# mock测试

### 一、依赖包

<dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>6.11</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>1.10.19</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest-all</artifactId>  
 <version>1.3</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.powermock</groupId>  
 <artifactId>powermock-api-mockito</artifactId>  
 <version>1.7.0</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.powermock</groupId>  
 <artifactId>powermock-module-junit4</artifactId>  
 <version>1.6.3</version>  
</dependency>  
<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
</dependency>  
<dependency>  
 <groupId>org.powermock</groupId>  
 <artifactId>powermock-core</artifactId>  
 <version>1.7.0</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>org.powermock</groupId>  
 <artifactId>powermock-module-testng</artifactId>  
 <version>1.7.0</version>  
 <scope>test</scope>  
</dependency>

（Junit不必须）

### 二、测试常见案例

测试类上的注解，（使用powerMock.mockstatic的类）需要加到prepareForTest中去

要测试的类使用InjectMock注释

要测试的类中的属性使用mock注释

beforeClass注释的初始化方法不可少

类继承PowerMockTestCase

@PrepareForTest({CatalogServiceImpl.class, UserSession.class,ObjectMapperFactory.class})  
public class CatalogControllerTest extends PowerMockTestCase {

@InjectMocks  
CatalogController catalogController;  
@Mock  
HttpServletRequest request;  
@Mock  
BaseDao baseDao;

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

}

案例一（在要测试的方法中有静态方法调用）：

UserLoginInfo loginUser = UserSession.*getLoginUserInfo*(request);

则需要在方法测试中增加

PowerMockito.*mockStatic*(UserSession.class);  
*when*(UserSession.*getLoginUserInfo*(*eq*(request))).thenReturn(new UserLoginInfo());

案例二（静态方法链式调用）：

TableCatalogTypePO catalogType = ObjectMapperFactory.*getObjectMapper*().readValue(json,TableCatalogTypePO.class);

则需要在测试方法中

PowerMockito.*mockStatic*(ObjectMapperFactory.class);  
ObjectMapper objectMapper = PowerMockito.*mock*(ObjectMapper.class);  
*when*(ObjectMapperFactory.*getObjectMapper*()).thenReturn(objectMapper);

TableCatalogTypePO catalogType = new TableCatalogTypePO();  
try {  
 *when*(objectMapper.readValue(*anyString*(), *eq*(TableCatalogTypePO.class))).thenReturn(catalogType);  
} catch (IOException e) {  
  
}

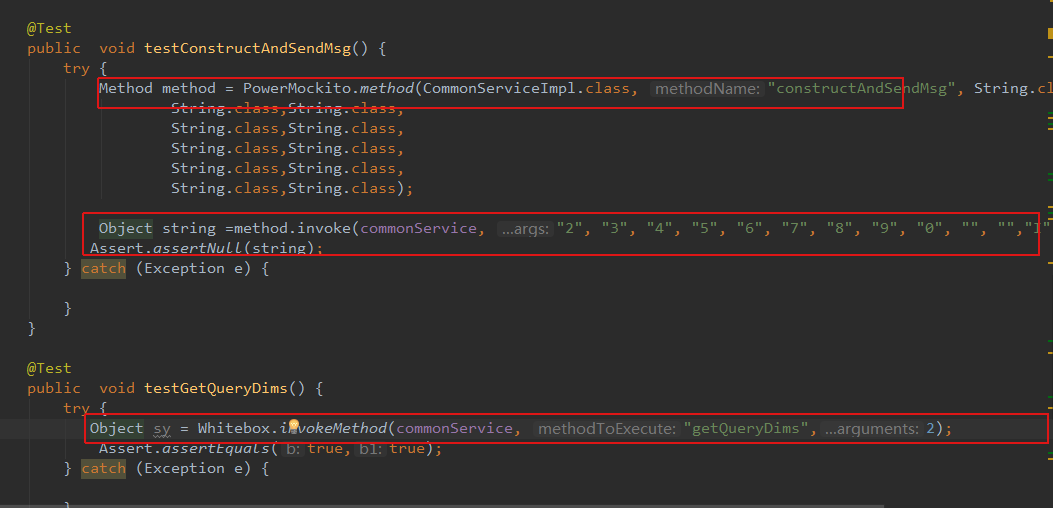
案例三（私有方法的测试）：

private ResultInfo catalogTypeInsertOrUpdate(int operType, String json, HttpServletRequest request)

mock私有方法返回：

PowerMockito.*verifyPrivate*(catalogController).invoke("catalogTypeInsertOrUpdate", 1, "json", request);

测试私有方法：\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*很重要！！！！！！！！！！！！！！！！！！！\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



案例四（设置私有变量对象的属性，Whitebox）：

public class DB2Connector extends Connector  
{  
 private String jdbcUrl;  
 private DruidDataSource druidDataSource;

}

测试

IPhysicalTableService ss = PowerMockito.*mock*(IPhysicalTableService.class);  
Whitebox.*setInternalState*(db2Connector, "DATA\_TABLE\_SERVICE", ss);

案例五（方法中包含参数对象无法mock，如局部变量时用isA）

public class{

public DataParamBean queryParamConverted{

DataParamBean dataParamBean = new DataParamBean();

dataDownloadService.queryParamConverted（dataParamBean）

}

}

测试方法中：

PowerMockito.*when*(dataDownloadService.queryParamConverted(Mockito.*isA*(DataParamBean.class))).thenReturn(dataParamBean);

案例六（期待抛出指定的异常）

@Test(expectedExceptions = { AssertionError.class })  
public void signature() {}

案例七 (静态方法返回的值无法构建，需要mock出来)

PowerMockito.*mockStatic*(ConnectorFactory.class);  
DataSource dataSource = PowerMockito.*mock*(DataSource.class);  
*when*(ConnectorFactory.*getDataSource*(*anyString*())).thenReturn(dataSource)

其他常见使用when().thenReturn();

When().thenThrow(); //抛出异常

常用anyString()，eq(),anyList(),anyMap(),anySet()等；

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注意点：

1.优先级 eq—>isA,anyList,anyMap,anyObject()-🡪any();

2.如果一个测试方法中有多个相同或者类似的，尽量少用通用的any，用eq，因为使用any通用很多地方会有莫名的测试失败；

3.when方法也尽量不要放在beforeclass注释的方法中，因为这样when对测试类所有的方法都有效，也最容易出错。

4.测试方法尽量用Asserts.assertEquals等方法结束。