Junit单元测试开发规范

## 测试规范

* 1. 单元测试类必须统一存放路径，单元测试代码必须写在如下工程目录：src/test/java，不允许写在业务代码目录下。
  2. 单元测试类及测试方法命名统一，测试类的命名规则是： 被测试的业务+ Test、被测试的接口+ Test、被测试的类+ Test，测试方法的命名规则是：test+测试方法。
  3. 所有public方法必须被单元测试覆盖，只测试public 和 proctected方法。
  4. 单元测试必须遵守AIR原则A: Automatic (自动化)，I：Independent （独立性），R：Repeatable（可重复）。
  5. 单元测试应该是全自动执行的，并且非交互式的。
  6. 保持单元测试的独立性。为了保证单元测试稳定可靠且便于维护，单元测试用例之 间决不能互相调用，也不能依赖执行的先后次序。
  7. 对于单元测试，要保证测试粒度足够小，有助于精确定位问题。单测粒度至多是类 级别，一般是方法级别。只有测试粒度小才能在出错时尽快定位到出错位置。单测不负责检查跨类或者跨系统的交互逻辑，那是集成测试的领域。
  8. 单元测试中不准使用System.out来进行人肉验证，必须使用assert来验证。
  9. 单元测试是可以重复执行的，不能受到外界环境的影响。
  10. 和数据库相关的单元测试，可以设定自动回滚机制，不给数据库造成脏数据 。或者对单元测试产生的数据有明确的前后缀标识。
  11. 对于不可测的代码建议做必要的重构，使代码变得可测，避免为了达到测 试要求而书写不规范测试代码。
  12. 单元测试作为一种质量保障手段，在项目提测前完成单元测试，不建议项目发布后 补充单元测试用例。

## 单元测试测试具体内容

* 1. 接口功能性测试: 接口功能的正确性,即保证接口能够被正常调用，并输出有效数据!
  2. 局部数据结构测试：保证数据结构的正确性。
  3. 边界条件测试：测试

-----> 变量无赋值(null)

-----> 变量是数值或字符

-----> 主要边界：最大值，最小值，无穷大

-----> 溢出边界：在边界外面取值+/-1

-----> 临近边界：在边界值之内取值+/-1

-----> 字符串的边界，引用 "变量字符"的边界

-----> 字符串的设置，空字符串

-----> 字符串的应用长度测试

-----> 空白集合

-----> 目标集合的类型和应用边界

-----> 集合的次序

-----> 变量是规律的，测试无穷大的极限，无穷小的极限

* 1. 所有独立代码测试：保证每一句代码，所有分支都测试完成，主要包括代码覆盖率，异常处理通路测试

----------> 语句覆盖率：每个语句都执行到了

----------> 判定覆盖率：每个分支都执行到了

----------> 条件覆盖率：每个条件都返回布尔

----------> 路径覆盖率：每个路径都覆盖到了

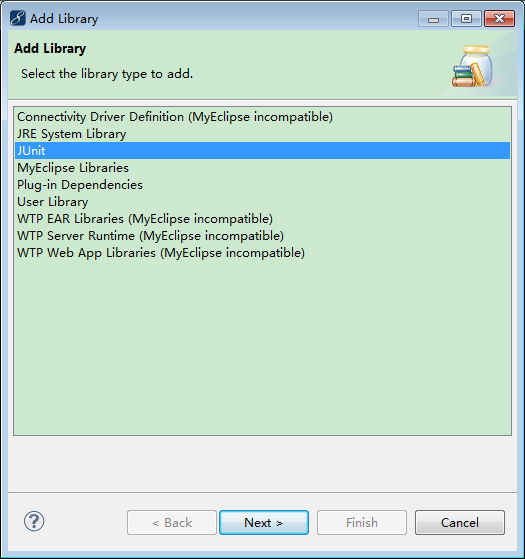
* 1. 异常模块测试，后续处理模块测试:是否包闭当前异常或者对异常形成消化,是否影响结果!

