Automation Approach

June 2017

WIZELINE®

General Overview

The following slides show the Automation Testing Approach to be implemented for Wizeline onboarding process.

Automation Testing Goals

Purpose

Why automated testing?

Automated testing today is considered critical for many big companies nowadays; manual testing is performed by QA engineers most of the time going to each functionality, trying several permutations or variations asserting results to the expected behavior, this comes very repetitive task during the development cycle for each code changes. With Automated Tests created it can be easily repeated, extend and perform tasks impossible with manual testing.

Objectives

- Propose an automation strategy that will allow tests to be developed and executed manually and by automation with regression test cycles.
- Develop a testing automation strategy that both technical and nontechnical tester can use and reuse for their projects.
- Automated Tests Suite as part of Continuous Integration Build Process to validate tests for each release of the application.
- Introduce developers to a TDD/BDD process to ensure the Quality of its code.
- Understand what and how will be automated.
- Increase efficiency and productivity
- Reduce time of manual execution

Automation test tools proposed

Protractor

Pros.
Is a wrapper on WebDriverJS which allows to perform live tests for webSites.

Uses smart wait/sleeps for angular services

Allow usage of features such for bindings, repeaters

Allow usage of different frameworks such as Jasmine, Cucumber or Mocha+Chai

Supports page object model, conf file and support for non angular pages.

Cons.
Requires Selenium Server (+ Webdriver & Chromedriver)

Currently requires patch for PhantomJS to work

Nightwatch.js

Pros.
Supports page object model, custom commands, custom assertions, and globals.js.

Cons.
Includes its own testing framework / assertions library, so, it Does not support frameworks like jasmine. Cucumber.

Appium

Pros.
1. Support for both platforms iOS and Android

2. Support Continuous Integration

3. Does not require access to library or source code.

4. Support various frameworks

5. Support many programing languages

Cons.
Limited to support android <4.1

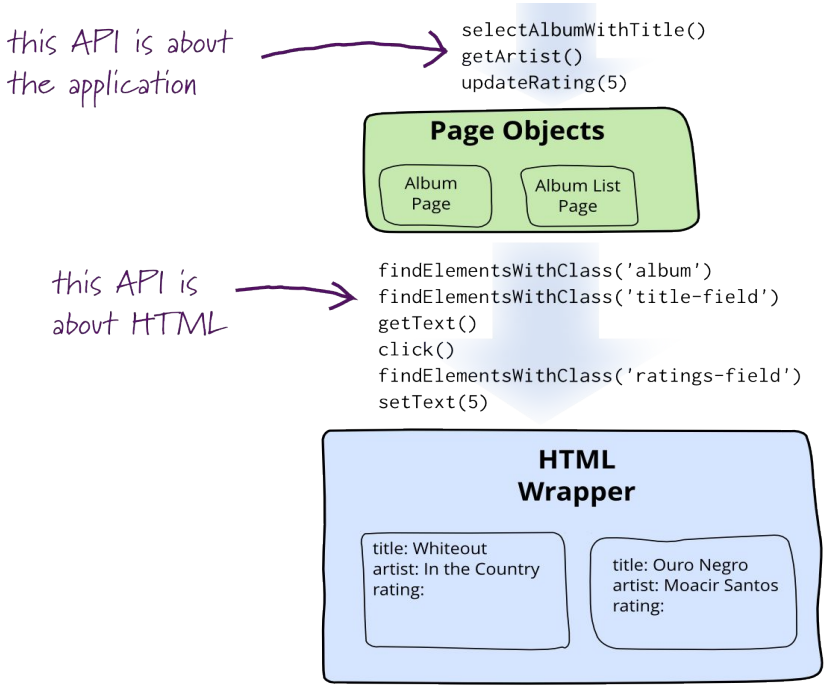
Poor documentations

Introduction to page object pattern

What is Page Object Pattern?

Is an object-oriented design technique implemented in test automation to enhance test code maintenance and avoid code duplication.

When you write tests against a web page, you need to refer to elements within that web page to click links and determine what's displayed.



Reference used:
https://martinfowler.com/bliki/PageObject.html

Page object pattern example.

