Debugging sucks. Testing rocks.

Testing on the Toilet

Exercise Service Call Contracts in Tests

The following test mocks out a service call to `CloudService`. **Does the test provide enough confidence that the service call is likely to work?**

```
@Test public void uploadFileToCloudStorage() {
  when(mockCloudService.write(
          WriteRequest.newBuilder().setUserId(“testuser”).setFileType(“plain/text”)...))
    .thenReturn(WriteResponse.newBuilder().setUploadId(“uploadId”).build());

  CloudUploader cloudUploader = new CloudUploader(mockCloudService);

  Uri uri = cloudUploader.uploadFile(new File(“/path/to/foo.txt”));
  // The uploaded file URI contains the user ID, file type, and upload ID. (Or does it?)
  assertThat(uri).isEqualTo(new Uri(“/testuser/text/uploadId.txt”));
```

Lots of things can go wrong, especially when service contracts get complex. For example, `plain/text` may not be a valid file type, and you can’t verify that the URI of the uploaded file is correct.

If the code under test relies on the contract of a service, prefer exercising the service call instead of mocking it out. This gives you more confidence that you are using the service correctly:

```
@Test public void uploadFileToCloudStorage() {
  CloudUploader cloudUploader = new CloudUploader(cloudService);

  Uri uri = cloudUploader.uploadFile(”/path/to/foo.txt”);
  assertThat(cloudService.retrieveFile(uri)).isEqualTo(readContent(“/path/to/foo.txt));
}
```

How can you exercise the service call?

1. **Use a fake**. A fake is a fast and lightweight implementation of the service that behaves just like the real implementation. A fake is usually maintained by the service owners; don’t create your own fake unless you can ensure its behavior will stay in sync with the real implementation.
   Learn more about fakes at testing.googleblog.com/2013/06/testing-on-toilet-fake-your-way-to.html.

2. **Use a hermetic server**. This is a real server that is brought up by the test and runs on the same machine that the test is running on. A downside of using a hermetic server is that starting it up and interacting with it can slow down tests.
   Learn more about hermetic servers at testing.googleblog.com/2012/10/hermetic-servers.html.

If the service you are using doesn’t have a fake or hermetic server, mocks may be the only tool at your disposal. But **if your tests are not exercising the service call contract, you must take extra care to ensure the service call works**, such as by having a comprehensive suite of end-to-end tests or resorting to manual QA (which can be inefficient and hard to scale)**More information, discussion, and archives:**

testing.googleblog.com