The Bug: One Static Driver, Two Tests

A static field is one box for the whole program, not one per test. Two parallel tests writing to it corrupt each other with no error, no exception, and a symptom that looks exactly like flakiness.

Automation Foundations · ashishautomationlabs.store

A **thread** is a single sequence of instructions being executed. Everything you have run so far used one thread: `main` started, ran your code line by line, and finished. Running tests in parallel means running several threads at once, each executing a different test method. The important consequence is this: threads share the memory of the program. Two threads running two tests can see and modify the very same variable, and neither of them knows the other exists. That is fine for anything each thread creates for itself. It is a serious problem for anything they share.

Here is the defect, in the smallest form that still shows it. A static field holds the driver, exactly as many first frameworks do, and two tests run at once:

```
public class TH1 {
    static String driver = "none";
    static void runTest(String name, String session) {
        driver = session;
        System.out.println(name + " set driver to   : " + session);
        sleep(150);
        System.out.println(name + " now sees driver : " + driver);
    }
}
```

```
TestA set driver to   : chrome-session-1
TestB set driver to   : chrome-session-2
TestA now sees driver : chrome-session-2
TestB now sees driver : chrome-session-2
```

Read the third line slowly. TestA set the driver to `session-1`, and a moment later it is looking at `session-2`. Nothing crashed. No exception, no error. TestA is now driving TestB's browser -- clicking in the wrong window, reading the wrong page, and reporting a failure that has nothing to do with the application. TestB happens to be fine this time, because it wrote last.

The cause traces straight back to Chapter 8. A static field belongs to the class, and there is exactly one of it for the whole program. Not one per test, and not one per thread. Both threads wrote to the same box, and the second write overwrote the first. This explains the symptoms exactly. It only appears in parallel, because with one thread there is nobody to overwrite you. It moves around between runs, because which thread writes last depends on timing. And it looks like flakiness, because the tests really do pass in isolation. (Threads make no promise about ordering, so the exact interleaving of these lines can differ between runs -- the corruption itself is

the reliable part, and it is what matters.)

One static field caused all of that. The fix doesn't touch the rest of the framework at all -- it changes exactly one thing about how that field is declared.