NOVEMBER - 2014

YEAR I ~ ISSUE I

SOFTWARE TOOLS MAGAZINE

CATCH SOFTWARE IMPROVES QA OUTCOMES

Software teams that want to avoid project failure, unnecessary rework and unforeseen costs can't afford to sacrifice product quality for speed. Around the globe, development, quality and test professionals are using Catch Software's test management software, execution, and integration tools to improve their overall effectiveness.

"Enterprise teams are incredibly passionate about the role of quality assurance because they realize better business outcomes depend on it."

- Bryce Day, Founder and CEO of Catch Software

SAVE TIME AND ELIMINATE DUPLICATION

Enterprise Tester is Catch Software's flagship product. It saves as much as 60 percent of test setup time by reusing test scripts. Enterprise Tester also eliminates tool and effort duplication by tightly integrating with requirements management and defect tracking tools to provide full traceability from requirements definition and testing to incident management. Unlike some other solutions, Enterprise Tester is simple to implement and it operates across diverse customer environments.

GET MORE WITH PLUG-INS

Catch Software also offers two Enterprise Tester plugins: Duette and Rover.

Duette allows users to import and view test results from IBM Rational Functional Tester, HP QuickTest Professional (QTP) and Selenium Remote Control (RC). Using Rover, they can run tests and log defects offline using a mobile device and then sync their results back to their Enterprise Tester server later. Rover provides that flexibility in a simple-to-configure plug-in.

BE EMPOWERED TO DO GREAT WORK

Catch Software listens closely to its customers, reflecting their needs and desires in iterations and constant release cycles.

"We want to ensure that our tools, features, and functions meet our customers' actual requirements. We regularly engage with our customers to better understand how their everyday challenges are evolving and what we can do to make them more effective and productive," said Day.

Since the complex details of the software development lifecycle are not obvious or well understood by business sponsors and outside observers, Catch Software helps quality assurance and test professionals navigate the pressures of their roles so they have more time to keep pace with the volume of work, learn new skills, and continually improve the outcomes of their efforts.

"QT people are often like 'hidden dragons.'" What they do affects a company's risk, reputation, and market position. Yet their importance can be under-recognized and they often lack the voice or platform to affect change," said Day."

"This is where Catch Software tools and expertise can help the most."

To learn more about Catch Software, Enterprise Tester, Duette, and Rover visit www.catchsoftware.com

Enterprise tester duette ENTERPRISE TESTER ROVER

table of contents

november 22

So you've decided you need a test management tool…

in this issue

So you want to go Opensource? *(p.5)*

Mobile Test Framework using Appium, Cucumber-JVM & Page Objects *(p.9)*

Accelerate Load Testing Cycles with LoadComplete 3.0 *(p.26)*

How HCE and RTE can help Testers? *(P.28)*

In Focus…

Rahul Mirakhur

“ Stay hungry stay foolish is my mantra for learning!

Latest Videos

TV for Testers

Your one stop shop for all software testing videos

Dynamic Test Analysis – Leah Stockley

EDITOR'S NOTE

STM MAGAZINE

FOUNDER & CHIEF EDITOR
Lalitkumar Bhamare
editor@teatimewithtesters.com

PUBLISHER & CONTROLLER
Tea-time with Testers

EDITOR
Dr. Meeta Prakash
meetaprakash@teatimewithtesters.com

STRATEGIC PARTNER
Kiran Kumar – Quality Testing

CREDITS
Cover picture – Bigstock
In Focus section- Lalitkumar Bhamare

Created and published by:

Tea-time with Testers, B2-101, Atlanta, Wakad Road Pune-411057 Maharashtra, India.

Well! We had a dream to fulfill as we started on Tea-time with Testers. The vision was to obtain, collate and publish articles from testing stalwarts at one single place for a good read to be available easily. We needed someone to do it and Lalit and his team got started on doing this through an e-zine TTwT. Over years, TTwT has stabilized and has gained repute of a leading magazine in software testing field. We have many leading authors and thought leaders from the field regularly contributing through their articles. We are privileged to have Jerry Weinberg himself as our mentor and with his deep involvement in content.

Now that we got stable on TTwT venture and it fell into sync with its objectives, we started dreaming a little more and thought why not get the tools that are important part of our lives into this and then thought of these special editions which are focused to bring more detailed innovative technological aspects with tips and tricks of software field.

We deliberated a lot on how to go about it and then finalized on this periodical.

Software Tools Magazine will not only focus on bringing highlight to industry standard tools and their applicability in scenarios but it will also cover those small invaluable tools that people write to accelerate their jobs.

