1、 Mockito

1.1 快速上手

代码示例：

```
1.  public class UserDao {
2.
3.      public boolean insert(User user){
4.          throw new UnsupportedOperationException();
5.      }
6.  }
7.
8.  public class UserService {
9.
10.     private UserDao userDao;
11.
12.     public UserService(UserDao userDao){
13.         this.userDao = userDao;
14.     }
15.
16.     public boolean insertUser(User user){
17.         if(userDao.insert(user)){
18.             return true;
19.         }else{
20.             return false;
21.         }
22.     }
23. }
24.
25. public class UserServiceTest {
26.
27.     //实例化并 Mock 对象
28.     @Mock
29.     private UserDao userDao;
30.
31.     //实例化并将 Mock 的对象植注入到依赖它的对象中，这里是 Constructor 注入
32.     @InjectMocks
33.     private UserService userService;
34.
35.     @Before
36.     public void init(){
37.         //规定写法，使 Mockito 注解生效
38.         MockitoAnnotations.initMocks(this);
39.     }
40.
41.     @Test
```

```
42.     public void testInsertUser(){
43.         User user = new User();
44.         //stubbing
45.         when(userDao.insert(user)).thenReturn(true);
46.         //assert
47.         assertThat(userService.insertUser(user),is(true));
48.     }
49. }
```

1.2 Stubbing

1.2.1 Stubbing 两种方式

1.2.1.1 针对有返回值的方法

Mockito.doThrow().when(*mockInstance*).*method*();
Mockito.doReturn().when(*mockInstance*).*method*();
Mockito.doAnswer().when(*mockInstance*).*method*();
Mockito.doCallRealMethod().when(*mockInstance*).*method*();
Mockito.when(*mockInstance.method*()).thenReturn();
Mockito.when(*mockInstance.method*()).thenCallRealMethod();
Mockito.when(*mockInstance.method*()).thenThrow();
Mockito.when(*mockInstance.method*()).thenAnswer();

1.2.1.2 针对 void 方法

Mockito.doCallRealMethod().when(*mockInstance*).*method*();
Mockito.doThrow().when(*mockInstance*).*method*();
Mockito.doAnswer().when(*mockInstance*).*method*();
Mockito.doNothing().when(*mockInstance*).*method*();

1.2.2 Stubbing 连续调用

代码示例：

```
1.  @Test
2.      public void testInsertUser() {
3.          User user = new User();
4.          Mockito.when(userDao.insert(user))
5.                  .thenReturn(true)
6.                  .thenReturn(false)
7.                  .thenThrow(new RuntimeException())
8.                  .thenAnswer(invocation -> true);
9.          assertThat(userService.insertUser(user),is(true));
10.         assertThat(userService.insertUser(user),is(false));
11.         assertThrows(RuntimeException.class,()->userService.insertUser(user)
    );
12.         assertThat(userService.insertUser(user),is(true));
13.     }
```

注：最后一个 stub 是继续生效的。

1.2.3 Argument matchers

1.2.3.1 org.mockito.ArgumentMatchers

因为只有在调用方法时传参和 Stubbing 时规定的参数相同时，Stubbing 才会生效，所以要想在传任何参数或某些符合规则的参数时 Stubbing 都生效，则需要参数匹配器。

any()	匹配任何内容，包括 null 和可变参数
any(Class<T> type)	匹配给定类型的任何对象，不包括 null
anyBoolean()	任何 boolean 或非 null Boolean
anyByte()	任何 byte 或者非 null Byte
anyChar()	任何 char 或者非 null Character
anyCollection()	任何非 null Collection
anyDouble()	任何 double 或者非 null Double
anyFloat()	任何 float 或者非 null Float
anyInt()	任何 int 或非 null Integer
anyIterable()	任何非 null Iterable
anyList()	任何非 null List
anyLong()	任何 long 或者非 null Long
anyMap()	任何非 null Map
anySet()	任何非 null Set
anyShort()	任何 short 或者非 null Short
anyString()	任何非 null String
contains(String substring)	String 包含给定子字符串的参数
endsWith(String suffix)	String 以给定后缀结尾的参数
eq(*field or class*)	等于给定值的参数
same(T value)	与给定值相同的对象参数。
isA(Class<T> type)	Object 实现给定类的参数
isNotNull() / notNull()	不是 null 的参数
isNull()	是 null 的参数
matches(Pattern pattern)	Pattern 与给定正则表达式匹配的参数
matches(String regex)	String 与给定正则表达式匹配的参数
nullable(Class<T> clazz)	参数要么是 null 或者给定类型的
startsWith(String prefix)	String 以给定前缀开头的参数
基本类型自定义参数匹配器：	
intThat(ArgumentMatcher<Integer> matcher)	允许创建自定义 int 参数匹配器
floatThat(ArgumentMatcher<Float> matcher)	允许创建自定义 float 参数匹配器
booleanThat(ArgumentMatcher<Boolean> matcher)	允许创建自定义 boolean 参数匹配器。
byteThat(ArgumentMatcher<Byte> matcher)	允许创建自定义 byte 参数匹配器。
charThat(ArgumentMatcher<Character>	允许创建自定义 char 参数匹配器。

