**@RunWith(PowerMockRunner.class)**

PowerMock allows you to test code that can't be tested with other mocking frameworks: it takes your production classes and **manipulates** those in order to allow mocking **static** methods or **final** classes/methods.

**@PrepareForTest({ xxxxxxx.class,xxxxxx.class})**

This annotation tells PowerMock to prepare certain classes for testing.

**@Mock and @InjectMocks**

@Mock creates a mock. @InjectMocks creates an instance of the class and injects the mocks that are created with the @Mock (or @Spy) annotations into this instance. Note that you must use @RunWith(PowerMockRunner.class) or MockitoAnnotations.initMocks() to initialise these mocks and inject them.

**@Before**

When writing tests, it is common to find that several tests need similar objects created before they can run.

@Before causes that method to be run before the [Test](http://junit.sourceforge.net/javadoc/org/junit/Test.html) method.

**@Test**

**The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded**

**MockitoAnnotations.initMocks()**

To initlaise mock objects

**PrivateAccessor.setField()**

Utility class to bypass the Java modifiers security and access protected and private fields and methods.

**assertEquals()**

A set of assertion methods useful for writing tests. Only failed assertions are recorded. These methods can be used directly: Assert.assertEquals(...),

**Mocking CQ classes :**

1)  Node pagecontent = *mock*(Node.**class**);

*when*(pagecontent.hasProperty("cq:tags")).thenReturn(**true**);

 2) Property property =  *mock*(Property.**class**);

*when*(pagecontent.getProperty("cq:tags")).thenReturn(   property);

  3) Value value = *mock*(Value.**class**);

*when*(value.getString()).thenReturn("/etc/tags/DocTypes/HOME/Products and  Services Area Root/Products IOS Cisco IOS Software Category Home/Products IOS Sub-Category Home/Products IOS Technology Home/Products IOS Protocol Group Home/Products IOS Protocol Option Home"

        Value value1 = *mock*(Value.**class**);

*when*(value1.getString()).thenReturn("/etc/tags/Products/Cisco Products/Cisco IOS and NX-OS Software/Cisco IOS Technologies/Cisco IOS Security/Cisco IOS IPsec/V3PN");

         Value[] tagsArray = **new** Value[]{value,value1};

*when*(pagecontent.getProperty("cq:tags").getValues()).thenReturn(tagsArray);

4) *Tag tag = mock(Tag.class);*

*when(tagManager.resolve("/etc/tags/DocTypes/HOME/Products and Services Area Root/Products IOS Cisco IOS Software Category Home/Products IOS Sub-Category Home/Products IOS Technology Home/Products IOS Protocol Group Home/Products IOS Protocol Option Home")).thenReturn(tag);*