This edition brings you some amazing reads from Diwakar Menon, Shankar Garg along with some great contribution from PractiTest, Testinsane, SmartBear,Ghostinpesctor, TQP and Catch Software. Do not miss 'in focus' section where Lalit writes about some amazing people in our industry.

With these specialized, focused editions, we intend to give a platform to thinking testers, programmers and IT tool vendors to highlight various helpers (like accelerators, add-on's, tips, tricks, expert advice, new utilities etc.) with the software community so that we all can move towards yet better software world.

We also invite all of you to share such things with us and definitely share your feedback as always. Happy Reading! ☺

Meeta Prakash

- **Dr. Meeta Prakash**

So you want to go Opensource?

- Diwakar Menon

If you believe what the analysts are saying, Opensource is only now becoming mainstream! A practitioner like you would say "What a blinding flash of the obvious!" and move on to your Jenkins.

That having been said, there are problems aplenty with Opensource adoption.

Functionality is not a problem by itself. By now, the abilities and weaknesses of all tools are known. What is often missed in the consequent problem – *getting the various tools to talk to each other!*

Given the provenance of these tools, you would think that all the tools would work cohesively together. But therein lies the rub – they don't! Well, ok, they do provide interfaces that allow you to get each of these tools to talk to each other, but they are like shy Englishmen – you have to spend the effort and time to get them to talk to each other. On their own, they do not integrate and inter-operate.

Let us look at what needs to be accomplished in a typical Agile project:

- A set of requirements have to be built as a part of a sprint
- These are then translated to user stories
- These stories have to have the appropriate levels of priority, risk weightages
- Test requirements, and test cases are then created and linked back to user stories
- Testing these cases will require, say, Selenium scripts to be created and executed. Again, these scripts have to be associated to builds.
- When these user stories are delivered, (as daily drops), they need to be tested 'continuously'
- The resulting defects have a life cycle of their own but do need to be traced back to the appropriate user story.
- Data across different tests are crunched to be able to throw up predictive 'test efforts', test effectiveness and other predictive metrics

Now if you chose to manage this project using various Opensource, a typical selection would look like this:

- Kunagi to manage sprints and user stories
- Test Link to manage test requirements, test cases
- Selenium to manage automation
- Jenkins for continuous integration and
- Bugzilla for defects.
- Jasper for test analytics

Each of these tools is great by itself (and has a great following in the community). However, (there is always a 'however' with Opensource tools!), they each come with their own way of setting up users, privileges, databases etc. What is lacking is seamless integration - the ability to ensure that information and relationships are inherited both upstream and downstream.

The mainstream tools from HP or IBM do not have these issues as they are integrated (in fact working with other tools in the family is a key requirement) from the word "Go". That integration, however, comes at a huge cost and does not go far enough. Which is probably why you are reading an article on Opensource tools, anyway!

There is a crying need for a single point solution to integrate all Opensource tools. Let us look at the characteristics that would define this integration tool:

Integration, not functionality

The existing toolsets are close to best-in-class. There is no need to add new functionality in the name of augmenting the tool. The relevant tool community knows what works best and it is ideal to leave those decisions to them. What is needed is a way to make them more coherent when they work with each other. This ensures a seamless integration across all the tools across the entire lifecycle. If it can be done with minimal or no intervention, that is so much icing on the cake!

The key requirement is that the inter-component workflow be logically coherent.

Management & Control

Most Opensource tools are stand alone. What is required is the ability to form a clear, end-to-end picture of the status of the project. Sufficient information for project management, estimation, risk analysis, functional and defect prioritization should be garnered.

With both upstream and downstream integration, the tool should be able to provide complete lifecycle coverage, and work for different lifecycle models adopted. It should work with scalable development models, and across a multiplicity of projects within an organization. A defect tracking tool should be able to tag defects to requirements picked up from the appropriate requirement tool.

Traceability

The ability to link together views across the different lifecycle elements is a functionality sorely missed in Opensource tools. In an ideal scenario, the tool should have the ability to identify a requirement and view its behavior downstream in terms of its behavior during test, its impact on release. Or, take a defect, and trace it backwards to weaknesses in requirement articulation, design or development. It should allow you to manage Agile user stories, link them to tests, link the tests to automated scripts, and link the test execution data to defects.

Analytics

Today the data generated at various points is only looked at that point with no attempt being made to glean insights from the patterns residing in the data. This, obviously, is because such analytics would mean integrating another tool with its attendant integration problems.

