Mocking in Unit Tests

Jenkins provides excellent support for integration tests by starting up a complete instance of Jenkins if your test class extends `HudsonTestCase`. However, sometimes you might not want to do this - e.g. because you just want to test a single method or just because launching a HudsonTestCase takes a long time (several seconds).

Mocking with Mockito

Unfortunately, it's not easy to test most Jenkins core classes in isolation as they implicitely depend on the `Jenkins#getInstance()` singleton, so creating/accessing instances of them often results in `NullPointerExceptions`. One technique which comes in handy in this case is stubbing / *mocking*. There are several good mocking frameworks for Java, but we propose Mockito - it's already included in Jenkins' core dependencies. A basic example looks like this:

```
AbstractBuild build = Mockito.mock(AbstractBuild.class);
Mockito.when(build.getResult()).thenReturn(Result.FAILURE);
```

This creates a new instance of AbstractBuild which returns FAILURE as build result. All methods which are not explicitly instrumented via one of the Mockito *when*, *doCallRealMethod*, *doThrow* (and so on) methods will do nothing (and return null if they have a return value). Note that there are several other options on how to *verify* that certain methods are called in your test case or to *spy* on real objects. You should read the Mockito documentation to get familiar with these features.

More complete examples can be found in existing Jenkins unit test. See for example: SurefireArchiverUnitTest

Handle more complicated cases with PowerMock

In a more complicated case this may still not be enough to test a Jenkins class in isolation. Possible reasons are static or final methods/classes (Mockito can't mock these) or access to Jenkins.getInstance() in constructors. In that case you can use PowerMock/PowerMockito to work around these problems - i.e. removing the 'final' restriction of Mockito or suppressing constructors. See MavenModuleTest for an example how to use it.

Here's an example using PowerMock to mock the `Jenkins#getInstance()` singleton from the Github OAuth Plugin.

```
package org.jenkinsci.plugins;

import jenkins.model.Jenkins;
import junit.framework.TestCase;
import org.jenkinsci.plugins.GithubSecurityRealm.DescriptorImpl;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.powermock.api.mockito.PowerMockito;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;

@RunWith(PowerMockRunner.class)
@PrepareForTest({Jenkins.class, GithubSecurityRealm.class, DescriptorImpl.class})
public class GithubLogoutActionTest extends TestCase {

    @Mock
    private Jenkins jenkins;

    @Mock
    private GithubSecurityRealm securityRealm;

    @Mock
    private DescriptorImpl descriptor;

    @Before
    public void setUp() throws Exception {
        PowerMockito.mockStatic(Jenkins.class);
        PowerMockito.when(Jenkins.getInstance()).thenReturn(jenkins);
        PowerMockito.when(jenkins.getSecurityRealm()).thenReturn(securityRealm);
        PowerMockito.when(securityRealm.getDescriptor()).thenReturn(descriptor);
        PowerMockito.when(descriptor.getDefaultGithubWebUri()).thenReturn("https://github.com");
    }

    private void mockGithubSecurityRealmWebUriFor(String host) {
        PowerMockito.when(securityRealm.getGithubWebUri()).thenReturn(host);
    }

    @Test
    public void testGetGitHubText_gh() {
        mockGithubSecurityRealmWebUriFor("https://github.com");
        GithubLogoutAction ghlogout = new GithubLogoutAction();
        assertEquals("GitHub", ghlogout.getGitHubText());
    }

    @Test
    public void testGetGitHubText_ghe() {
        mockGithubSecurityRealmWebUriFor("https://ghe.example.com");
        GithubLogoutAction ghlogout = new GithubLogoutAction();
        assertEquals("GitHub Enterprise", ghlogout.getGitHubText());
    }
}
```

In the above example, the `GithubLogoutAction` class is being tested. It calls `Jenkins.getInstance().getSecurityRealm()` (which returns the mocked `securityRealm`).

> ⚠ PowerMocking the static `Jenkins` class appears to negatively interfere with `org.jvnet.hudson.test.HudsonTestCase`. When using `HudsonTestCase` in the same code above an unhelpful exception is thrown: `java.lang.ExceptionInInitializerError: null`.

Troubleshooting

java.lang.ClassCastException: com.sun.crypto.provider.AESCipher cannot be cast to javax.crypto.CipherSpi

In case you see CCE happening with similar messages, you may want to prevent the mocking library of changing things in the crypto land.

```
Error injecting constructor, java.lang.ClassCastException: com.sun.crypto.provider.AESCipher cannot be cast to
javax.crypto.CipherSpi
  at hudson.security.csrf.DefaultCrumbIssuer$DescriptorImpl.<init>(DefaultCrumbIssuer.java:127)
```

You can disable that in Powermockito with the following annotation: PowerMockIgnore. Here's one example.

```
// snip
@RunWith(PowerMockRunner.class)
@PrepareForTest({StaplerRequest.class})
@PowerMockIgnore({"javax.crypto.*" })
public class TestAbstractUnoChoiceParameter {
//
}
// snip
```

PermGen errors with Java 7 when using PowerMockito

When running your plug-in tests with Java 7 and PowerMockito, you may see PermGen errors. You can increase the PermGen size if you have access to the JVM settings, but sometimes you are not able to change that. In this case, sometimes this error happens when PowerMockito creates several objects in the PermGen, that are never collected. This can be avoided if you add the PrepareForTest annotation to your test.

```
// snip
@RunWith(PowerMockRunner.class)
@PrepareForTest({StaplerRequest.class})
@PowerMockIgnore({"javax.crypto.*" })
public class TestAbstractUnoChoiceParameter {
//
}
// snip
```

The class above was not annotated with the PrepareFortest, but had the PowerMockIgnore and RunWith, and there were PermGen errors. After adding it, PowerMockito takes care to carefully create instances as required for that class, keeping a pool and avoiding pollution in the PermGen.