

Manual QA course

Lecture 28. Автоматизация тестирования

Дорофеев Максим

По степени автоматизации

- Ручное тестирование.
- Автоматизированное тестирование.
- Полуавтоматизированное тестирование.

Полуавтоматизированное тестирование



Автоматизированное тестирование

Процесс верификации программного обеспечения, при котором основные функции и шаги теста, такие как выполнение предусловий и постусловий, запуск, инициализация, выполнение, анализ и выдача результата, выполняются автоматически при помощи инструментов для автоматизированного тестирования

Три главных вопроса



Три главных вопроса

Why? What? How?

Why: Decision Criteria

- Экономическая выгода / ROI (возврат инвестиций) / простая математика
- Процесс тестирования программного обеспечения
- Обеспечение качества
- “Долгоиграющий” проект
- Желание и способность
- Квалифицированные специалисты
- **Требования заказчика**

Why: Decision Criteria

ROI = (стоимость ручного - стоимость автоматизации) /
стоимость автоматизации

Простая математика и простая логика

- Автоматизация в 10 раз дороже, чем Manual
- ROI менее 1 года
- Частота выполнения

Оптимизация процесса тестирования

- Автоматизированные тесты работают значительно быстрее, чем человек
- Тестирование проходит намного чаще
- Избегаем человеческого фактора
- Моделирование мультиюзер тестирования

Запросы заказчика

- Для того, чтобы сэкономить деньги -> ROI
- Чтобы выполнить Agile process -> доставить качественный продукт быстрее
- Для улучшения качества -> необходимо определить основную причину плохого качества
- Для покрытия ручных тест кейсов автоматизированными -> необходимо определить истинную цель

Итоги

- Какова реальная цель автоматизации
- Является ли данный проект пригодным для автоматизации
- Какой функционал будет эффективно автоматизировать
- Определить высшую оценку времени для прототипирования, разработки и реализации автоматизации
- Какой эффект для проекта даст автоматизация

What?

Smoke Test

Regression testing

Performance and Load testing

Unit testing

Repeatable actions / Routine tasks

Preconditions and test data for other tests

How: Approaches

Unit testing - для тестирования отдельного модуля

TDD - сначала тест, затем разработка (все тесты не пройдены), цель: все тесты пройдены

BDD - управляется и технической стороной и стороной бизнеса: бизнес-интересы + техническое понимание

Keyword Driven - определить ключевые слова (или слова действия) для каждой функции, которую мы хотим проверить

Data Driven - создание reusable тестовой логики для снижения затрат на поддержку и улучшить тестовое покрытие; тесты выполняются и проверяются на основании данных

ИТОГИ

- Автоматизация тестирования требует поставленного процесса тестирования и правильного планирования
- Не автоматизировать, если у вас не хватает времени даже для ручного тестирования
- Автоматизация тестирования является фуллтайм процессом, поэтому использовать нужных людей
- Автоматизация - это больше, чем выполнение тестов (управление, поддержка, проведение, отчет о результатах, управление тестовой средой)
- Выберите подходящую технику тестирования для проекта
- Не пытайтесь автоматизировать все
- Не разрабатывайте программу, чтобы протестировать другую программу
- Управляйте процессом автоматизации так же, как процессом разработки
- Разрабатывайте тесты и функционал для тестов так, чтобы их было проще использовать и переиспользовать
- Анализируйте и совершенствуйте процесс реализации после каждого проекта

Как выбрать инструмент для тестирования?

Прежде всего, необходимо проанализировать наиболее популярные инструменты в различных аспектах. Затем вам нужно поделиться результатами этого анализа с командой автоматизации тестирования и обсудить его

Как выбрать инструмент для тестирования?

Окружение проекта

Требования заказчика

Подход к тестированию

Знание и умение работать в команде автоматизации тестирования с этими инструментами

Как выбрать инструмент для тестирования?

Особенности

Легкий в использовании, IDE

Запись / воспроизведение

Распознавание объектов (нахождения элементов и т. д.)

Как выбрать инструмент для тестирования?

Удобство использования

Поддерживаемые ОС и платформы для тестирования

Поддержка инструмента

Интеграция в процесс разработки ПО (CI, системы контроля версий и т.д.)

Как выбрать инструмент для тестирования?

