Junit单元测试

## 优点

1. 提升软件质量
2. 促进代码优化
3. 提升研发效率
4. 增加重构自信

## 单元测试要求

1. 单元测试要求符合AIR要求（Automatic 自动化 Independent 独立性 Repeatable 可重复）
2. 单元测试要求测试粒度足够小，有利于定位问题，一般需要符合（BCDE原则）

即：B：Border 边界值测试 C:Correct 正确写入得到预期结果 D:Design 与设计文档符合 E:Error 与预期错误一致，证明程序运行时，发生正确的错误的程序正确性

1. 代码的覆盖率：包含了类覆盖和方法覆盖两种，类覆盖是指类中的代码或变量能够执行到，方法覆盖是指在方法中，某个方法被调用了，但是一般这种是粗粒度的情况。

方法覆盖又包含了：行覆盖、分支覆盖、条件判定覆盖

## 单元测试API\*\*\*

|  |
| --- |
| @After  运行在一个方法执行完成后，一般是释放资源  import org.junit.After; import org.junit.Test;  public class AfterTest {  @Test  public void test1() {  System.*out*.println("test1");  }  @Test  public void test2() {  System.*out*.println("test2");  }  @After  public void testAfter() {  System.*out*.println("----after -------");  } } |
| @Test  用于测试，用于方法之上  import org.junit.Test;  public class TestTest {   @Test  public void testTest () {  System.*out*.println("this is testTest");  }  }  参数1：**timeout=100 超时，如果超过timeout标记的时间，那么就报异常，单位毫秒**  @Test(timeout = 100) public void testOutTime() throws InterruptedException {  Thread.*sleep*(1); }  参数2：**expected 期望异常**  @Test(expected = ArithmeticException.class) public void expectTest() {  int a = 1/0; } |

|  |
| --- |
| @Before  用于一个方法上面  @Before public void beforeTest() {  System.*out*.println("before test"); } |

|  |
| --- |
| @BeforeClass  用于此类执行之前，一般是静态的方法  @BeforeClass public static void beforClassTest () {  System.*out*.println("before class Test"); } |

|  |
| --- |
| @AfterClass  用于此类执行之后，一般是静态的方法  @AfterClass public static void afterClassTest() {  System.*out*.println("after class test"); } |