matcher)	
doubleThat(ArgumentMatcher<Double> matcher)	允许创建自定义 double 参数匹配器。
longThat(ArgumentMatcher<Long> matcher)	允许创建自定义 long 参数匹配器。
shortThat(ArgumentMatcher<Short> matcher)	允许创建自定义 short 参数匹配器。
引用类型自定义参数匹配器:	
argThat(ArgumentMatcher<T> matcher)	允许创建自定义参数匹配器。

代码示例：

```
1.  @Test
2.  public void testInsertUser() {
3.      Mockito.when(userDao.insert(any())).thenReturn(true);
4.      assertThat(userService.insertUser(new User()),is(true));
5.      Mockito.when(userDao.insert(any(User.class))).thenReturn(false);
6.      assertThat(userService.insertUser(new User("user",18)),is(false));
7.  }
```

1.2.3.2 自定义参数匹配器

代码示例：

```
1.  @Test
2.  public void testInsertUser() {
3.      Mockito.when(userDao.insert(argThat(new ArgumentMatcher<User>() {
4.          @Override
5.          public boolean matches(User user) {
6.              //只匹配 user.name 为 tip 的 User 对象
7.              if(user.name.equals("tip")){
8.                  return true;
9.              }
10.             return false;
11.         }
12.     }))).thenReturn(true);
13.     assertThat(userService.insertUser(new User("tip",18)),is(true));
14.     //利用 java 8 Lambda 表达式简化
15.     Mockito.doReturn(false).when(userDao)
16.             .insert(argThat(user -> user.name.equals("sandbox")));
17.     assertThat(userService.insertUser(new User("sandbox",18)),is(false));
18. }
```

1.2.3.3 要注意的问题

当方法有多个参数，其中任何一个参数使用了参数匹配器，那么其它的参数也一定要使

用参数匹配器，否则会报错。
代码示例：

```
1.  public class UserDao {
2.
3.      public boolean insert(String name,int age){
4.          throw new UnsupportedOperationException();
5.      }
6.  }
7.
8.  public class UserService {
9.
10.     private UserDao userDao;
11.
12.     public UserService(UserDao userDao){
13.         this.userDao = userDao;
14.     }
15.
16.     public boolean insertUser(String name,int age) {
17.         if (userDao.insert(name,age)) {
18.             return true;
19.         } else {
20.             return false;
21.         }
22.     }
23. }
24.
25. public class UserServiceTest {
26.
27.     @Mock
28.     private UserDao userDao;
29.
30.     @InjectMocks
31.     private UserService userService;
32.
33.     @Before
34.     public void init(){
35.         MockitoAnnotations.initMocks(this);
36.     }
37.
38.     @Test
39.     public void testInsertUser() {
40.         //以下一行代码会报错
41.         //Mockito.when(userDao.insert(anyString(),18)).thenReturn(true);
```

```
        //以下一行代码为正确使用
        Mockito.when(userDao.insert(anyString(),eq(18))).thenReturn(true);
        assertThat(userService.insertUser("tip",18),is(true));
    }
}
```

当方法参数为基本类型时，不要使用 argThat() 来自定义参数匹配器，否则会出现 NPE
当方法参数为包装类型时，是可以使用基本类型的自定义参数匹配器的
代码示例：

```
  Mockito.doReturn(true).when(userDao).insert(
                argThat(name -> name.equals("tip")),
                //以下一行代码会抛出空指针异常
                argThat(age -> age >= 18)
        );
   //正确使用
   Mockito.doReturn(true).when(userDao).insert(
                argThat(name -> name.equals("tip")),
                intThat(age -> age >= 18)
        );
```