Стоимость

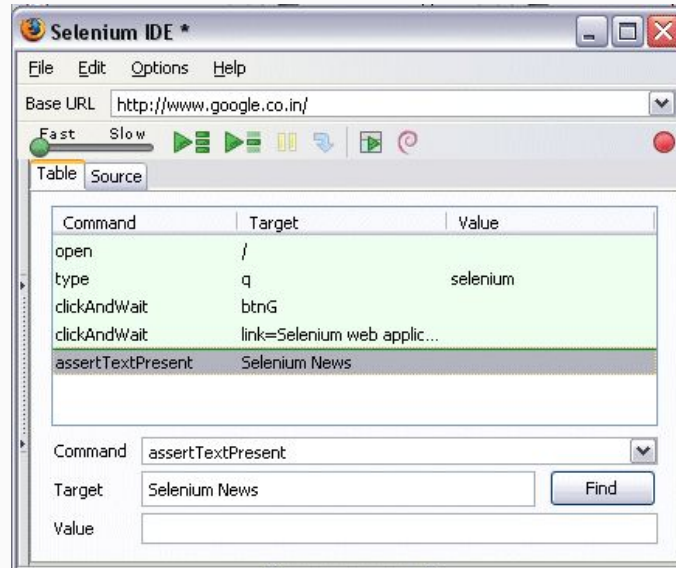
Free / Commercial

Selenium IDE

- Open Source
- расширение для Firefox
- множество плагинов



Selenium IDE



Page Object model

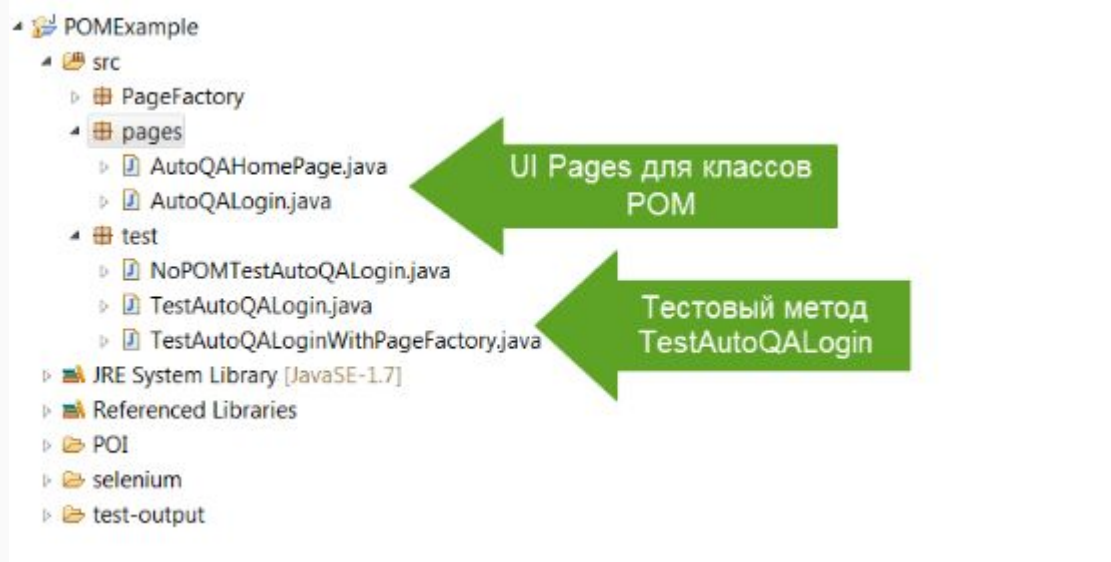
```
2  
3 I /**  
4  * This test case will login in http://autoqa.pp.ua/wp-login.php  
5  * Login to application  
6  * Verify the home page using Dashboard message  
7  */  
8  @Test(priority=0)  
9  public void test_Home_Page_Appear_Correct(){  
10     WebDriver driver = new FirefoxDriver();  
11     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  
12     driver.get("http://autoqa.pp.ua/wp-login.php");  
13     //Find user name and fill user name  
14     driver.findElement(By.id("user_login")).sendKeys("subscriber");  
15     //find password and fill it  
16     driver.findElement(By.id("user_pass")).sendKeys("2016subscriberpassword2016");  
17     //click login button  
18     driver.findElement(By.id("wp-submit")).click();  
19     String homeText = driver.findElement(By.xpath("//div[@id='profile-page']/h2")).getText();  
20     //verify login success  
21     Assert.assertTrue(homeText.toLowerCase().contains("profile"));  
22 }  
23 }
```

Page Object model

Page Object Model – это паттерн проектирования для создания **Object Repository** для элементов UI.

Согласно этому паттерну – для каждой страницы приложения/сайта должен быть определен соответствующий класс.

Page Object model



Page Object model. Преимущества.

1. Элементы объявляются отдельно от реализации теста;
2. Независимость класса с объектами от реализации тестов;
3. Становится меньше кода;
4. Методы получают более реальные имена.