|  |
| --- |
| @Assert  **assertArrayEquals用于 比较数组** [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(byte[],%20byte[]))(byte[] expecteds, byte[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(int[],%20int[]))(int[] expecteds, int[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(long[],%20long[]))(long[] expecteds, long[] actuals) [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.Object[],%20java.lang.Object[]))([Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)[] expecteds, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(short[],%20short[]))(short[] expecteds, short[] actuals)  @Test public void test1 () {  Student[] mockStudent = new Student[]{new Student("12", "小米")};  Assert.*assertArrayEquals*(sArray, mockStudent); }    \*注意：如果是对象，那么需要重写equals和hasCode方法  **assertArrayEquals(\*[] a1, \*[] a2, \* delta)**  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(double[],%20double[],%20double))(double[] expecteds, double[] actuals, double delta)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(float[],%20float[],%20float))(float[] expecteds, float[] actuals, float delta)  delta ：范围内  @Test public void test2() {  double[] ex = {1,2};  double[] ac = {3,4};  Assert.*assertArrayEquals*(ex, ac, 2); }    [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20char[],%20char[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, char[] expecteds, char[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20byte[],%20byte[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, byte[] expecteds, byte[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20int[],%20int[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, int[] expecteds, int[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20long[],%20long[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, long[] expecteds, long[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20java.lang.Object[],%20java.lang.Object[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)[] expecteds, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)[] actuals)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20short[],%20short[]))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, short[] expecteds, short[] actuals)  @Test public void test3 () {  Student[] mockS = {new Student("12", "小米6")};  Assert.*assertArrayEquals*("调用失败了", mockS, sArray); }    **assertArrayEquals(msg,\*[],\*[],delta) msg :错误提示，delta：相差值** [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20double[],%20double[],%20double))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, double[] expecteds, double[] actuals, double delta)  [**assertArrayEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertArrayEquals(java.lang.String,%20float[],%20float[],%20float))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, float[] expecteds, float[] actuals, float delta)  @Test public void test4 () {  double[] da1 = {1,1,1};  double[] da2 = {3,-1,5};  Assert.*assertArrayEquals*("error", da1, da2, 2); }    [**assertEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertEquals(double,%20double,%20double))(double expected, double actual, double delta)  [**assertEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertEquals(java.lang.String,%20long,%20long))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, long expected, long actual)  [**assertEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertEquals(double,%20double,%20double))(double expected, double actual, double delta)  [**assertEquals**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertEquals(java.lang.String,%20java.lang.Object,%20java.lang.Object))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) expected, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) actual)  **判断返回是否为false** [**assertFalse**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertFalse(boolean))(boolean condition)  [**assertFalse**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertFalse(java.lang.String,%20boolean))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, boolean condition)  [**assertTrue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertTrue(boolean))(boolean condition)  [**assertTrue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertTrue(java.lang.String,%20boolean))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, boolean condition)  @Test public void tesst5 () {  Assert.*assertTrue*("返回了false", true); }    判断是否为null  [**assertNotNull**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNotNull(java.lang.Object))([Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) object)  [**assertNotNull**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNotNull(java.lang.String,%20java.lang.Object))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) object)  [**assertNull**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNull(java.lang.String,%20java.lang.Object))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) object)  [**assertNull**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNull(java.lang.Object))([Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) object)  @Test public void test6 () {  Student s = new Student();  s = null;  Assert.*assertNotNull*("对象为Null了", s); }    判断是否是同一个对象  [**assertSame**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertSame(java.lang.Object,%20java.lang.Object))([Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) expected, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) actual)  [**assertSame**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertSame(java.lang.String,%20java.lang.Object,%20java.lang.Object))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) expected, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) actual)  [**assertNotSame**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNotSame(java.lang.Object,%20java.lang.Object))([Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) unexpected, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) actual)  [**assertNotSame**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertNotSame(java.lang.String,%20java.lang.Object,%20java.lang.Object))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) unexpected, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) actual)  @Test  public void test7() {  Student s1 = new Student("12", "12");  Student s2 = new Student("12", "12"); // s2 = s1;  Assert.*assertSame*("对象不同", s1,s2);   }    肯定失败  [**fail**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#fail())()  [**fail**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#fail(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message)  @Test public void test8 () {  Assert.*fail*(); }  [**assertThat**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertThat(T,%20org.hamcrest.Matcher))(T actual, [Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<T> matcher  [**assertThat**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/junit/Assert.html#assertThat(java.lang.String,%20T,%20org.hamcrest.Matcher))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) reason, T actual, [Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<T> matcher) |

