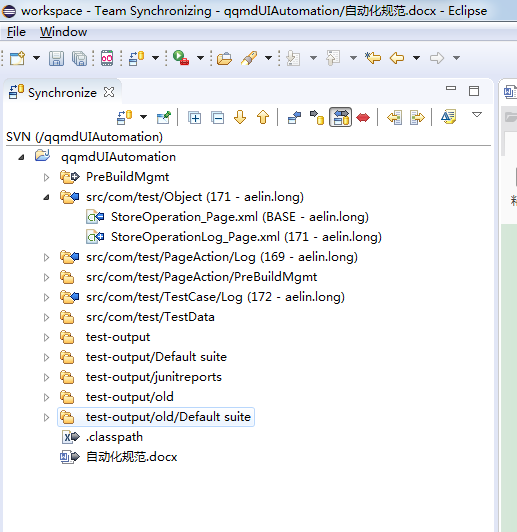
说明：

右键，与资源库同步



第一步：更新代码，有加号减号的，点击【更新】

第二步：灰色箭头代表与服务器上有不同，双击文件，查看不同处，一般情况下，是不会跟自己的代码有重复或冲突，那就从服务器上把新代码更新到本地



**三、效率提升**

**1、提高用例执行效率：**

1.1 减少操作步骤（如经过三四步才能打开我们要测试的页面，就可以直接通过网址来打开，减少不必要的操作）

1.2 中断页面加载（对于不影响测试的内容，设置超时时间）

1.3 设置等待时间，可以sleep固定，也可以检测某个元素出现后中断等待

1.4 配置testNG实现多线程（前提实现松耦合）

**2、测试用例执行的策略：**

2.1 监控：jenkins设置定时任务

2.2 必须回归的用例或冒烟目的：设置触发式执行，让自动化与build任务绑定

3.3 不需要经常执行的测试用例：如全量测试用例，jenkins创建任务，人工构建

**3、对case本身结构进行重构：**

3.1 抽出一些基本功能的case来做smoke test

3.2 把case按照产品功能的逻辑性进行从上到下分层归类,哪些case先run,哪些case后run

3.3 提高结果检查效率：用例里加上，执行过程记录error日志文件，并对log进行分类优化，方便查找问题

3.4 case本身有效性：删除合并那些测试点重复的case；想办法如何减少case的数量