Each of the Opensource toolsets stores a rich amount of data. Any integration tool should have, built-in, a powerful, customizable analytics engine that has the ability to weave the data together, and then leverage data along different dimensions. For example, analytics should be able to identify potential defect hotspots given a certain context. This feature will help testers to concentrate their efforts on the weakest interfaces instead of following a spray-and-pray approach.

We need the ability to look beyond the data and to comprehensively predict patterns for the future.

Plug & Play Capability

Given the diversity of toolsets in the Opensource world, there is a need to achieve seamless integration across a diverse set of tools. One should be able to plug in different tools for various functionalities during the course of the project life cycle with minimal changes. The replacement could be another Opensource tool or even a COTS tool.

Enabling the strong integration & interworking between Opensource toolsets is the need of the hour. These integrated features will help in the faster adoption of Opensource tools while being an invaluable productivity enhancer.

It would pay for itself within a very short time merely from the effort saved, leave alone the management and control capabilities or analytics.

Author: Diwakar Menon

Diwakar is the founding director of Last Mile Consultants Technology Solutions Pvt. Ltd. Prior to founding Last Mile, Diwakar has worked in senior management roles in global organisations like Tech Mahindra, Perot Systems (now Dell), Deutsche Software. He has also been a key part of strategic engagements in Tata Unisys (now TCS) and CMC Ltd.

Diwakar has over 25 years of delivery and consulting experience ranging from test consulting to pre-sales to test delivery. He also has exposure to a wide variety of domains, from Healthcare, Telecom, Capital Markets and Travel & Transportation. He has pioneered and set up end-to-end test services across multiple lines of business, consolidating the services across the organisation. Diwakar has also led consulting engagements directly with end customers, to enable them transform their test organisations driving process improvements, cost transformations and enabling shared services.

About Last Mile

We have been grappling with this problem over the past year, and have created an integrated framework using the Opensource toolsets (with some levels of customizations). The result is **TestALM.** (See: http://lastmileconsultants.com/opensource-testalm/)

We have solved most of the above problems, and are working on solving the integration conundrum. **TestALM** is available as a complete instance for you to install, adapt, and circulate within your organization, or as a hosted solution. We provide services to help you migrate from existing test solutions to this stack, and also provide services around test design, test automation and test analytics.

Last Mile Consultants is the brain child of four industry practitioners with more than 100 years of experience between them. We help our customers overcome any challenge that they face in testing.

www.lastmileconsultants.com

Mobile Test Framework using Appium, Cucumber-JVM & Page Objects

- Shankar Garg

For teams developing & maintaining mobile apps for both Android & iOS platforms, functional testing is a huge challenge. The current process of updating apps over the air is very easy, raising users' expectations of new features delivered sooner. So what this means for development teams is, they have to deliver many releases frequently and in turn what this means for Test teams is, they are continuously tasked with testing entire applications faster and more often.

Solution for such problems is Test Automation. Appium is a mobile Test Automation tool that has made life a lot easier for testers because of its capabilities and powers. Appium has made it easier for teams to test their apps for multiple platforms (one test case can test an app on iOS and android both).

But as mobile application matures and the automation test cases grow in number and project enters into a state where Regression test cases are changed frequently, that is when Test Teams starts to feel the heat about maintainability of test automation project and simultaneously adding new test cases.

And with the advent of Agile practices, Behavior Driven Development/Testing has also gained more popularity.

No tool alone can solve the problem mentioned above. Appium alone cannot solve the challenges around implementing behavior driven development & frequent UI/functionality updates. What is needed is the combination of tools which can address these problems when combined together. So on this thought process; i created a Test Framework combining Appium, Cucumber-JVM & Page Objects.

Before we move ahead, let's try list down the features/qualities that we expect from a good Test Automation Framework for mobile:

1. Even Non-Technical people like PO and BA's' can help in Test Automation.

2. Requirements are directly converted into Test cases (to eliminate difference in understanding of Dev and QA)

3. Automation tool supports multiple platforms i.e. iOS, Android etc. (Same tool can be used to write test case for iOS and Android)

4. Inter portability of same test cases on multiple platforms (Same test case can be run for iOS and Android).

5. Maintainability (How easy it is to change/update a test case)

6. Re-usability (How easy it is to use existing code to write new test cases)

To solve these problems or to achieve these features we integrated Cucumber-JVM and Appium. Let's see what the advantages of Cucumber and Appium are:

Cucumber:

It is a key tool in implementing BDD. Some of the advantages of Cucumber are:

• Cucumber Enables Non-Technical team members to help in Test Automation (by writing feature files).

• Cucumber can be implemented in multiple languages i.e. Java, Ruby and JavaScript. Since i am more comfortable with Java that's why i used java implementation of cucumber known as cucumber-JVM. But using others implementation should also be more or less same.