|  |
| --- |
| [Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)  --->coreMatchs  代替任何事物，永远返回true  [**anything**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#anything())()  [**anything**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#anything(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) description)  @Test public void test9 () {  Assert.*assertThat*("lala", "11", CoreMatchers.*anything*()); }  任何类型对于type的  [**any**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#any(java.lang.Class))([Class](http://java.sun.com/javase/6/docs/api/java/lang/Class.html?is-external=true)<T> type)  基于 关键字 instance of，一般适合多态  @Test public void test10 () {  Assert.*assertThat*("333", new Student("2","3"), CoreMatchers.*any*(Student.class));  System.*out*.println(CoreMatchers.*any*(Student.class).matches(new Student())); }  判断两个对象是否相等 基于 == 使用的是Object类中的equals  [**equalTo**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#equalTo(T))(T operand)  @Test public void test11 () {  List<Student> ls = Arrays.*asList*(new Student("11", "22"));  Assert.*assertThat*("error", new Student(), CoreMatchers.*equalTo*("23")); }  用于泛型的判断  [**instanceOf**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#instanceOf(java.lang.Class))([Class](http://java.sun.com/javase/6/docs/api/java/lang/Class.html?is-external=true)<?> type  @Test public void test12 () {  Assert.*assertThat*("error", new Student(), CoreMatchers.*instanceOf*(Student.class)); }  与equals(T t)一样  [**is**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#is(T))(T value)  [**is**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#is(org.hamcrest.Matcher))([Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<T> matcher)  判断是否是同一类  [**not**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#not(T))(T value)  @Test public void test14 () {  Assert.*assertThat*("error", new Student(), CoreMatchers.*not*(new Student())); }  **就是给一个不为空的值，只要不为空就直接真。** [**notNullValue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#notNullValue())()  [**notNullValue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#notNullValue(java.lang.Class))([Class](http://java.sun.com/javase/6/docs/api/java/lang/Class.html?is-external=true)<T> type) 【标注这个类不为null】  @Test public void test15 () {  Assert.*assertThat*("error", null, CoreMatchers.*notNullValue*()); }  @Test public void test16() {  Assert.*assertThat*("error", null, CoreMatchers.*notNullValue*(Integer.class)); }  **判断一个值为null**  [**nullValue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#nullValue())()  [**nullValue**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#nullValue(java.lang.Class))([Class](http://java.sun.com/javase/6/docs/api/java/lang/Class.html?is-external=true)<T> type)  判断是否是同一个对象，完全相同的初始化  [**sameInstance**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#sameInstance(T))(T object)  @Test public void test17 () {  final Student student = new Student();  Assert.*assertThat*("error", student, CoreMatchers.*sameInstance*(student)); }  后面的值要全部相等  [**allOf**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#allOf(java.lang.Iterable))([Iterable](http://java.sun.com/javase/6/docs/api/java/lang/Iterable.html?is-external=true)<[Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<? extends T>> matchers)  [**allOf**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#allOf(org.hamcrest.Matcher...))([Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<? extends T>... matchers)  new Student()用于约束super Student的  @Test public void test18 () {  List<Matcher<? super Student>> ls = new ArrayList<>();  ls.add(CoreMatchers.*instanceOf*(Student.class));  ls.add(CoreMatchers.*instanceOf*(Student.class));  ls.add(CoreMatchers.*not*(String.class));  Assert.*assertThat*("error", new Student(), CoreMatchers.*allOf*(ls)); }  [**describedAs**](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/CoreMatchers.html#describedAs(java.lang.String,%20org.hamcrest.Matcher,%20java.lang.Object...))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) description, [Matcher](http://tool.oschina.net/uploads/apidocs/junit-4.10/org/hamcrest/Matcher.html)<T> matcher, [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)... values)  是否包含某个字符串  *containsString（String s）*  是否按某个串开头  CoreMatchers.*startsWith*(String startWith)  是否以某个串结尾  *endsWith*(String endWith)  用于测试&&和或||  both().and() 或者使用both().or()  @Test public void test26 () {  Assert.*assertThat*("albumen", CoreMatchers.*both*(CoreMatchers.*containsString*("a")).and(CoreMatchers.*containsString*("b"))  .and(CoreMatchers.*containsString*("e")).or(CoreMatchers.*containsString*("z"))); }    或者  *either().or()*  @Test public void test27 () {  Assert.*assertThat*("albumen", CoreMatchers.*either*(CoreMatchers.*containsString*("a"))  .or(CoreMatchers.*containsString*("a"))); }    是否包含某项  *hasItem(? Extend Object)*  是否每一项都包含  *everyItem(*CoreMatchers c*)*  @Test public void test29() {  Assert.*assertThat*(Arrays.*asList*( "143", "23", "3" ),  CoreMatchers.*everyItem*(CoreMatchers.*containsString*("3"))); } |

|  |
| --- |
| Suite：  容器注解，[需要配合@RunWith(Suite.class)](mailto:需要配合@RunWith(Suite.class))此类是一个Suite容器，结合@SuiteClasses(Class[] clazzs) 一起使用，将clazzs数组放到Suite容器中一起运行  @RunWith(Suite.class) @Suite.SuiteClasses({Demo1.class, Demo3.class}) public class Demo2 {  } |

# MockMvc:

首先引入对应的pom.xml文件

<!--测试的依赖包-->  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
</dependency>

进行静态导入：

import static org.junit.Assert.\*;  
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;  
import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.*print*;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;  
import static org.springframework.test.web.servlet.setup.MockMvcBuilders.\*;

其次在对应的test包中对Controller层进行测试：

@SpringBootTest  
@RunWith(SpringRunner.class)  
public class GoodsControllerTest {  
  
 @Autowired  
 private WebApplicationContext wac;  
  
 private MockMvc mockGoodsMvc;  
  
 @Before  
 public void setUp() throws Exception {  
 mockGoodsMvc = MockMvcBuilders.*webAppContextSetup*(this.wac).build();  
 }

}

### 验证405

|  |
| --- |
| */\*\*  \** ***@deprecation:*** *验证get和post请求 ，验证405  \*/* @Test public void goodsById() throws Exception {  mockGoodsMvc.perform(*post*("/goods/1"))  .andExpect(*status*().is(405)); } |

### 验证请求参数

#### RESTful请求参数

|  |
| --- |
| @Test public void getById3 () throws Exception {  mockGoodsMvc.perform(*get*("/goods/{id}", "34")).andExpect(*status*().isOk()); } |

#### url带参数

|  |
| --- |
| @Test public void getById4 () throws Exception {  mockGoodsMvc.perform(*get*("/goods").param("id", "34")).andExpect(*status*().isOk()); } |

#### 添加一个实体

|  |
| --- |
|  |