9. Introduction to test automation. Advantages and disadvantages.

We don’t do manual regression testing :)

Astea Solutions QA Team

Overview

- Why
- Software testing pyramid
- Tools
- Advices from experts
- But, wait! What about manual testing

Resources

- http://googletesting.blogspot.bg
- https://developers.google.com/google-test-automation-conference/
- http://www.guru99.com/selenium-tutorial.html
- http://www.toolsqa.com/selenium-webdriver/selenium-tutorial/

Why (1)

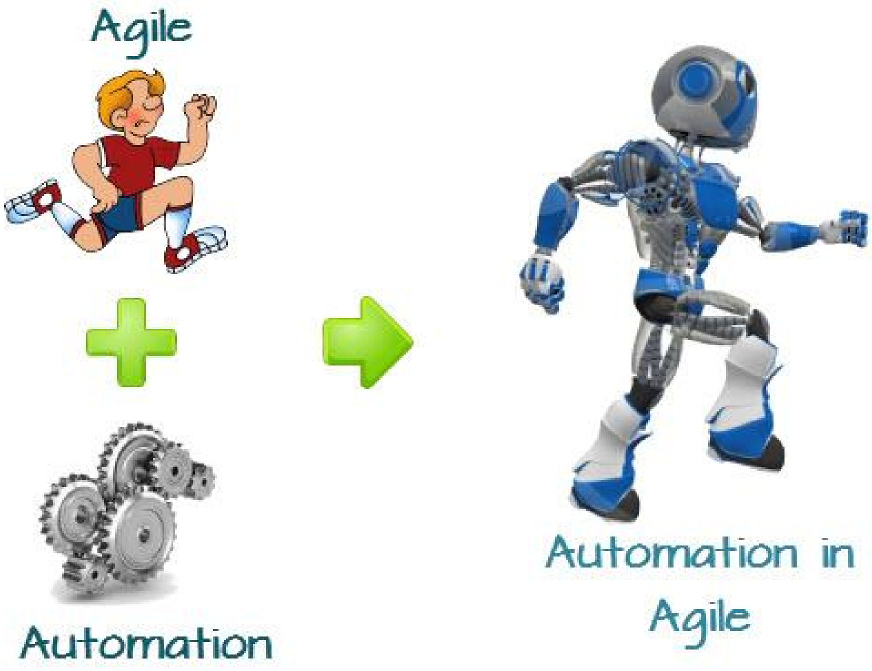
- Automated tests
  - allow to regress products thoroughly with each change
  - let QA specialists focus on risk assessment and test planning.
  - help respond to customers' needs quickly

Why (2)

- It’s all about Agile software development today

- Test automation and continuous integration are the cornerstone of Agile development

- Move Fast & Don’t Break Things



Why (3)

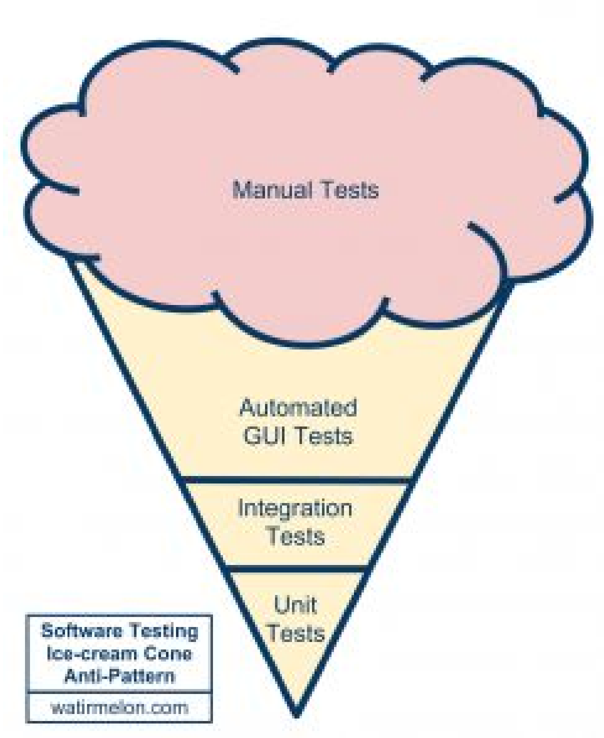
- Repetitive tasks like manual testing numb the mind and invite mistakes
- Testers slowly become less focused on their task, and more likely to let defects slip through
- But a computer isn't affected by this phenomenon. So automation ensures that each test is rigorously executed every time
- If testers are executing a set of test cases over and over again, we should automate them, unless in rare circumstances the ROI to automate them isn’t there