1.2.4 Answer

Answer 用于自定义方法的执行和返回结果。
代码示例：

```
@Test
    public void testInsertUser() {
        Mockito.doAnswer(new Answer() {
            @Override
            public Object answer(InvocationOnMock invocationOnMock) throws T
hrowable {
                //自定义方法执行过程，返回的类型应该和原来的返回类型一致
                if(invocationOnMock.getArgument(0).equals("tip")
                        && invocationOnMock.getArgument(1).equals(18)){
                    return true;
                }
                return false;
            }
        }).when(userDao).insert(anyString(),anyInt());
        assertThat(userService.insertUser("tip",18),is(true));
        assertThat(userService.insertUser("sandbox",18),is(false));
        //利用 java 8 Lambda 表达式简化
        Mockito.when(userDao.insert(anyString(),anyInt())).thenAnswer(i -> {

            if(i.getArgument(0).equals("tip")
```

```
                    && i.getArgument(1).equals(18)){
                return false;
            }
            return true;
        });
        assertThat(userService.insertUser("tip",18),is(false));
        assertThat(userService.insertUser("sandbox",18),is(true));
    }
```

1.3 Verify

1.3.1 验证调用次数

代码示例：

```
@Test
    public void testInsertUser() {
        userService.insertUser("tip",0);
        userService.insertUser("sandbox",0);
        userService.insertUser("sandbox",0);
        //默认验证 times(1)
        verify(userDao).insert("tip",0);
        verify(userDao,times(2)).insert("sandbox",0);
        verify(userDao,never()).insert("cloud",0);
        verify(userDao,atLeastOnce()).insert("tip",0);
        verify(userDao,atLeast(2)).insert("sandbox",0);
        verify(userDao,atMost(1)).insert("cloud",0);
        //验证是否多余的调用
        verifyZeroInteractions(userDao);
        verifyNoMoreInteractions(userDao);
    }
```

1.3.2 验证调用顺序

代码示例：

```
@Test
    public void testInsertUser() {
        userService.insertUser("tip",0);
        userService.insertUser("sandbox",0);
        //为 mock 对象创建一个 inOrder 对象
        // 这里可以为多个 mock 对象创建,例如 InOrder inOrder = inOrder(userDao,....);
        InOrder inOrder = inOrder(userDao);
        //验证执行顺序
        inOrder.verify(userDao).insert("tip",0);
        inOrder.verify(userDao).insert("sandbox",0);
    }
```

1.3.3 验证调用超时

代码示例：

```
@Test
    public void testInsertUser() {
        userService.insertUser("tip",0);
        userService.insertUser("sandbox",0);
        verify(userDao,timeout(100)).insert("tip",0);
        verify(userDao,timeout(100).times(1)).insert("sandbox",0);
    }
```

1.3.4 自定义验证失败消息

代码示例：

```
@Test
    public void testInsertUser() {
        userService.insertUser("tip",0);
        userService.insertUser("sandbox",0);
        verify(userDao,description("应该只执行一次")).insert("tip",0);
        verify(userDao,timeout(100).description("应该在 0.1 秒内完成
    ")).insert("sandbox",0);
    }
```

1.4 Spy

1.4.1 Spy 与 Mock 的区别

当 Spy 对象没有 Stubbing 时，对其方法的调用会使用其真实的方法，当 Spy 对象有 Stubbing 时，则会被 mock。

代码示例：

```
public class UserServiceTest {

    @Spy
    private UserDao userDao;

    @InjectMocks
    private UserService userService;

    @Before
    public void init(){
        MockitoAnnotations.initMocks(this);
    }

    @Test
    public void testInsertUser() {
        assertThrows(UnsupportedOperationException.class,
```

```
                () -> userService.insertUser("tip",0));
        Mockito.doReturn(true).when(userDao).insert("tip",0);
        assertThat(userService.insertUser("tip",0),is(true));
    }
}
```

1.4.2 Spy 的重要问题

因为调用 Spy 对象的方法时会调用到真实方法，所以使用 Stubbing 时建议使用 doXXX().when().method()的 Stubbing 方式。

代码示例：

```
@Test
    public void testInsertUser() {
        //由于调用了真实方法，以下一行代码会抛出 UnsupportedOperationException
        //Mockito.when(userDao.insert("tip",0)).thenReturn(true);
        Mockito.doReturn(true).when(userDao).insert("tip",0);
        assertThat(userService.insertUser("tip",0),is(true));
    }
```

