## 一段常见的代码

单元测试是项目的重要组成部分。尤其是对持续发展的产品，单元测试在后期的维护，回归有重要等方面有重要作用。

  这样代码在项目中随处可见，看看我们应该如何测试

**Java代码  [IMG_256](http://jilen.iteye.com/blog/javascript:void())**

1. **public** **class** NotifyService {
2. **private** UserCenter uc;
3. **private** MessageCenter mc;
5. **public** **void** sendMessage(**long** userId, String message) {
6. String email = uc.getUser(userId).getEmail();
7. mc.sendEmail(email, message);
8. }
10. **public** **void** setUc(UserCenter uc) {
11. **this**.uc = uc;
12. }
14. **public** **void** setMc(MessageCenter mc) {
15. **this**.mc = mc;
16. }
17. }

UserCenter和MessageCenter都是接口，User是一个简单的JavaBean

由于uc和mc乃外部依赖，此类不需也不应保证uc和mc的正确性，此类只需保证：

  假设uc和mc是正确的，那么我也是正确的。

所以需要隔离依赖--使用****mock****

## 使用EasyMock

如果使用EasyMock，此类之单元测试或许如下[需要static import org.easymock.EasyMock类的相关方法]

**Java代码  [IMG_257](http://jilen.iteye.com/blog/javascript:void())**

1. **public** **class** EasyMockNotifyServiceTest {
2. **private** NotifyService notifyService;
3. **private** UserCenter uc;
4. **private** MessageCenter mc;
6. @Before
7. **public** **void** setUp() {
8. notifyService = **new** NotifyService();
9. uc = createMock(UserCenter.**class**);
10. mc = createMock(MessageCenter.**class**);
11. notifyService.setUc(uc);
12. notifyService.setMc(mc);
13. }
15. @Test
16. **public** **void** testSendMessage() {
17. Long id = 1L;
18. String email = "foo@bar";
19. String message = "hello";
20. expect(uc.getUser(id)).andReturn(createUserWithEmail(email));
21. mc.sendEmail(eq(email), eq(message));
22. replay(uc);
23. replay(mc);
24. notifyService.sendMessage(id, message);
25. verify(mc);//verify a mocked behavior
26. }
28. **private** User createUserWithEmail(String email) {
29. User user = **new** User();
30. user.setEmail(email);
31. **return** user;
32. }
33. }

   当然，此测试并不充分，easymock需要mock每个依赖，对mock的所有方法调用作expect，然后验证需要验证的行为。

**Java代码  [IMG_258](http://jilen.iteye.com/blog/javascript:void())**

1. mc.sendEmail(eq(email), eq(message));

    此处代码理解起来有些怪异，实际上，此处的语义应该是 expect mc.sendEmail... called，即方法执行完毕后sendMail必以正确的参数调用，奈何java 泛型中并未覆盖void型，所以通常会在mock行为调用之后

加上此行以明确语义

**Java代码  [IMG_259](http://jilen.iteye.com/blog/javascript:void())**

1. expectLastCall().times(1);

    另外一个值得注意的地方

**Java代码  [IMG_260](http://jilen.iteye.com/blog/javascript:void())**

1. verify(mc);//verify a mocked behavior

    并未验证uc，我的想法是对于uc我们需要它提供数据(桩)，而不需要验证其行为。

## 使用Mockito

    假使使用Mockito，单元测试也许是这个样子的[需要static import org.mockito.Mockito类的相关方法]

**Java代码  [IMG_261](http://jilen.iteye.com/blog/javascript:void())**

1. **public** **class** NotifyServiceTest {
2. **private** NotifyService notifyService;
3. **private** UserCenter uc;
4. **private** MessageCenter mc;
6. @Before
7. **public** **void** setUp() {
8. notifyService = **new** NotifyService();
9. uc = mock(UserCenter.**class**);
10. mc = mock(MessageCenter.**class**);
11. notifyService.setUc(uc);
12. notifyService.setMc(mc);
13. }
15. @Test
16. **public** **void** testSendMessage() {
17. **long** userId = 1L;
18. String email = "foo@bar";
19. when(uc.getUser(userId)).thenReturn(createUserWithEmail(email));
20. notifyService.sendMessage(userId, "hello");
21. verify(mc).sendEmail(eq(email), eq("hello"));
22. }
24. **private** User createUserWithEmail(String email) {
25. User user = **new** User();
26. user.setEmail(email);
27. **return** user;
28. }
30. }

看到testSendMessage方法

**Java代码  [IMG_262](http://jilen.iteye.com/blog/javascript:void())**

1. @Test
2. **public** **void** testSendMessage() {
3. **long** userId = 1L;
4. String email = "foo@bar";
5. when(uc.getUser(userId)).thenReturn(createUserWithEmail(email));
6. notifyService.sendMessage(userId, "hello");
7. verify(mc).sendEmail(eq(email), eq("hello"));
8. }

语义不言自明

测试前，从uc获得email

测试后，必须调用mc.sendEmail，所以验证之