ABOUT ROGER HUGHES

Testing an Object's Internal State with PowerMock

Posted by: Roger Hughes in Core Java October 24th, 2011

Most unit testing focuses on testing an object's behaviour in order to prove that it works. This is achieved by writing a JUnit test that calls an object's public methods and then testing that the return values from these calls match some previously defined set of expected values. This is a very common and successful technique; however, it shouldn't be forgotten that objects also exhibit state; something that is, by virtue of the fact that it's hidden, often overlooked.

Grady Booch's 1994 Book *Object Oriented Analysis and Design*, which I first read in the summer of 1995 defines an object's state in the following way:

> *The state of an object encompasses all of the (usually static) properties of the object plus the current (usually dynamic) values of each of these properties.*

He defines the difference between static state and dynamic state using a vending machine example. Static state is exhibited by the way that the machine is always ready to take your money, whilst dynamic state is how much of your money it's got at any given instance.

I suspect that at this point, you'll quite rightly argue that explicit behavioural tests do test an object's state by virtue of the fact that a given method call returned the correct result and that to get the correct result the object's state had to also be correct… and I'll agree. There are, however, those very few cases where classic behavioural testing isn't applicable. This occurs when a public method call has no output and does nothing to an object except change its state. An example of this would be a method that returned void or a constructor. For example, given a method with the following signature:

```
1 public void init();
```

…how do you ensure it's done its job? It turns out that there are several methods you can use to achieve this…

- Add lots of getter methods to your class. This is not a particularly good idea, as you're simply loosening encapsulation by the back door.
- Relax encapsulation: make private instance variables package private. A very debatable thing to do. You could pragmatically argue that having well tested, correct and reliable code may be better than having a high degree of encapsulation, but I'm not too sure here. This may be a short term fix, but could lead to all kinds of problems in the future and there should be a way of writing well tested, correct and reliable code that doesn't include breaking an object's encapsulation
- Write some code that uses reflection to access an object's internal state. This is the best idea to date. The down side is that it's a fair amount of effort and requires a reasonable amount of programming competence.
- Use PowerMock's Whitebox testing class to do the hard work for you.

The following fully contrived scenario demonstrates the use of PowerMock's Whitebox class. It takes a very simple AnchorTag <a> class that will build an anchor tag after testing that an input URL string is valid.

```
01 public class AnchorTag {
02
03   private static final Logger logger = LoggerFactory.getLogger(AnchorTag.class);
04
05   /** Use the regex to figure out if the argument is a URL */
```

NEWSLETTER

179,260 insiders are already er
weekly updates and complimentary
whitepapers!
Join them now to gain ex
access to the latest news in the J
as well as insights about Android, Sc
Groovy and other related technologi

Email address:

Your email address

Sign up

JOIN US

With 1,240,60
unique visitors
500 authors
placed among
related sites ar
Constantly bei
lookout for par
encourage you
So If you have
unique and interesting content then yo

```
  private final Pattern pattern = Pattern.compile("^([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?\\.)+[a-zA-Z]{2,6}$");

  /**
   * A public method that uses the private method
   */
  public String getTag(String url, String description) {

    validate(url, description);
    String anchor = createNewTag(url, description);

    logger.info("This is the new tag: " + anchor);
    return "The tag is okay";
  }

  /**
   * A private method that's used internally, but is complex enough to require testing in its own right
   */
  private void validate(String url, String description) {

    Matcher m = pattern.matcher(url);

    if (!m.matches()) {
      throw new IllegalArgumentException();
    }
  }

  private String createNewTag(String url, String description) {
    return "<a href=\"" + url + "\">" + description + "</a>";
  }
}
```

The URL validation test is done using a regular expression and a Java Pattern object. Using the Whitebox class will ensure that the pattern object is configured correctly and that our AnchorTag is in the correct state. This demonstrated by the JUnit test below:

```
/**
   * Works for private instance vars. Does not work for static vars.
   */
  @Test
  public void accessPrivateInstanceVarTest() throws Exception {

    Pattern result = Whitebox.<pattern> getInternalState(instance, "pattern");

    logger.info("Broke encapsulation to get hold of state: " + result.pattern());
    assertEquals("^([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?\\.)+[a-zA-Z]{2,6}$", result.pattern());
  }
```

The crux of this test is the line:

```
Pattern result = Whitebox.<pattern> getInternalState(instance, "pattern");
```

...which uses reflection to return the Pattern object private instance variable. Once we have access to this object, we simply ask it if it has been initialised correctly be calling:

```
assertEquals("^([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?\\.)+[a-zA-Z]{2,6}$", result.pattern());
```

In conclusion I would suggest that using PowerMock to explicitly test an object's internal state should be used only when you can't use straight forward classic JUnit test for behavioural testing. Having said that, it is another tool in your toolbox that'll help you to write better code.

Reference: Testing an Object's Internal State with PowerMock from our JCG partner Roger at Captain Debug's Blog.

Related Articles :

- Rules in JUnit 4.9 (beta 3)
- 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 POWERMOCK

Do you want to know how to develop your skillset to become a Java Rockstar?

Subscribe to our newsletter to start Rocking right now!
To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions

check out our **JCG** partners program. \
be a **guest writer** for Java Code Geek
your writing skills!

CAREER OPPORTUNITIES

Job Title
developer
Location
City, state, or zip
Search

within 100 miles within 30 da

No jobs found. Check
popular search terms
again:

Account Manager Jobs
Customer Service Jobs
Management Jobs
Sales Associate Jobs
Software Engineer Jobs

Job Search by ZipRecr