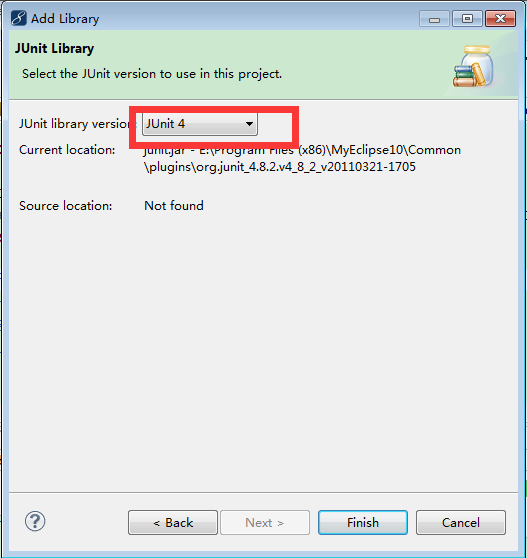
## Junit测试示例

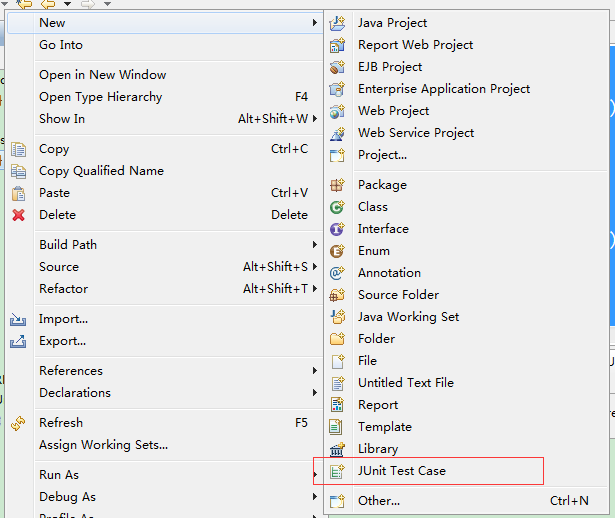
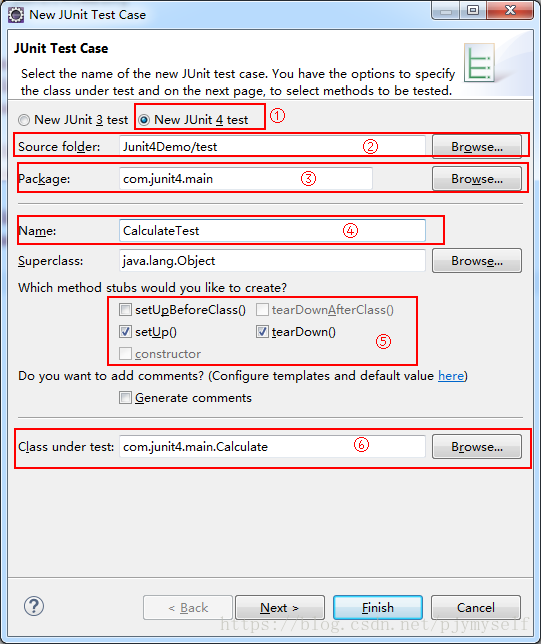
* 1. 环境搭建

这里使用的开发工具是MyEclipse，首先新建一个Java工程，将Junit4的jar包引入，eclipse和MyEclipse都集成了Junit的jar包，详细操作如下图。

1、新增一个Junit的jar包，版本选择Junit4





* 1. 新建一个Junit的测试类用来测试上面的测试方法，新增Junit的测试类方法如下： 

① Junit的版本选择 有Junit3 和Junit4两个版本可选，此处推荐使用Junit4较为方便编码。

② 测试文件位置 一般我们把测试代码都放在 “test” 文件夹里，此处选择 “项目名/test>”

③ 包名 测试文件所属的包名，这里与被测文件包名保持一致。

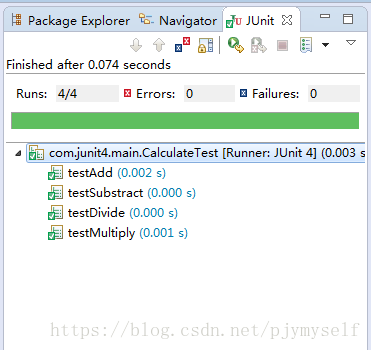
④ 测试文件名 一般以被测文件名为基础， “被测文件名+Test”

⑤ 生成测试方法 此处的测试方法可按需要进行勾选。

⑥ 被测文件 这里需指定这个test case是测试哪个文件的。

* 1. 编写测试代码

1. 构建被测对象   
   2、调用所要测试的函数   
   3、断言
   1. 运行测试代码

选择测试文件，右键点击 Run As -> JUnit Test 来执行测试代码。执行结果如下图：

## Junit一些常用注解

* @Test:将一个普通方法修饰成一个测试方法
  + @Test(excepted=xx.class): xx.class表示异常类，表示测试的方法抛出此异常时，认为是正常的测试通过的
  + @Test(timeout=毫秒数) :测试方法执行时间是否符合预期
* @BeforeClass： 会在所有的方法执行前被执行，static方法
* @AfterClass：会在所有的方法执行之后进行执行，static方法
* @Before：会在每一个测试方法被运行前执行一次
* @After：会在每一个测试方法运行后被执行一次
* @Ignore：所修饰的测试方法会被测试运行器忽略
* @RunWith：可以更改测试运行器org.junit.runner.Runner
* Parameters：参数化注解