Why (4)

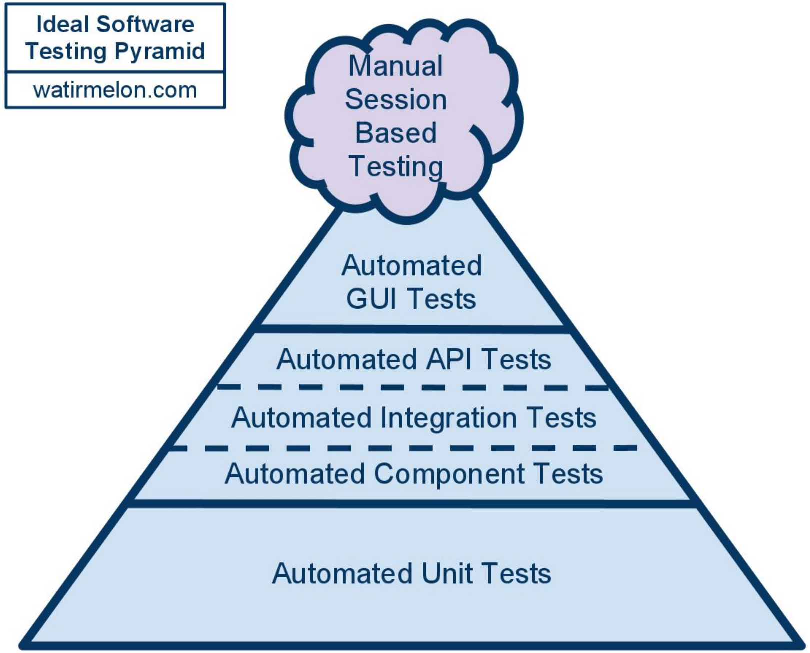
- OK! You convinced me. And what about small apps (projects)
  - If testers are executing a set of test cases over and over again, we should automate them, unless in rare circumstances the ROI to automate them isn’t there
  - Automate (just) enough to eliminate manual regression testing

Software testing pyramid (1)

What many teams do



How it should be done



Software testing pyramid (2)

What is wrong with automated e2e test?

- Scripting your manual tests this way takes far longer than just executing them manually
  - Well, let's automate only tests that are important, and will be executed again and again in regression testing
- The UI is one of the least stable interfaces of any system, so we can start automating quite late in the development phase
- Maintenance of the tests takes a significant amount of time
- Execution is slow, and sometimes cumbersome
  - Execution might be slow, but it is still faster than manual testing

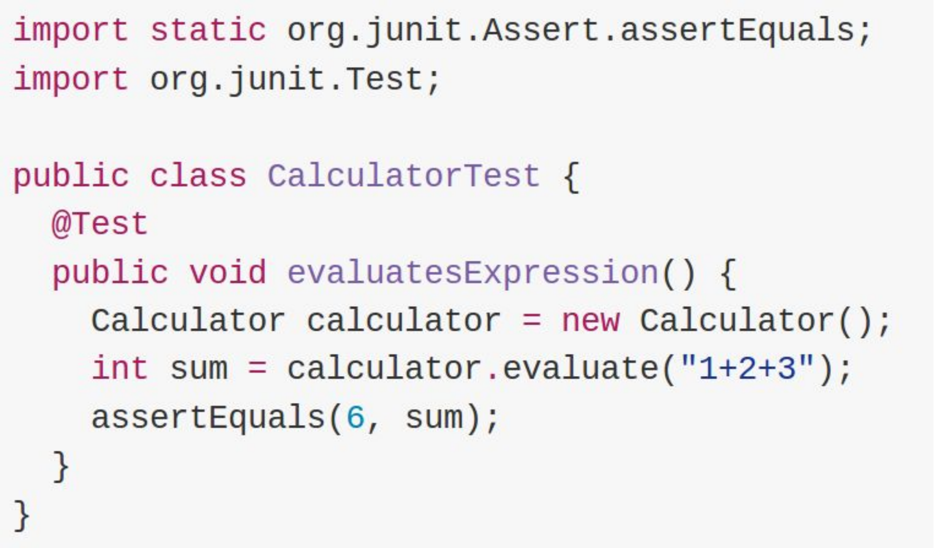
	Unit	End-toEnd
Fast	☺	☹
Reliable	☺	☹
Isolates Failures	☺	☹
Simulates a Real User	☹	☺

