Test Automation

Tony Godwin

(Day 3 of 4)

Test Automation Class Overview

- Less than twelve hours to learn “Test Automation”

- Automation of Browser Client
  - Record and Playback (Day 1)
  - Record and Playback (Day 2)
  - Write code (Day 3) <= YOU ARE HERE
  - Write code (Day 4)

Introduction to Programming

- Assignment
- Decision Making
  - conditions
  - If then else
  - Case
- Looping
  - For loop (counting)
  - While loop (conditional)

Resources for self-education

- ON-LINE
  - https://www.lynda.com
    - Currently free
  - https://www.codeschool.com
    - Has limited free continent (intros)
  - https://www.w3schools.com
    - HTML and JavaScript
  - https://code.org
    - Aimed at middle school and High School
- REAL PEOPLE
  - https://www.meetup.com
    - LOTS and LOTS of Technology Meetups in Austin
  - Community college
  - https://www.kenzie.academy
    - Become a developer

Test Script vs Automated Test

"A script has the connotation of procedural programming. It is how manual testers often think about automation: write code to accomplish each step taken by a user and put it in a single file.

When Record and Playback tools generate code, that code can be considered a test script.

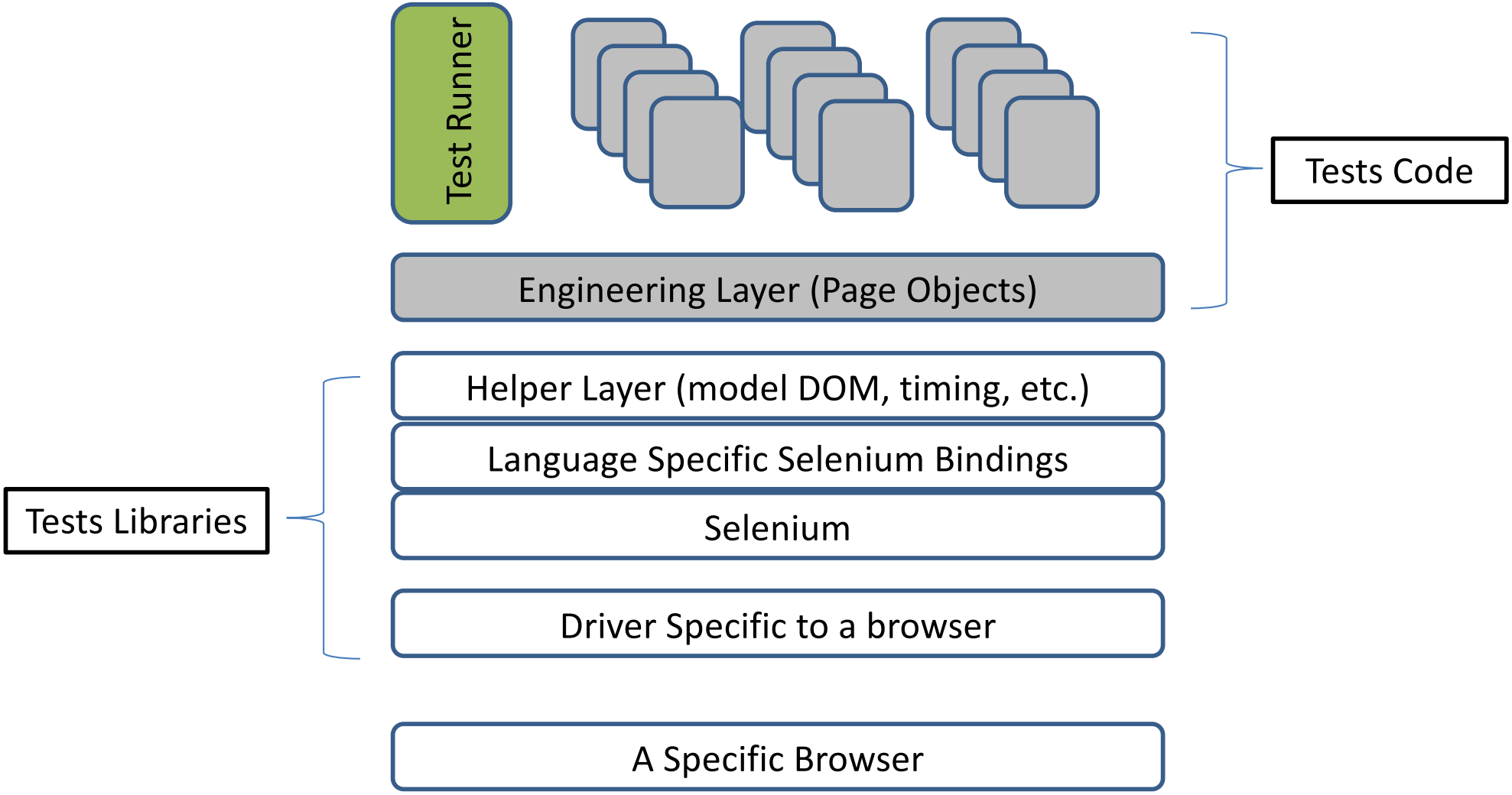
If the plan is a comprehensive test suite, then that means writing good maintainable code. Writing maintainable code means proper abstractions and modeling. It means doing more than merely writing a procedural recipe, it means using page objects and controllers and managing test data and configurations.

