ABOUT ROGER HUGHES

Mock Static Methods with PowerMock

Posted by: Roger Hughes in Core Java November 4th, 2011 0 40 Views (*0 rating, 0 votes*)

In a recent blog, I tried to highlight the benefits of using depende
expressing the idea that one of the main benefits of this techniqu
to test your code more easily by providing a high degree of isolati
and coming to the conclusion that lots of good tests equals good

But, what happens when you don't have dependency injection an
party library that contains classes of a certain vintage that contai
One way is to isolate those classes by writing a wrapper or adapt
using this to provide isolation during testing; however, there's als
PowerMock. PowerMock is a mocking framework that extends oth
frameworks to provide much needed additional functionality. To p
advert: "it refreshes the parts that other mocking frameworks fail

This blog takes a look at PowerMock's ability to mock static meth
example of mocking the JDK's ResourceBundle class, which as many of you know uses ResourceBundle.getBundle(...) to,
bundles.

I, like many other bloggers and writers, usually present some highly contrived scenario to highlight the problem. Today is
got a class that uses a ResourceBundle called: UsesResourceBundle:

```
public class UsesResourceBundle {

  private static Logger logger = LoggerFactory.getLogger(UsesResourceBundle.class);
```

```

    if (isNull(bundle)) {
      // Lazy load of the resource bundle
      Locale locale = getLocale();

      if (isNotNull(locale)) {
        this.bundle = ResourceBundle.getBundle("SomeBundleName", locale);
      } else {
        handleError();
      }
    }

    return bundle.getString(key);
  }

  private boolean isNull(Object obj) {
    return obj == null;
  }

  private Locale getLocale() {

    return Locale.ENGLISH;
  }

  private boolean isNotNull(Object obj) {
    return obj != null;
  }

  private void handleError() {
    String msg = "Failed to retrieve the locale for this page";
    logger.error(msg);
    throw new RuntimeException(msg);
  }
}
```

You can see that there's one method: getResourceString(...), which given a key will retrieve a resource string from a bunc
this work a little more efficiently, I've lazily loaded my resource bundle, and once loaded, I call bundle.getString(key) to re
To test this I've written a PowerMock JUnit test:

```
import static org.easymock.EasyMock.expect;
import static org.junit.Assert.assertEquals;
import static org.powermock.api.easymock.PowerMock.mockStatic;
import static org.powermock.api.easymock.PowerMock.replayAll;
import static org.powermock.api.easymock.PowerMock.verifyAll;

import java.util.Locale;
import java.util.MissingResourceException;
import java.util.ResourceBundle;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.powermock.api.easymock.annotation.Mock;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;

@RunWith(PowerMockRunner.class)
@PrepareForTest(UsesResourceBundle.class)
public class UsesResourceBundleTest {

  @Mock
  private ResourceBundle bundle;

  private UsesResourceBundle instance;

  @Before
  public void setUp() {
    instance = new UsesResourceBundle();
  }

  @Test
```

```

    final String key = "DUMMY";
    final String message = "This is a Message";
    expect(bundle.getString(key)).andReturn(message);

    replayAll();
    String result = instance.getResourceString(key);
    verifyAll();
    assertEquals(message, result);
  }

  @Test(expected = MissingResourceException.class)
  public final void testGetResourceStringWithStringMissing() {

    mockStatic(ResourceBundle.class);
    expect(ResourceBundle.getBundle("SomeBundleName", Locale.ENGLISH)).andReturn(bu

    final String key = "DUMMY";
    Exception e = new MissingResourceException(key, key, key);
    expect(bundle.getString(key)).andThrow(e);

    replayAll();
    instance.getResourceString(key);
  }

  @Test(expected = MissingResourceException.class)
  public final void testGetResourceStringWithBundleMissing() {

    mockStatic(ResourceBundle.class);
    final String key = "DUMMY";
    Exception e = new MissingResourceException(key, key, key);
    expect(ResourceBundle.getBundle("SomeBundleName", Locale.ENGLISH)).andThrow(e);

    replayAll();
    instance.getResourceString(key);
  }

}
```

In the code above I've taken the unusual step of including the import statements. This is to highlight that we're using Pow
the import statics and not EasyMock's. If you accidentally import EasyMock's statics, then the whole thing just won't work

There are four easy steps in setting up a test that mocks a static call:

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
```

The rest is plain sailing, you set up expectations for other standard method calls and the tell PowerMock/EasyMock to run
the results:

```
final String key = "DUMMY";
final String message = "This is a Message";
```

```
verifyAll();
```

PowerMock can do lots more, such as mocking constructors and private method calls. More on that later perhaps…

Reference: Using PowerMock to Mock Static Methods from our JCG partner Roger at Captain Debug's Blog.

Related Articles :

- Rules in JUnit 4.9 (beta 3)
- Testing an Object's Internal State with PowerMock
- Servlet 3.0 Async Processing for Tenfold Increase in Server Throughput
- Testing with Scala
- Java Tools: Source Code Optimization and Analysis
- Java Tutorials and Android Tutorials list

Tagged with: JUNIT MOCK POWERMOCK

(*0 rating, 0 votes*)

You need to be a registered member to rate this. Start the discussion 40 Views Tweet it!