Software testing pyramid (3)

- How we can achieve software testing pyramid?
  - Is there a database? If so, the verification should probably not be performed against the UI but against the database
  - Do we need to interface with a supplier? If so, how should this interaction look?
  - Is the same functionality available via an API? If so, it should be tested through the API, and the UI should just be checked to interact with the API correctly.

Tools (1)

- Test framework
  - A set of guidelines like coding standards, test-data handling, which when followed during automation scripting produce beneficial outcomes like increase code re-usage, higher portability, reduced script maintenance cost etc. □

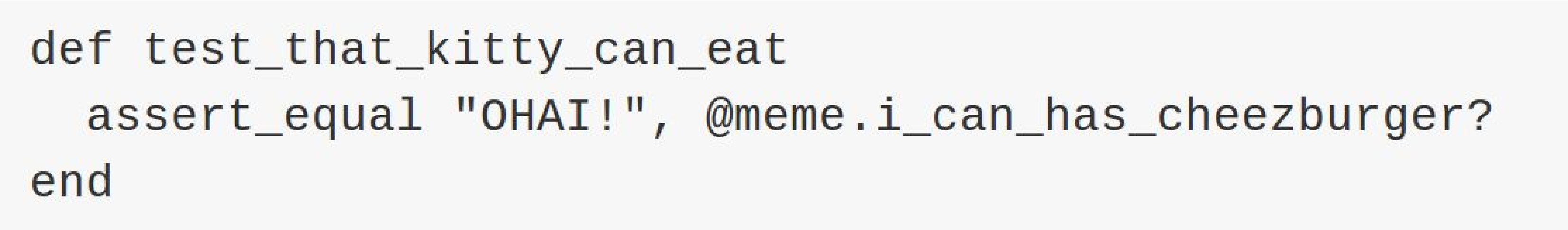


```
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculatorTest {
  @Test
  public void evaluatesExpression() {
    Calculator calculator = new Calculator();
    int sum = calculator.evaluate("1+2+3");
    assertEquals(6, sum);
  }
}
```

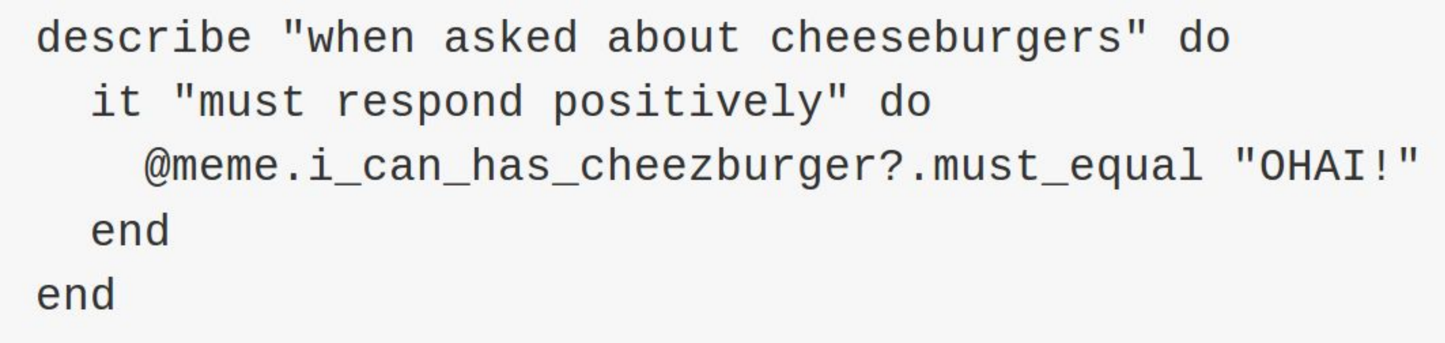
Tools (2)