• Cucumber can be used to implement projects in wide business areas like enterprise web, Mobile, Web Services etc.

• Cucumber is very easy to install, implement and learn.

Appium:

It is very good for Mobile Automation and some of the advantages of Appium are:

• Support for multiple platforms iOS, Android and Firefox OS.

• Support for Native, Web and Hybrid Applications.

• Can be implemented in multiple languages Java, PHP, Python, Ruby, C# etc.

• No need to import any library in the application code.

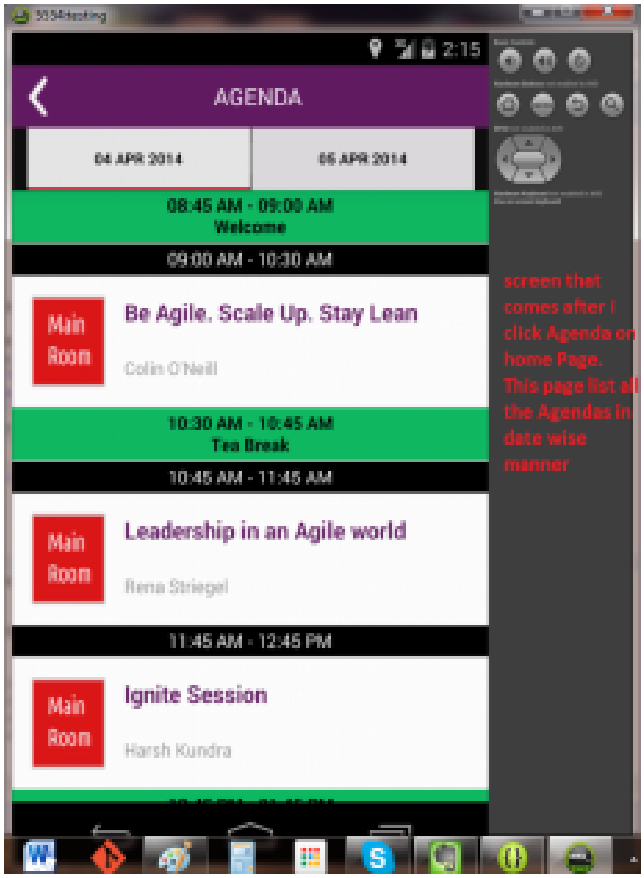
• Can be implemented with any Testing Framework i.e. Junit, TestNG, Cucumber etc.

Note: Please click on the images to enlarge them and understand the content.

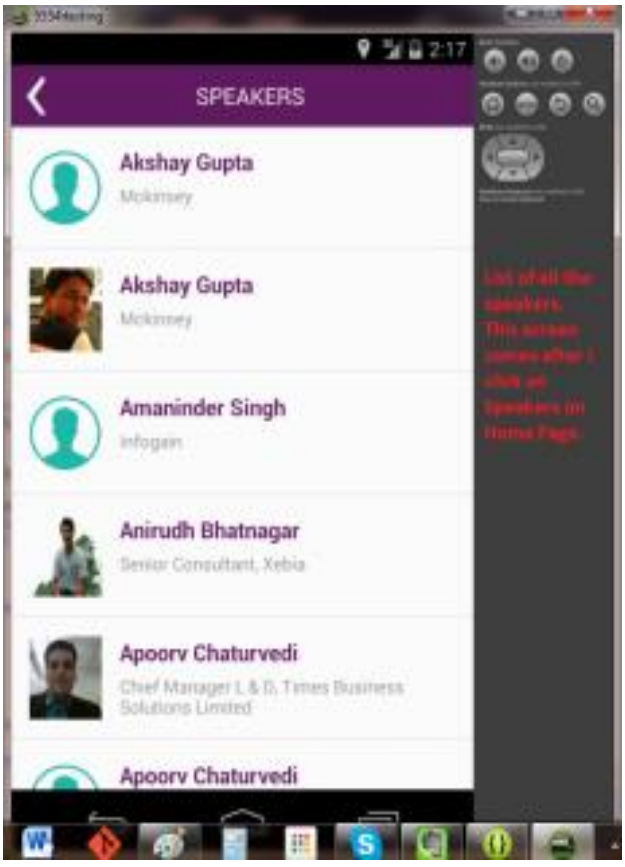
Let's first understand the Application under Test.

First Screen is the Application Landing Page, Which has four buttons: Agenda, Speakers, My schedule and Location.

Second screen is displayed when user clicks on Agenda in Home Screen.



Third screen is displayed when user clicks on Speakers on Home Screen.