Page Object model. Пример.

```
6
7 public class AutoQALogin {
8
9     WebDriver driver;
10    By userName = By.id("user_login");
11    By password = By.id("user_pass");
12    By login = By.id("wp-submit");
13
14    public AutoQALogin(WebDriver driver){
15        this.driver = driver;
16    }
17
18    //Set user name in textbox
19    public void setUserName(String strUserName){
20        driver.findElement(userName).sendKeys(strUserName);
21    }
22
23    //Set password in password textbox
24    public void setPassword(String strPassword){
25        driver.findElement(password).sendKeys(strPassword);
26    }
27
28    //Click on login button
29    public void clickLogin(){
30        driver.findElement(login).click();
31    }
32
33    /**
34     * This POM method will be exposed in test case to login in the application
35     * @param strUserName
36     * @param strPasword
37     * @return
38     */
39    public void loginToAutoQA(String strUserName,String strPasword){
40        //Fill user name
41        this.setUserName(strUserName);
42        //Fill password
43        this.setPassword(strPasword);
44        //Click login button
45        this.clickLogin();
46    }
47 }
```

Page Object model. Пример.

```
3 import java.util.concurrent.TimeUnit;
4
5 import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.firefox.FirefoxDriver;
7 import org.testng.Assert;
8 import org.testng.annotations.BeforeTest;
9 import org.testng.annotations.Test;
10
11 import PageFactory.AutoQAHomPage;
12 import PageFactory.AutoQALogin;
13
14 public class TestAutoQALogin {
15
16     WebDriver driver;
17     AutoQALogin objLogin;
18     AutoQAHomPage objHomePage;
19
20     @BeforeTest
21     public void setup(){
22         driver = new FirefoxDriver();
23         driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
24         driver.get("http://autoqa.pp.ua/wp-login.php");
25     }
26
27     /**
28      * This test case will login in http://autoqa.pp.ua/wp-login.php
29      * Login to application
30      * Verify the home page using Dashboard message
31      */
32     @Test(priority=0)
33     public void test_Home_Page_Appear_Correct(){
34         //Create Login Page object
35         objLogin = new AutoQALogin(driver);
36         //login to application
37         objLogin.loginToAutoQA("subscriber", "subscriberpass");
38         // go the next page
39         objHomePage = new AutoQAHomPage(driver);
40         //Verify home page
41         Assert.assertTrue(objHomePage.getHomePageDashboardName().toLowerCase().contains("
42     }
43 }
```

BDD

Как (As a) [X]

Я хочу (I want) [Y]

Чтобы (so that) [Z]

Допустим (**Given**) некоторый начальный контекст (данность),

Если (**When**) происходит событие,

То (**then**) убедится, что получены некоторые результаты.

BDD

Название: Клиент изымает наличные

Как клиент,

*Я хочу получить наличные из банкомата,
чтобы мне не пришлось стоять в очереди в банке.*

BDD

+Сценарий 1: На счету есть деньги+

Допустим на счету есть деньги

И Карточка валидная

И в банкомате есть наличность

Если Клиент запрашивает наличность

То Убедиться в том, что сумма вычтена со счета

И убедиться в том, что деньги выданы

И убедиться в том, что карточка возвращена

BDD

+Сценарий 2: счет превышен за рамки лимита +

Допустим счет превышен

И карточка валидная

Если клиент запрашивает наличность

То убедиться в том, что показано сообщение об отказе

И убедиться в том, что наличность не выдана

И убедиться в том, что карточка возвращена

BDD

Feature: Calculator

In order to avoid silly mistakes

As a math idiot

I want to be told the sum of two numbers

BDD

@mytag

Scenario: Add two numbers

Given I have entered *50* into the calculator

And I have also entered *70* into the calculator

When I press add

Then the result should be *120* on the screen

BDD

```
using System;
using TechTalk.SpecFlow;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Example;

namespace MyProject.Specs
{
    [Binding]
    public class CalculatorSteps
    {
        private int result;
        private Calculator calculator = new Calculator();

        [Given(@"I have entered (.*) into the calculator")]
        public void GivenIHaveEnteredIntoTheCalculator(int number)
        {
            calculator.FirstNumber = number;
        }

        [Given(@"I have also entered (.*) into the calculator")]
        public void GivenIHaveAlsoEnteredIntoTheCalculator(int number)
        {
            calculator.SecondNumber = number;
        }

        [When(@"I press add")]
        public void WhenIPressAdd()
        {
            result = calculator.Add();
        }

        [Then(@"the result should be (.*) on the screen")]
        public void ThenTheResultShouldBeOnTheScreen(int expectedResult)
        {
            Assert.AreEqual(expectedResult, result);
        }
    }
}
```

Вопросы и ответы



Ссылки

1. <http://bugscatcher.net/archives/124>
2. <http://automated-testing.info/>
3. <http://www.protesting.ru/automation/>
4. <http://www.sikuli.org/>
5. <https://www.youtube.com/watch?v=9pt5Ajf7xUw>
6. <http://sahipro.com/sahi-open-source/>
7. <http://www.seleniumhq.org/projects/ide/>
8. <http://seleniumbuilder.github.io/se-builder/>
9. http://en.wikipedia.org/wiki/List_of_web_testing_tools
10. <https://cucumber.io/>