- There are mainly two kinds of test frameworks
  - Unit tests style



```
def test_that_kitty_can_eat
  assert_equal "OHAI!", @meme.i_can_has_cheezburger?
end
```

  - Spec or Behaviour Driven Development (BDD) style



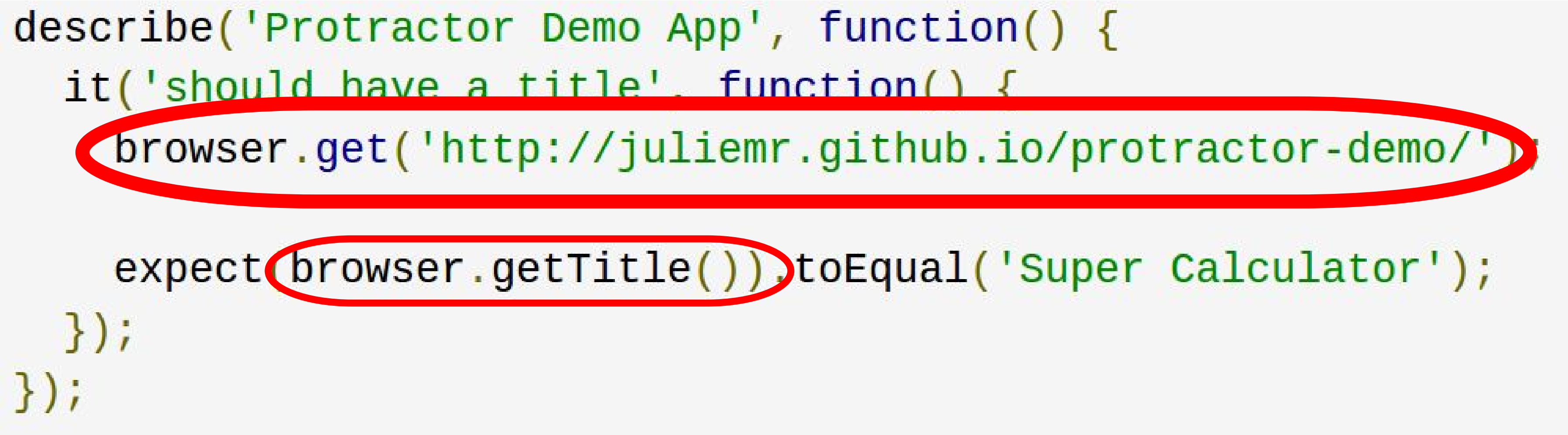
```
describe "when asked about cheeseburgers" do
  it "must respond positively" do
    @meme.i_can_has_cheezburger?.must_equal "OHAI!"
  end
end
```

Tools (3)

- List of test frameworks
  - Java
    - JUnit
    - TestNG
  - JavaScript
    - QUnit
    - Jasmine
  - Ruby
    - RSpec
    - MiniTest
  - .NET Languages
    - MSTest
    - NUnit

Tools (4)

- Testing framework
    - Library/API/Framework about how we interact with the software under test