Page Objects Pattern

```
var contactPage = function() {
    this.nameOfCourseTextBox = element(by.css('input[placeholder="Name of course"]'))
    this.messageTextArea = element(by.css('textarea[placeholder="Message"]'))
    this.submitButton = element(by.css('button.contact-send-btn.dark-button'))
    this.contactErrorMessages = element.all(by.css('.FormError'))
    this.contactMessageError = element.all(by.css('.FormError'))
    this.coursesDropdown = element(by.css('select.contact-form-select'))
    this.contactConfirmation = element(by.css('p.contact-confirmation-description'))
    this.contactOptions = {
        Select_Topic: this.coursesDropdown.element(by.css('option[value="0"]')),
        Courses: this.coursesDropdown.element(by.css('option[value="1"]')),
        Teachers: this.coursesDropdown.element(by.css('option[value="2"]')),
        Calendar: this.coursesDropdown.element(by.css('option[value="3"]'))
    }

    this.clickOnContactOption = function(option) {
        this.coursesDropdown.click()
        option.click()
    }
}
module.exports = new contactPage()
```

Usage on Test Script

```
it('Verify user can select any option from course option dropdown', function() {
    contactPage.clickOnContactOption(contactPage.contactOptions.Courses)
    contactPage.coursesDropdown.getText().then(function(elementText) {
        expect(elementText).toEqual('Courses')
    })
})
```

Optimizing tests readability - Gherkins Pattern.

Gherkin Pattern

Automation Approach Standardization.

Test Scenario and Script Example using Gherkin Pattern.

Scenario: Verify my courses items should be displayed when user navigate to my courses.
GIVEN: A user whose sign into the application using $username and $password.
WHEN: Users navigate to my courses page.
THEN: My courses items should be displayed.

Test example:

```
it('Login into Wize Learn then navigate to my courses page.', function() {
    browser.get('/login')
    Given.userSignIntoWizelearn(browser.params.Login.username, browser.params.Login.password)
    When.userNavigateToMyCourses()
    Then.verifyMyCoursesItemsAreDisplayed()
})
```

How tests clean should be.

What makes a test clean?

First: Readability → **Then: Readability** → **And by last: Readability**

Why Readability so important?

Readability is perhaps even more important in tests than it is in production code.

Automation can make test process F.I.R.S.T

Fast	• The test should be fast and ran quickly. • The test should get feedback as soon as possible.
Independent	• Tests should not depend on each other. Prevent cascade of failures when one test fails.
Repeatable	• Tests should repeatable in every environment. Don't make the environment an excuse for failing test cases.
Self Validating	• Each test should have a boolean output. A boolean output is easy to verify, helps to prevent extended manual verification.
Timely	• The tests should cover every use case scenario and NOT just aim for 100% coverage.

Continuous Integration

Why Automated Tests are so important in a CI Pipeline?

Having a proper automated tests that run fast, with good coverage and no results errors, won't allow to have a CI Pipeline successfully.

The Tests can be divided into multiple "Suites" each with their own objective. See the list below that gives a small overview.

Unit Test

Should be run first by developers before adding changes to the repository. The purpose of Unit tests is to test individual classes or functions.

Owned by developers.

Integration Test

Next level from unit tests is integration tests this to make sure that the modules from the application work properly with each other. This tests should run in environments that are similar to the production environment

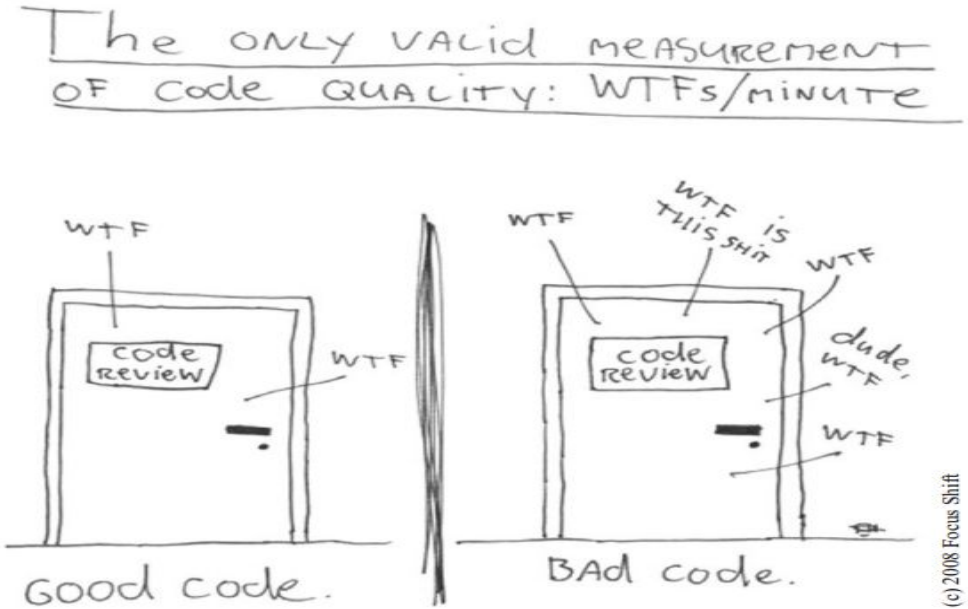
Owned by Developers / QA

Functional Test

At the end system tests should test the entire system in an environment closer to the real production environment. Ideally system tests should cover as much as the functionality involved on the release.

Owned by QA Engineers.

Why code quality is so important?



THANK YOU