At that level it is not writing a test script, it is writing an automated test and using it in an automated test suite. The software industry needs to get better at either encouraging developers who already know good software engineering practices to write the tests, or training existing testers to become correspondingly skilled software developers. "

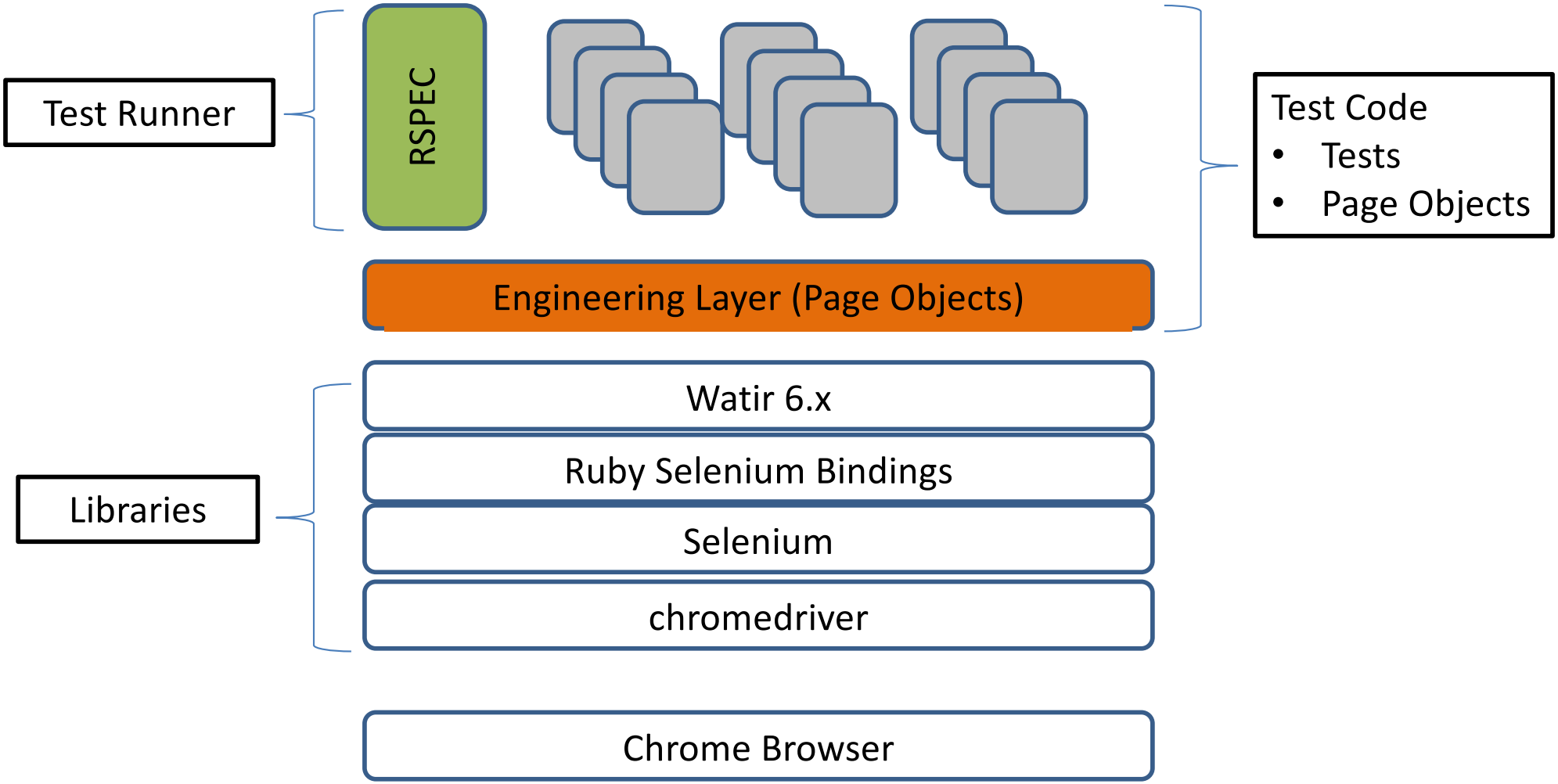
-- Titus Fortner

http://watirtight.com/2016/12/19/stop-saying-that.html

Automation Tool Stack



Ruby Automation Tool Stack



Remember: Define a Test Strategy

A great way to increase your chances of automated web testing success is to focus your efforts by mapping out a testing strategy.

The best way to do that is to answer four questions:

1. How does your business make money (or generate value for the end-user)?
2. How do your users use your application?
3. What browsers are your users using?
4. What things have broken in the application before?"

-- "*Selenium Bootcamp*" by Dave Haeffner – Sauce Labs

Behavior Driven Development

- Behavior Driven Development is a methodology for developing software through continuous example-based communication between developers, QAs, Support, and BAs

- Anyone can create scenarios (IMHO, the best scenarios are written by two or more and then have a review!). The scenarios explain how a given feature should behave in different situations with different input parameters.

- For automation you create an “Executable Specification” – the scenario serves as both specification and input into automated tests.

- BDD Scenarios have a structure
  “As a [role] I want [feature] so that [benefit]”.
  - Given [initial context],
  - When [event occurs],
  - Then [ensure some outcomes].

Behavior Driven Development