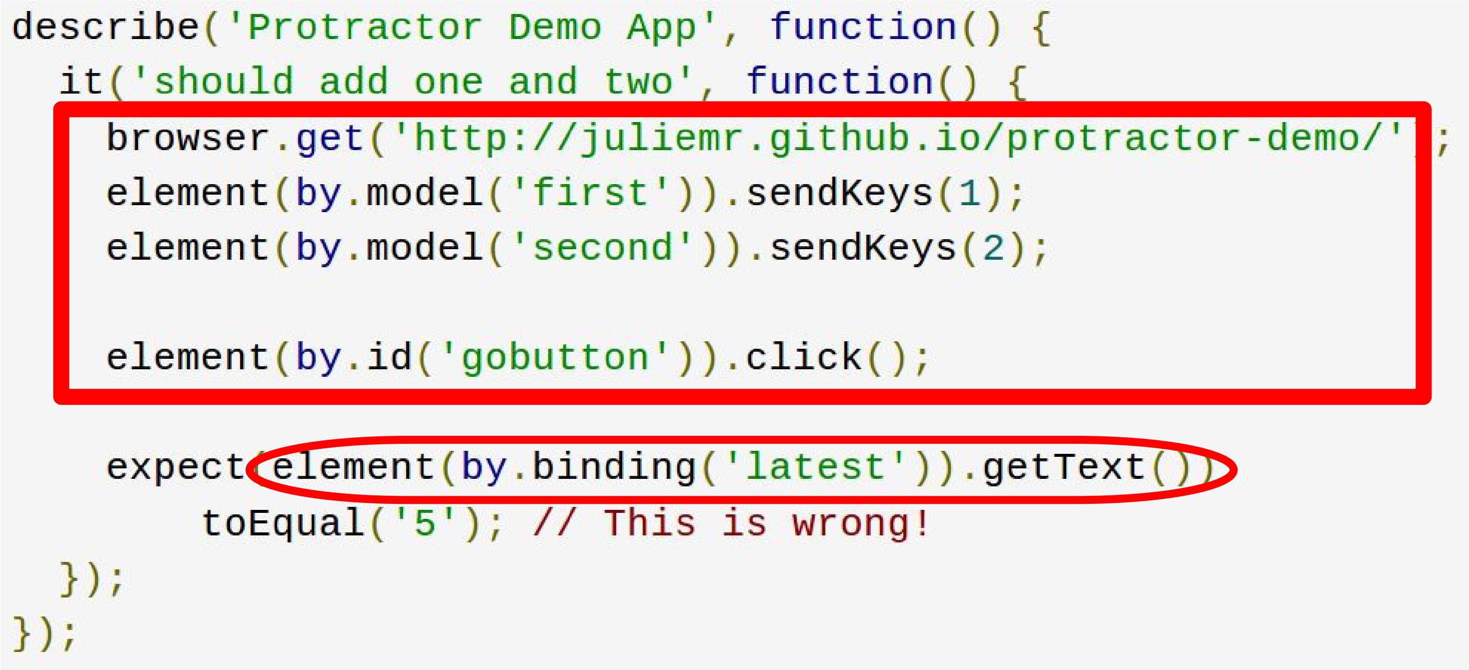


```
describe('Protractor Demo App', function() {
  it('should have a title', function() {
    browser.get('http://juliemr.github.io/protractor-demo/');

    expect(browser.getTitle()).toEqual('Super Calculator');
  });
});
```

Tools (5)

- Testing framework
  - help selecting UI elements
  - tells what the application should do



```
describe('Protractor Demo App', function() {
  it('should add one and two', function() {
    browser.get('http://juliemr.github.io/protractor-demo/');
    element(by.model('first')).sendKeys(1);
    element(by.model('second')).sendKeys(2);

    element(by.id('gobutton')).click();

    expect(element(by.binding('latest')).getText())
        toEqual('5'); // This is wrong!
  });
});
```

Tools (6)

- List of testing frameworks
  - Selenium WebDriver
  - Capybara
  - Watir
  - Protractor

Tools (7)

- Drivers for web end-to-end testing
    - The tool that drives the browser => performs the actions
- List of drivers for web end-to-end testing
    - Selenium WebDriver
        - Firefox is default
        - third party
            - Chrome
            - Opera
            - Safari
            - …
    - PhantomJS

Tools (8)

- Appium
  - Appium is an open source test automation framework for use with native, hybrid and mobile web apps.
  - It drives iOS and Android apps using the WebDriver protocol.

Advices from experts

- Whole team is responsible for automated testing
- Run your tests on every commit
- There are guys that run their e2e tests in production
- Google push on green
  - “As soon as test suites go green, deployment to production is automatically started”
- “Exploratory testing is extremely valuable and I keep wishing my teams do more of this. As your automation can only hit so many code paths and it keeps hitting them over and over; whereas; exploratory testing helps expose bugs in code paths which could have never been hit before”

But….what about manual testing

- Manual testing is amazing!
- Automated testing is great for regression testing of existing functionality. But it sucks for story testing, it doesn’t explore your system
- Automated regression tests enable you to spend more time on real testing. Better manual testing, better story testing, better exploratory testing
- In your automated testing fury: don’t forget good old human manual testing. Our users are human after all (at least for now).

QUESTIONS

Summary

- Why do we automate the testing process?
- What is the difference between test framework and testing framework?
- Why manual testing is important?