轻量级B2C自动化测试框架

Joey

2011-01-11

前言

- 电商行业的网站研发具有快速迭代和交付的特点，因此自动化测试既有其开展的紧迫性，又有其人力成本、维护成本、维护周期上的限制。我们试图探索一个最佳组合，使得自动化测试框架在低维护、低前期投入的同时保持轻量级的架构。希望能和同行分享我们在实践中的一些成果、经验和思考。

周周上线，天天回归

项目测试也不能拉下

要保证线上质量，上线后还要测试

自动化测试：银弹还是炸弹？

自动化方案设计

开源的？商业的？

自己做？外包？

- 成本考虑
  - 商业外包方案：工具license，5人数个月
  - 自己做：额外的框架设计开发时间花费
- 战略考虑
  - 需要关键技术掌握在自己手中
  - 长期的维护和开发成本和改进的机会
  - 真正适合自身业务特点的框架
- 是否有Capable的人

如何降低风险

- 使用先进的测试理念
- 尽量使用成熟的开源组件，避免重复开发
- 减少初次迭代的内容，尽早建立起一个初步的骨架

Behavior Driven Development

- BDD is a **second-generation**, **outside–in**, **pull-based**, **multiple-stakeholder**, **multiple-scale**, **high-automation**, **agile** methodology.
- It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters.

BDD的几个特征

- second-generation
- outside–in
- pull-based
- multiple-stakeholder
- multiple-scale
- high-automation
- agile

- Given， When， Then

user story	行为描述语言
As a [X]	Given [Conditions /条件]
I want [Y]	When [事件发生：Actor+ Action /角色+行为]
so that [Z]	Then [observable results /可观察的结果]

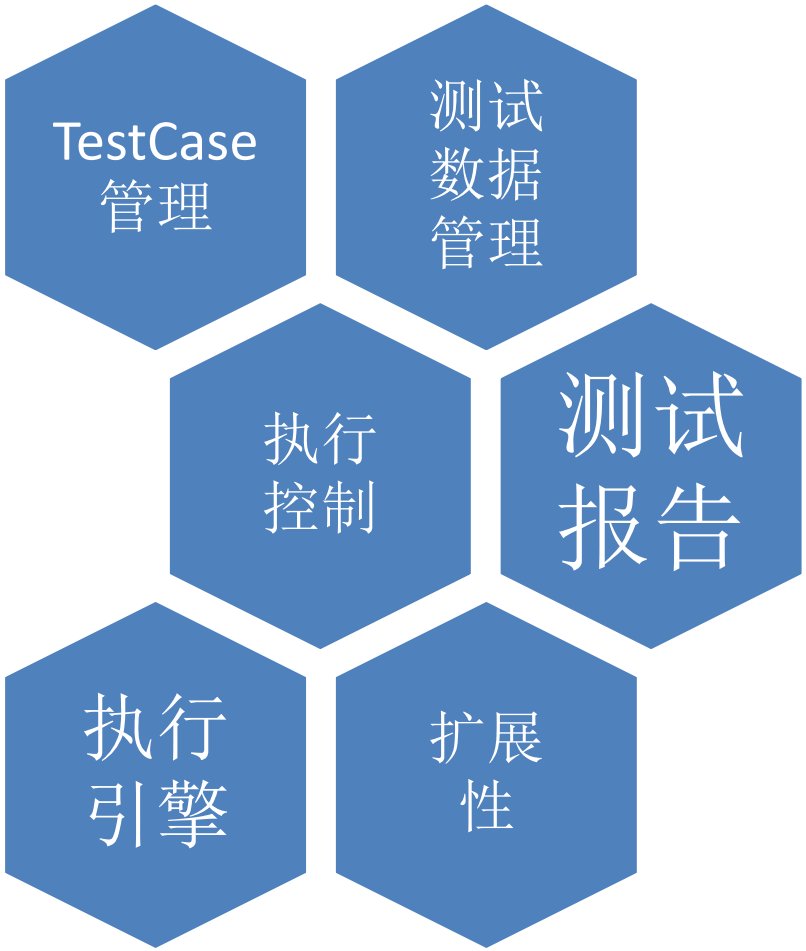
框架的设计目标

- 我们需要什么样的框架
- 现代框架包含的组件
- 如何选择相应的开源项目进行替代

我们需要什么样的框架

- 低维护成本
- 完整的自动化框架
- 易扩展，可以应对未来测试需求变化的要求
- 开发友好，易调试

自动化框架包含的组件



TestCase管理

- 可读性强，方便直接进行讨论和review，无歧义
- 方便自动化脚本的模块化和代码复用
- 用例目录组织灵活

测试数据管理

- 尽可能减少数据对用例执行的影响
- 以最小的代价支持多个stage的测试（需要不同的测试数据集）
- 测试数据与用例分离
- 实时的测试数据组装
- 方便的测试数据初始化

