www.froglogic.com - contact@froglogic.com

The 10 Pitfalls Of Automated JAVA GUI Testing

Reginald Stadlbauer

froglogic

Java Forum Stuttgart 2009

About me

www.froglogic.com - contact@froglogic.com

- Reginald Stadlbauer <reggie@froglogic.com>
- Co-founder and CEO of froglogic GmbH, Hamburg, Germany
- Specialized of automated GUI testing tool “Squish”
- Talk based on experience with 1000+ customers

Intro: Types of Testing

froglogic®
www.froglogic.com - contact@froglogic.com

- Unit Testing
- Performance Testing
- ...
- Functional GUI Testing
  - Black/Gray Box Testing
  - Assume user's point of view
  - Automate to spot regressions
  - Combinable with profiling tools

Intro: Why Automate?

froglogic®
www.froglogic.com - contact@froglogic.com

- Faster
  - Get results quicker
  - Run more tests in the same time
- Trivial to replay in different configurations
- Reliable, reproducible and repeatable
- Relieve testers from monotonous tasks

1. Rely on capture & replay

froglogic®
www.froglogic.com - contact@froglogic.com

- Produces massive test scripts
- Not readable
- Not maintainable
- No code re-use possible
- Brittle against changes in the UI

- Solution: Scripting & Refactoring

froglogic®
www.froglogic.com - contact@froglogic.com

2. Rely on screen coordinates

- Addresses screen positions and not UI controls
- Breaks with UI layout changes
- Depends on GUI style and platform
- Scripts hard to understand

- Solution: Address objects based on properties

3. Rely on screen captures / OCR

froglogic®
www.froglogic.com - contact@froglogic.com

- No knowledge of GUI controls
- Too much heuristics
- Depends on irrelevant data (colors, fonts, etc.)
- Many incorrect fails / errors

- Solution: Identify on and compare object properties

4. Rely on “Windows” test tools

froglogic®
www.froglogic.com - contact@froglogic.com

- Only “knows” standard Windows controls
- Cannot drill into Java controls
- Object identification based on limited amount of properties
- Not cross-platform

- Solution: Use a tool which understands Java controls

5. Use primitive macro language

froglogic®
www.froglogic.com - contact@froglogic.com

- Limited to small set of features
- No way to “break out”
- No way to utilize 3rd party libraries (database access, etc.)
- No way to deal with dynamic tests

- Solution: Use scripting solution for test automation

6. Tests embedded in application

froglogic®
www.froglogic.com - contact@froglogic.com

- Tempting to test API rather than GUI
- Application crash or freeze not handled well
- Can only test one application per test case
- Modifies application

- Solution: Run test in a separate process

froglogic®
www.froglogic.com - contact@froglogic.com

7. Rely on unique object IDs

- Burden for developers
- Not realistically doable if testing is introduced later
- Need uniqueness checking

- Solution: Use multi-property naming

8. Rely on AUT's object hierarchy

froglogic®
www.froglogic.com - contact@froglogic.com

- Long and unreadable names
- Relies on application internal “helper widgets”
- Small layout changes breaks naming

- Solution: Use multi-property naming

9. Create tests “on the side”

froglogic®
www.froglogic.com - contact@froglogic.com

- Development resources are already restricted
- There is always “one more important dev task”
- Easy to delay “until tomorrow”

- Solution: Dedicated resource for testing

10. Setup automation “when ready”

froglogic®
www.froglogic.com - contact@froglogic.com

- Nobody runs the tests and sees the fails/errors
- Tests will become unmaintained and not work anymore
- Tests will be forgotten

- Solution: First task: set up automation, then start creating tests

www.froglogic.com - contact@froglogic.com

More about Squish at www.froglogic.com

Get an evaluation at www.froglogic.com/evaluate

or visit our booth!

Thank you for your attention!