**Challenges  faced and solutions implemented**

|  |  |  |
| --- | --- | --- |
| **Description** | **Solution** | **Remarks** |
| Junit code for logger statements | PowerMockito.mockStatic(LoggerFactory.class); when(LoggerFactory.getLogger(GoodbyeController.class)).thenReturn(loggerMock); when(loggerMock.isDebugEnabled()).thenReturn(true); |  |
| When a new instance is getting created with in the method using 'new' operator then use the following | **Syntax:** ClassA ca = Mockito.mock(ClassA.class); PowerMockito.whenNew(ClassA.class).withArguments(argumentsifany).thenReturn(ca);  **Example:** ObjectName ldapSyncObjectName = Mockito.mock(ObjectName.class); PowerMockito.whenNew(ObjectName.class).withArguments(LDAPSyncConstants.LDAP\_SYNC\_MBEAN\_DOMAINNAME).thenReturn(ldapSyncObjectName); |  |
| To avoid traversing methods in other bundles, mock the bundle reference | @Mock private RenditionCommonUtil rendcomUtil;  In setUp() : rendcomUtil = PowerMockito.spy(new RenditionCommonUtil());  PrivateAccessor.setField(eotRenditionPageCreation, "rendcomUtil", rendcomUtil); PowerMockito.doReturn(session).when(rendcomUtil, "getUserSession",session, workItem); |  |
| Need to access the out of  box methods(with & without returnType) | **With returnType:**  **Syntax**: PowerMockito.doReturn(ObjectReturnedByTheMethod).when(Class classmockreference, String methodToBeExecuted, Object parametersOfMethod);  **Example:** PowerMockito.doReturn(asset).when(renitionPageCreation, "getAssetFromPayload", workItem, jcrSession);    **Without returnType:**  **Syntax:** PowerMockito.doNothing().when(Class classmock, String methodToBeExecuted, Object parametersOfMethod); |  |
| When a new instance is getting created with in the method using 'new' operator without arguments | **Syntax:**  ClassA ca= mock(ClassA.class); PowerMockito.whenNew(ClassA.class).withNoArguments().thenReturn(ca);  Add the **ClassA.class**in @PrepareForTest  **Example:** ObjectName ldapSyncObjectName = Mockito.mock(ObjectName.class); PowerMockito.whenNew(ObjectName.class).withNoArguments().thenReturn(ldapSyncObjectName); |  |
| when to mock and when to spy | 1. If you are creating a complete mock or fake object , then use mock while in spy, there is the real object and you just spying or stubbing specific methods of it. 2. If you want to be safe and avoid calling external services and just want to test the logic inside of the unit, then use mock 3. If you want to run the program as it is and just stub specific methods, then use spy. |  |
| How to mock method call and return value without running the method | use  Mockito.doReturn(true).when(spy).Iscompatible();(here i'm calling Iscompatible() method and returning true not going inside the method)  You can use doThrow(), doAnswer(), doNothing(), doReturn() and doCallRealMethod() in place of the corresponding call with when(), for any method. It is necessary when you  I .stub void methods II .stub methods on spy objects III.stub the same method more than once, to change the behaviour of a mock in the middle of a test.  **Syntax -** PowerMockito.doReturn(ObjectReturnedByTheMethod).when(Class classmockreference, String methodToBeExecuted, Object parametersOfMethod);  **Example:**  PowerMockito.doReturn(asset).when(renitionPageCreation, "getAssetFromPayload", workItem, jcrSession); |  |
| To Use an object of other class twice , once as real , once as mock | use only one object.Use spy insted of mock , it is called partial mock. you can mock only specific methods(as mock) ,and remaining works as normal methods.  please refer <http://www.baeldung.com/mockito-spy> to know how to implement spy. |  |
| How to mock static methods | there are 4 steps involved 1.Use the PowerMock JUnit runner: @RunWith(PowerMockRunner.class) 2.Declare the test class that we’re mocking: @PrepareForTest(UsesResourceBundle.class) 3.Tell PowerMock the name of the class that contains static methods: mockStatic(ResourceBundle.class); 4.Setup the expectations, telling PowerMock to expect a call to a static method: expect(ResourceBundle.getBundle("SomeBundleName", Locale.ENGLISH)).andReturn(bundle);  **Example:**  @RunWith(PowerMockRunner.class)  @PrepareForTest({AuthoringUitilityService.class,**JcrResourceUtil.class**,UUID.class})   PowerMockito.mockStatic(JcrResourceUtil.class);  when(JcrResourceUtil.createPath("/content/4", null, DamConstants.NT\_SLING\_ORDEREDFOLDER, session, true)).thenReturn(subAuditNode); |  |
| Mocking the out of box methods(with & without returnType) | **Method I :**  With returnType: Syntax - PowerMockito.doReturn(ObjectReturnedByTheMethod).when(Class classmockreference, String methodToBeExecuted, Object parametersOfMethod); Example for "firstClass.getFirstMethod(arg1, arg2)" whose returnType is 'SecondClass' object is below →PowerMockito.doReturn(secondClass).when(firstClass, "getFirstMethod", arg1, arg2);  Without returnType: Syntax - PowerMockito.doNothing().when(Class classmockreference, String methodToBeExecuted, Object parametersOfMethod); Example for "firstClass.getFirstMethod(arg1, arg2)" whose returnType is nothing is below →PowerMockito.doNothing().when(firstClass, "getFirstMethod", arg1, arg2); |  |
| Mocking the out of box methods(with & without returnType) | **Method II(When Method I doesn't works) :**  With returnType: Syntax - PowerMockito.doReturn(ObjectReturnedByTheMethod).when(classmockreference).methodToBeExecuted(parametersOfMethod); Example for "firstClass.getFirstMethod(arg1, arg2)" whose returnType is 'SecondClass' object is below →PowerMockito.doReturn(secondClass).when(firstClass).getFirstMethod(arg1, arg2);  Without returnType: Syntax - PowerMockito.doNothing().when(Class classmockreference, String methodToBeExecuted, Object parametersOfMethod); Example for "firstClass.getFirstMethod(arg1, arg2)" whose returnType is nothing is below →PowerMockito.doNothing().when(firstClass).getFirstMethod(arg1, arg2); | The way of calling and handling the statement is different in Method II |
| Mocking the private method | Syntax : Whitebox.invokeMethod(Class classreference, String methodToExecute, Object... arguments); |  |
| When we get dynamic argument for a method, such as creating a audit node logic uses current date and time. such cases can be mocked with Matchers.any() | eg:  folderNode = JcrResourceUtil.createPath(auditNodePath.toString(), DamConstants.NT\_SLING\_ORDEREDFOLDER, DamConstants.NT\_SLING\_ORDEREDFOLDER, session, true); //audit node path uses calender object to create audit node for present day.  This can be mocked as:  when(JcrResourceUtil.createPath(**Matchers.anyString()**, Matchers.eq(DamConstants.NT\_SLING\_ORDEREDFOLDER), Matchers.eq(DamConstants.NT\_SLING\_ORDEREDFOLDER), Matchers.eq(session), Matchers.eq(true))).thenReturn(auditParentNode);  however, the other arguments should be prefixed with **Matchers.eq()**method. |  |
| Mocking the exceptions on methods | 1.First we need to mock that particular exception as below →ParticularException pe = mock(ParticularException.class);  2.Then we can mock or stub the above exception as below, see with example Example for "firstClass.getFirstMethod(arg1, arg2)" which throws 'ParticularException' exception is below, →PowerMockito.doThrow(pe).when(firstClass, "getFirstMethod", arg1, arg2); OR →when(firstClass.getFirstMethod(arg1, arg2)).thenThrow(pe); |  |
| For 'TooManyConstructorsFoundException' | Syntax: RequiredClass requiredClass = mock(RequiredClass.class); PowerMockito.whenNew(RequiredClass.class).withParameterTypes(ParameterType1.class, ParameterType2.class).withArguments(parameterType1, parameterType2).thenReturn(requiredClass);  Example of sample code: PowerMockito.whenNew(ResponseEntity.class).withParameterTypes(MultiValueMap.class,HttpStatus.class).withArguments(headers, statusCode).thenReturn(responseEntity); |  |