Now let's see what Test Cases we are going to write:

Test Case 1: From Home Screen, user goes to agenda screen and comes back to home screen

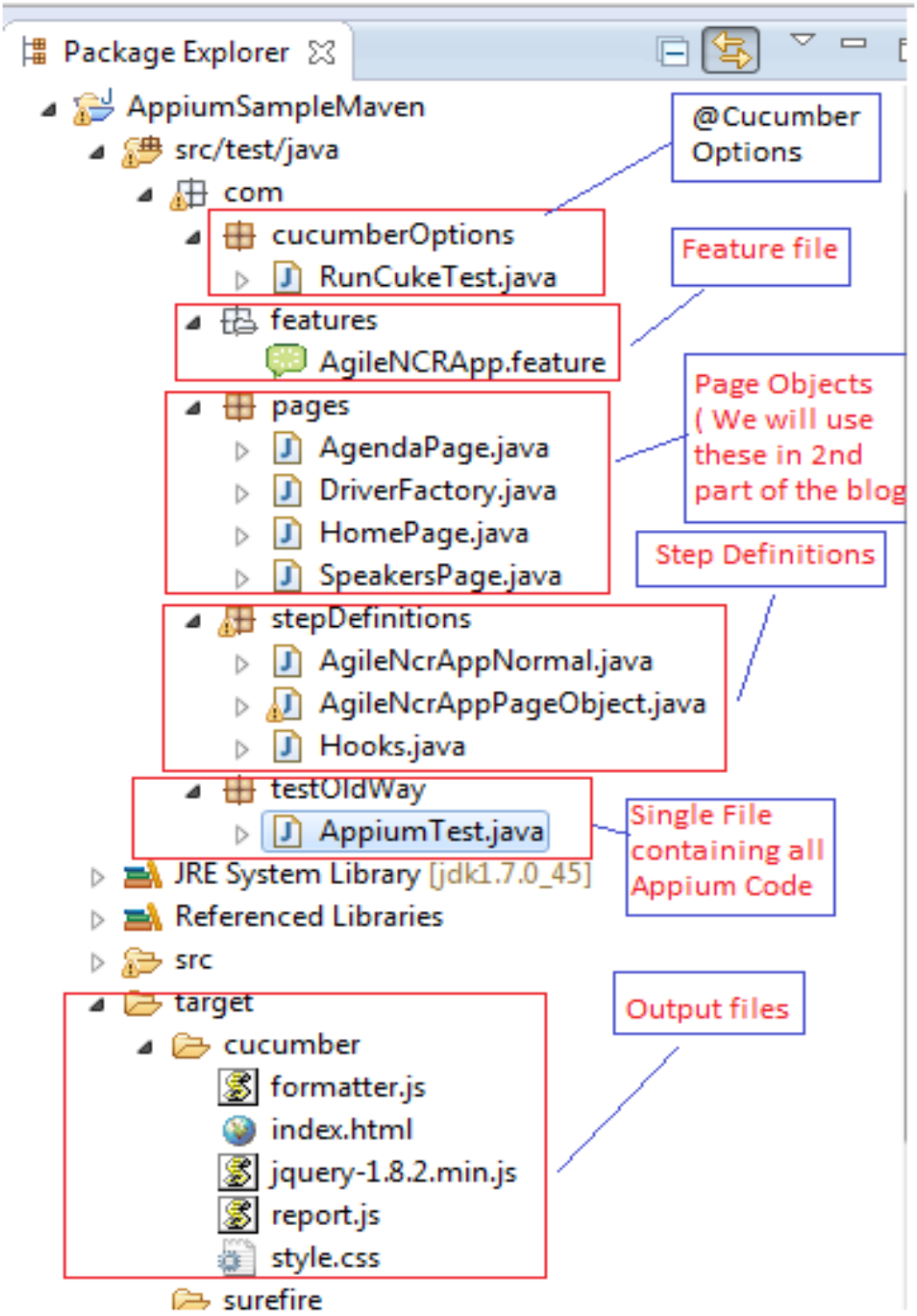
Test Case 2: From Home Screen, user goes to Speakers screen and comes back to home screen

Now let's understand the Maven Project that we created for this framework.