Write the test first!

1. Use the SPEC folder on computer for your tests
2. Open your text editor and create an “empty” test
3. Save the test
4. Open a command line and run the empty test

Using RSPEC write an empty test

“As a [role] I want [feature] so that [benefit]”. :
Given [initial context],
when [event occurs],
then [ensure some outcomes].

```
describe ‘As a user I must have a password to make an appointment” do
   it 'Given I want to make an appointment'
   it 'and I enter a valid username'
   it 'When I enter a bad password'
   it 'and I click the Login button'
   it 'Then I get an error message'
end
```

Steps to writing browser automation

1. Find the HTML elements you want to use
2. Write "Selenium code" to use those HTML elements
3. Verify timing – wait for HTML elements
4. Figure out what to assert/expect
5. Double check assertion by causing test to fail

Fill in the empty test

- Launch Interactive Ruby - irb
- require ‘watir’
- b = Watir::Browser.new
- b.goto ‘http://demoaut.katalon.com’

- Open the ‘Developer Tools’
- User inspector to find the element
- Translate into watir
- verify in IRB

- Open Text Editor
- Copy command from IRB into “empty test” file

Other "Frameworks"

- Python
  - https://github.com/watir/nerodia
  - https://robotframework.org
- JavaScript
  - https://nightwatchjs.org
  - https://www.cypress.io
- Java
  - Yuck

Timing

- Modern Browsers have JavaScript code to
  - Add / remove HTML elements
  - Make HTML elements appear/disappear
- This happens in reaction to events
  - User click
  - Twist widget to display additional data
- Hands-On:
  - http://the-internet.herokuapp.com/dynamic_controls
  - http://the-internet.herokuapp.com/dynamic_loading/1
  - http://the-internet.herokuapp.com/dynamic_loading/2

Questions?