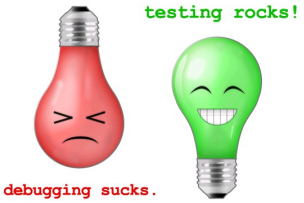

Testing on the Toilet

Sleeping != Synchronization

August 21, 2008

You've got some code that uses threads, and it's making your tests flaky and slow. How do you fix it? First, most of the code is probably still single-threaded: test those parts separately. But how to test the **threading behavior** itself?

Often, threaded tests start out using sleeps to wait for something to happen. This test is trying to verify that `DelegateToIntern` spawns its work argument into a parallel thread and invokes a callback when it's done.

```
def testInternMakesCoffee(self):

  self.caffeinated = False
  def DrinkCoffee(): self.caffeinated = True

  DelegateToIntern(work=Intern().MakeCoffee, callback=DrinkCoffee)

  self.assertFalse(self.caffeinated, "I ~~watch YouTube~~work; intern brews")
  time.sleep(60)  # 1min should be long enough to make coffee, right?
  self.assertTrue(self.caffeinated, "Where's mah coffee?!?")
```

Aside from abusing your intern every time you run the test, this test takes a minute longer than it needs to, and it may even fail when the machine (or intern!) is loaded in odd ways. You should always **be skeptical of sleep statements**, especially in tests. How can we make the test more reliable?

The answer is to **explicitly control** when things happen within `DelegateToIntern` with a **`threading.Event`** in Python, a **`Notification`** in C++, or a **`CountDownLatch(1)`** in Java.

```
def testInternMakesCoffee(self):

  is_started, can_finish, is_done = Event(), Event(), Event()

  def FakeCoffeeMaker():
    is_started.set()  # Allow is_started.wait() to return.
    # Wait up to 1min for can_finish.set() to be called. The timeout
    # prevents failures from hanging, but doesn't delay a passing test.
    can_finish.wait(timeout=60)  # .await() in Java

  DelegateToIntern(work=FakeCoffeeMaker, callback=lambda:is_done.set())

  is_started.wait(timeout=60)
  self.assertTrue(is_started.isSet(), "FakeCoffeeMaker should have started")
  self.assertFalse(is_done.isSet(), "Don't bug me before coffee's made")

  can_finish.set()  # Now let FakeCoffeeMaker return.
  is_done.wait(timeout=60)
  self.assertTrue(is_done.isSet(), "Intern should ping when coffee's ready")
```

Now we're guaranteed that no part of the test runs faster than we expect, and the test passes **very quickly**. It could run slowly when it fails, but you can easily lower the timeouts while you're debugging it.

We'll look at testing for race conditions in a future episode.

No interns were harmed in the making of this TotT.

More information, discussion, and archives:
http://googletesting.blogspot.com