Step 1: Since it's a maven project, we need to add dependencies in the pom.xml for Cucumber and Appium. Below is the snapshot of the pom.xml:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.o
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.xebia</groupId>
  <artifactId>AppiumCucumberPageObjectsMaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>AppiumCucumberPageObjectsMaven</name>
  <description>Appium</description>
    <properties>
        <maven.compiler.version>2.3.2</maven.compiler.version>
        <selenium.version>2.42.2</selenium.version>
        <junit.version>4.11</junit.version>
        <cucumber.version>1.1.7</cucumber.version>
    </properties>
  <dependencies>
        <!-- cucumber -->
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-java</artifactId>
            <version>${cucumber.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>info.cukes</groupId>
            <artifactId>cucumber-junit</artifactId>
            <version>${cucumber.version}</version>
            <scope>test</scope>
        </dependency>
        <!--  Appium -->
        <dependency>
          <groupId>io.appium</groupId>
          <artifactId>java-client</artifactId>
          <version>1.5.0</version>
        </dependency>
    </dependencies>
</project>
```

Step 2: Let's understand the project structure by next snapshot:



Step 3: Since it's a cucumber project, next step is to write a feature file. I have created the feature file as per the application (screen shot and test cases attached above). Below is the snapshot of the feature file:

```
1
2  Feature: Agile NCR App
3      In order to look at Agile NCR Conference
4      As a Registered user
5      I want to specify the flow to Agenda and Speakers
6
7
8  Scenario: Agenda
9      Given user is on Application Home Page
10     Then user gets an option to choose Agenda, Speakers, Locaton and My sechedule
11     When user selects Agenda
12     Then user is on Agenda Screen
13     When user chooses to go back
14     Then user is on Application Home Page
15
16
17 Scenario: Speakers
18     Given user is on Application Home Page
19     Then user gets an option to choose Agenda, Speakers, Locaton and My sechedule
20     When user selects Speakers
21     Then user is on Speakers Screen
22     When user chooses to go back
23     Then user is on Application Home Page
```

Step 4: Cucumber needs Glue code for implementation. So next logical step would be to right step definitions. Here is the snapshot for step definitions:

```
1  package com.stepDefinitions;
2
3  import java.net.MalformedURLException;
8
9  public class AgileNcrAppNormal {
10
11     AppiumTest test = new AppiumTest();
12
13     @Given("^user is on Application Home Page$")
14     public void user_is_on_Application_Home_Page() throws MalformedURLException  {
15         test.verifyHomePage();
16     }
17     @Then("^user gets an option to choose Agenda, Speakers, Locaton and My sechedule$")
18     public void user_gets_an_option_to_choose_Agenda_Speakers_Locaton_and_My_sechedule() {
19         test.verifyHomePageOptions();
20     }
21     @When("^user selects Agenda$")
22     public void user_selects_Agenda() {
23         test.clickAgenda();
24     }
25     @Then("^user is on Agenda Screen$")
26     public void user_is_on_Agenda_Screen() {
27         test.verifyAgendaScreen();
28     }
29     @When("^user chooses to go back$")
30     public void user_chooses_to_go_back() {
31         test.clickBack();
32     }
33     @When("^user selects Speakers$")
34     public void user_selects_Speakers() {
35         test.clickSpeakers();
36     }
37     @Then("^user is on Speakers Screen$")
38     public void user_is_on_Speakers_Screen() {
39         test.verifySpeakersScreen();
40     }
```

Step 5: Next step is to right some hooks code, which will invoke Appium Instance and close the instance after test case is finished. Here is the snapshot for the same:

```
1  package com.stepDefinitions;
2
3  import java.net.MalformedURLException;
10
11 public class Hooks {
12
13     DriverFactory df = new DriverFactory();
14     AppiumTest at = new AppiumTest();
15
16     @Before
17     public void beforeHookfunction() throws MalformedURLException
18     {
19         at.setUp();
20     }
21
22     @After
23     public void afterHookfunction()
24     {
25         at.teardown();
26     }
27
28 }
29
```

Step 6: Here is the code for invoking and closing Appium Instance. You can keep this code in any class and then use that class in hooks class accordingly. I have created a Java Class AppiumTest.java and declared this code there. Here is the code snapshot for the same:

```
1  package com.testOldWay;
2
3  import io.appium.java_client.AppiumDriver;
16
17 public class AppiumTest {
18
19  static WebDriver driver = null;
20  protected static WebDriverWait waitVar = null;
21
22
23      public void setUp() throws MalformedURLException{
24
25          //initiate Appium Instance:
26          DesiredCapabilities capabilities = new DesiredCapabilities();
27          capabilities.setCapability(CapabilityType.BROWSER_NAME, "");
28          capabilities.setCapability("deviceName","Android Emulator");
29          capabilities.setCapability("platformVersion", "4.4");
30          capabilities.setCapability("platformName","Android");
31          capabilities.setCapability("appPackage", "com.xebia.eventsapp");
32          capabilities.setCapability("appActivity", "com.xebia.eventsapp.HomePageActivity");
33          driver = new AppiumDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
34
35          waitVar = new WebDriverWait(driver,90 );
36      }
37
38
39      public void teardown(){
40          //close the app
41          driver.quit();
42      }
43
```

Step 7: Next we write the code for functions that we expect our test cases to perform and the functions that we have called in the Step Definition file. This code is written in AppiumTest.java. Here is the snapshot for that code:

```
    public void verifyHomePage(){
        //wait for HomePage image and verify if the image is present is or not
        waitVar.until(ExpectedConditions.presenceOfElementLocated(By.id("com.xebia.eventsapp:id/home_banner_imageView")));
        assertTrue(driver.findElement(By.id("com.xebia.eventsapp:id/home_banner_imageView")).isDisplayed());
    }

    public void verifyHomePageOptions(){
        //wait for home page options and then verify if home page options are present or not:
        waitVar.until(ExpectedConditions.presenceOfElementLocated(By.id("com.xebia.eventsapp:id/home_banner_imageView")));
        waitVar.until(ExpectedConditions.elementToBeClickable(By.id("com.xebia.eventsapp:id/ll_agenda")));

        assertTrue(driver.findElement(By.id("com.xebia.eventsapp:id/home_agenda_title")).isDisplayed());
        assertTrue(driver.findElement(By.id("com.xebia.eventsapp:id/home_speakers_title")).isDisplayed());
        assertTrue(driver.findElement(By.id("com.xebia.eventsapp:id/home_mySchedule_title")).isDisplayed());
        assertTrue(driver.findElement(By.id("com.xebia.eventsapp:id/home_location_title")).isDisplayed());

        //print all the options on HomePage
        System.out.println(driver.findElement(By.id("com.xebia.eventsapp:id/home_agenda_title")).getText());
        System.out.println(driver.findElement(By.id("com.xebia.eventsapp:id/home_speakers_title")).getText());
        System.out.println(driver.findElement(By.id("com.xebia.eventsapp:id/home_mySchedule_title")).getText());
        System.out.println(driver.findElement(By.id("com.xebia.eventsapp:id/home_location_title")).getText());
    }

    public void clickAgenda(){
        //click on Agenda Button and wait for next page to appear:
        driver.findElement(By.id("com.xebia.eventsapp:id/home_agenda_title")).click();
        waitVar.until(ExpectedConditions.presenceOfElementLocated(By.id("com.xebia.eventsapp:id/action_bar_custom_title")));
    }

    public void verifyAgendaScreen(){
        // verify Agena Screen Title:
        assertEquals("Agenda",driver.findElement(By.id("com.xebia.eventsapp:id/action_bar_custom_title")).getText());
    }
```

```
    public void verifyAgendaScreen(){
        // verify Agena Screen Title:
        assertEquals("Agenda",driver.findElement(By.id("com.xebia.eventsapp:id/action_bar_custom_title")).getText());
    }

    public void clickBack() {
        //click Back button:
       WebElement backHome = driver.findElement(By.id("android:id/home"));
       backHome.click();
       waitVar.until(ExpectedConditions.elementToBeClickable(By.id("com.xebia.eventsapp:id/ll_agenda")));

       if (driver.findElement(By.id("com.xebia.eventsapp:id/ll_agenda")).isDisplayed()){
           System.out.println("Successfully reach to home page");
       }
       else {
           System.out.println("Did Not reach to home page");
       }
    }

    public void clickSpeakers(){
        //click on Speakers button and wait for the next page to come:
        driver.findElement(By.id("com.xebia.eventsapp:id/home_speakers_title")).click();
        waitVar.until(ExpectedConditions.presenceOfElementLocated(By.id("com.xebia.eventsapp:id/action_bar_custom_title")));
    }