执行控制

- 可按预先定义的测试集批量执行测试
- 可执行指定的一个或数个用例
- 实现用例的执行错误恢复
- 支持定时执行
- 可进行分布式执行，以加快测试用例执行速度
- 易上手的执行UI，方便让不同人员操作（回归测试团队，运维Monitor）

执行引擎

- 跨浏览器兼容
- 支持丰富的元素选择器
- 运行快速并且稳定
- 支持远程执行
- 支持截图
- 调试方便

测试报告

- 提供可读性强的报告
- 提供足够的错误日志和截图以便分析用例的执行错误的原因
- 易分发

扩展性

- 方便的和其他系统集成（Testlink等）
- 方便的访问数据库进行操作
- 和已有系统相容，能方便的调用系统内部的service
- 提供对各个环节进行介入修改的手段

开源组件的方案

开源组件：

- Cucumber
- Watir-Webdriver
- Hudson
- Testlink

开发语言：

jruby

Cucumber

1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

2: Write a step definition in Ruby

```
Given /I have entered (.*) into the calculator/ do |n|
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```

3: Run and watch it fail

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
      uninitialized constant Calculator (NameError)
      ./features/step_definitons/calculator_steps.rb:2:in `Given /
      features/addition.feature:7:in `Given I have entered 50 into
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/additi
    Then the result should be 120 on the screen      # features/additi
```

4. Write code to make the step pass

```
class Calculator
  def push(n)
    @args ||= []
    @args << n
  end
end
```

5. Run again and see the step pass

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/additi
    Then the result should be 120 on the screen      # features/additi
```

6. Repeat 2-5 until green like a cuke

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/additi
    Then the result should be 120 on the screen      # features/step_d
```

http://cukes.info/

- Cucumber是一个实现了BDD理念的框架
- 角色：
  - TestCase管理
  - 执行控制
  - 测试报告
  - 方便的扩展能力

TestCase和脚本管理

- 文本化风格的用例
- 灵活的目录结构
- 内聚的项目文件组织

执行控制

- 通过features文件+ @tag
  - 相同模块内的用例放在同一个.feature文件中
  - 相同测试计划的用例按执行场景标记tag，如回归用例标记@regression
  - 相同执行环境的用例按执行环境标记tag，如生产环境标记@prod_env，Staging环境标记@stg_env
  - 用例按自己在testlink上的编号标记tag，如@YFN-112，方便个别执行失败的用例二次执行

测试报告

- 提供了一个html的富文本报告
- 支持折叠展开
- 能显示场景出错的日志

方便的扩展能力

- 在用例执行各个阶段提供了hook
- 通过hook可以实现：
  - 用例执行的错误恢复
  - 出错场景的额外日志记录和截图
  - 和testlink集成，回写测试结果

Watir-Webdriver

the most elegant way to use WebDriver with ruby.

Watir优雅的API

WebDriver的跨浏览器跨平台支持

http://watirwebdriver.com/

- 角色
  - 测试执行引擎：用来操作浏览器
- 优点
  - 简洁的API，支持xpath，使得生产效率得到提升。
  - 支持远程执行的特性使得和hudson的集成更加平滑自然
  - 更好的跨浏览器支持

HUDSON

Extensible continuous integration server

- 角色：
  - 通过shell命令方式调用cucumber
  - 是一个更高层面的执行控制
  - 提供自动化用例执行人员操作的Interface

- Hudson是一个更高层面的执行控制
  - 支持定时方式执行自动化脚本
  - 每次执行结果的存档
  - 实现每次执行后结果的email通知

- 自动化用例执行的Interface
  - 支持参数化执行
  - 支持concurrent build模式
  - 多个hudson任务配合多个不同环境的selenium server方便实现跨环境

Project Automation-frontend-production

This build requires parameters:

tag_parameters	features\mat_feature --tags @P0 cucumber 执行参数
submit_to_testlink	☐ 是否将结果返回testlink
project_name	yihaodian-frontend
testplan_name	regression-auto
port	4567 remote server port
build_name	 构建名称

Build

Testlink

- 角色
  - 统一的测试执行管理
  - 自动化测试和手动测试结果汇总在一起提供给测试管理者

开发语言：JRuby

- 提供最强大的扩展性支持
- 和我们网站的应用能进行结合，解决之前提出的测试数据准备方面的难题
- 即使之后加入webservice的自动化测试用例执行，也能胜任。

测试数据的准备

- 实时的测试数据组装
  - 通过jruby调用搜索的client jar，获取相应环境最及时有效的测试数据
- 方便的测试数据初始化
  - 通过jruby+ActiveRecord+Oracle jdbc，方便的访问数据库，操作数据库

总结

- 适合自己的才是最好的。
- 开源的比自己重复造轮子靠谱，因此尽可能多使用开源组件。
- 人才梯队建设很重要。
- 有分享才能碰撞出火花。下面是提问交流的时间。

END

- 谢谢大家