2、 PowerMock

2.1 模拟 final 类

代码示例：

```
final public class UserDao {

    public boolean insert(User user){
        throw new UnsupportedOperationException();
    }
}

public class UserService {

    private UserDao userDao;

    public UserService(UserDao userDao){
        this.userDao = userDao;
    }

    public boolean insertUser(User user) {
        if (userDao.insert(user)) {
            return true;
        } else {
            return false;
        }
    }
```

```
}

//使用 PowerMock 提供的 Runner
@RunWith(PowerMockRunner.class)
//说明测试涉及到的类
@PrepareForTest({UserDao.class,UserService.class})
public class UserServiceTest {

    @Mock
    private UserDao userDao;

    @InjectMocks
    private UserService userService;

    @Test
    public void testInsertUser() throws Exception {
        User user = new User();
        PowerMockito.when(userDao.insert(user)).thenReturn(true);
        assertThat(userService.insertUser(user),is(true));
    }
}
```

2.2 模拟构造方法（模拟局部的引用变量）

代码示例：

```
public class UserService {

    public boolean insertUser(User user){
        UserDao userDao = new UserDao();
        if(userDao.insert(user)){
            return true;
        }else{
            return false;
        }
    }
}

//使用 PowerMock 提供的 Runner
@RunWith(PowerMockRunner.class)
//说明测试涉及到的类
@PrepareForTest({UserDao.class,UserService.class})
public class UserServiceTest {
    @Test
    public void testInsertUser() throws Exception {
        User user = new User();
```

```
        UserService userService = new UserService();
        //首先 mock 出一个对象
        UserDao userDao = PowerMockito.mock(UserDao.class);
        //模拟构造方法,返回 mock 的对象
        PowerMockito.whenNew(UserDao.class).withNoArguments().thenReturn(use
rDao);
        PowerMockito.when(userDao.insert(user)).thenReturn(true);
        assertThat(userService.insertUser(user),is(true));
    }
}
```

2.3 模拟静态方法

代码示例：

```
public class UserDao {

    public static boolean insert(User user){
        throw new UnsupportedOperationException();
    }
}

public class UserService {

    public boolean insertUser(User user){
        if(UserDao.insert(user)){
            return true;
        }else{
            return false;
        }
    }
}

@RunWith(PowerMockRunner.class)
@PrepareForTest({UserDao.class,UserService.class})
public class UserServiceTest {

    @Test
    public void testInsertUser(){
        User user = new User();
        UserService userService = new UserService();
        PowerMockito.mockStatic(UserDao.class);
        PowerMockito.when(UserDao.insert(user)).thenReturn(true);
        assertThat(userService.insertUser(user),is(true));
    }
}
```

2.4 模拟 final 方法

代码示例：

```
1.  public class UserDao {
2.
3.      public final boolean insert(User user){
4.          throw new UnsupportedOperationException();
5.      }
6.  }
7.
8.  public class UserService {
9.
10.     private UserDao userDao;
11.
12.     public UserService(UserDao userDao){
13.         this.userDao = userDao;
14.     }
15.
16.     public boolean insertUser(User user) {
17.         if (userDao.insert(user)) {
18.             return true;
19.         } else {
20.             return false;
21.         }
22.     }
23. }
24.
25. @RunWith(PowerMockRunner.class)
26. @PrepareForTest({UserDao.class,UserService.class})
27. public class UserServiceTest {
28.
29.     @Test
30.     public void testInsertUser(){
31.         User user = new User();
32.         UserDao userDao = PowerMockito.mock(UserDao.class);
33.         UserService userService = new UserService(userDao);
34.         PowerMockito.when(userDao.insert(user)).thenReturn(true);
35.         assertThat(userService.insertUser(user),is(true));
36.     }
37. }
```

2.5 模拟私有方法

代码示例：

```java
public class UserService {

    private UserDao userDao;

    public UserService(UserDao userDao){
        this.userDao = userDao;
    }

    private boolean insertUser(User user) {
        if (userDao.insert(user)) {
            return true;
        } else {
            return false;
        }
    }

    public boolean useInsertUser(User user){
        return insertUser(user);
    }
}

@RunWith(PowerMockRunner.class)
@PrepareForTest({UserDao.class,UserService.class})
public class UserServiceTest {

    @Mock
    private UserDao userDao;

    @InjectMocks
    private UserService userService;

    @Test
    public void testInsertUser() throws Exception {
        User user = new User();
        UserService service = PowerMockito.spy(userService);
        //第一个参数为实例，第二个参数为方法名，后面的参数为方法参数列表
        PowerMockito.doReturn(true).when(service,"insertUser",user);
        assertThat(service.useInsertUser(user),is(true));
    }
}
```