    public void verifySpeakersScreen(){
        //verify speaker screen Title:
        assertEquals("speakers",driver.findElement(By.id("com.xebia.eventsapp:id/action_bar_custom_title")).getText().toLowerCase());
    }
}
```

Now, we are a stage where we have created a small cucumber-Appium-maven project that verifies behavior of the application. This project now contains two automation test cases.

After integrating Cucumber and Appium, let see from our list of desired feature which are achieved and which are still not achieved:

1. Even Non-Technical people like PO and Managers can help in Test Automation (Cucumber)

2. Requirements are directly converted into Test cases (Cucumber)

3. Automation tool supports multiple platforms i.e. iOS, Android etc. (Appium)

4. Inter portability of same test cases on Multiple platform (Same test case can be run for iOS and Android) (Appium)

5. Maintainability

6. Re-usability

Problem Statement: If we use this approach, two major challenges that still exist are:

1. Locators (code to find elements) are spread across whole file and if someone needs to change a locator then he will have to make changes at multiple places and we face a risk of some locators not being updated at all. To know at what all places the particular locator was defined is the biggest challenge.

2. There is no scope defined for a file, it may contain function for entire application or for some part only. If a new team member joins, then he will face hard time finding a function already defined.

If you would have closely noticed the AppiumTest File, then you would have seen all the locators are scattered across the file. And these are locators only for 2 test cases, imagine what would happen if this file contains locators for the whole project.

Key to the solution is Page Object. Page Objects is a framework design approach for maintaining & accessing components & controls spread across test scenarios. Page Object creates a DSL for our tests that if something changes on the page we don't need to change the test; we just need to update the object that represents the page.

To implement page objects, we need to define rules and communicate these rules to all team members. Rules could be (but not limited to):

1. Each Page object scope will be limited to one page of the application only.

2. All locators related to that page will be defined in the page object (all locators will be grouped at one place, possibly on the Top so that it is easy to find and update these).

3. All behaviors associated to a page will be described in the page object as functions.

Now let's understand what changes we made to our code for introducing Page Objects.

Change 1: We are not going to use AppiumTest.java

Change 2: Since invoking and closing Appium Session does not fall in any page scope, we created a new file DriverFactory.Java. Here is the snapshot:

*DriverFactory.java

```
1  package com.pages;
2
3  import io.appium.java_client.AppiumDriver;
12
13 public class DriverFactory {
14
15     public static WebDriver driver = null;
16     public static WebDriverWait waitVar = null;
17
18     public void setUp() throws MalformedURLException{
19
20          DesiredCapabilities capabilities = new DesiredCapabilities();
21          capabilities.setCapability(CapabilityType.BROWSER_NAME, "");
22          capabilities.setCapability("deviceName","Android Emulator");
23          capabilities.setCapability("platformVersion", "4.4");
24          capabilities.setCapability("platformName","Android");
25          capabilities.setCapability("appPackage", "com.xebia.eventsapp");
26          capabilities.setCapability("appActivity", "com.xebia.eventsapp.HomePageActivity");
27          driver = new AppiumDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
28
29          waitVar = new WebDriverWait(driver,90 );
30      }
31
32
33      public void teardown(){
34          //close the app
35          driver.quit();
36      }
37
```

Change 3: Now we created a Page Object file for each page of the application. The beauty of these files is that just by looking at the code, you can relate for which part of the application these files belong to.

First Page object is: Home page of the application (HomePage.java)

```
1  package com.pages;
2
3  import static org.junit.Assert.assertTrue;
7
8  public class HomePage extends DriverFactory{
9
10     // All the locators for Home page will be defined here
11     By homePageImage = By.id("com.xebia.eventsapp:id/home_banner_imageView");
12     By agendaButton = By.id("com.xebia.eventsapp:id/home_agenda_title");
13     By speakersButton = By.id("com.xebia.eventsapp:id/home_speakers_title");
14     By myScheduleButton = By.id("com.xebia.eventsapp:id/home_mySchedule_title");
15     By locationButton = By.id("com.xebia.eventsapp:id/home_location_title");
16
17     By backButton = By.id("android:id/home");
18
19     // All the behavior of home page will be defined here in functions
20     public void verifyHomePage()
21     {
22         waitVar.until(ExpectedConditions.presenceOfElementLocated(homePageImage));
23         assertTrue(driver.findElement(homePageImage).isDisplayed());
24     }
25
26     public void verifyHomePageOptions()
27     {
28          waitVar.until(ExpectedConditions.presenceOfElementLocated(homePageImage));
29          waitVar.until(ExpectedConditions.elementToBeClickable(agendaButton));
30
31          assertTrue(driver.findElement(agendaButton).isDisplayed());
32          assertTrue(driver.findElement(speakersButton).isDisplayed());
33          assertTrue(driver.findElement(locationButton).isDisplayed());
34          assertTrue(driver.findElement(myScheduleButton).isDisplayed());
35
36          //print all the options on HomePage
37          System.out.println(driver.findElement(agendaButton).getText());
38          System.out.println(driver.findElement(speakersButton).getText());
39          System.out.println(driver.findElement(myScheduleButton).getText());
40          System.out.println(driver.findElement(locationButton).getText());
41     }
```

Second page object is for Agenda Page. (AgendaPage.java)

```
1  package com.pages;
2
3  import static org.junit.Assert.assertEquals;
6
7  public class AgendaPage extends DriverFactory{
8
9
10     // All the locators for Agenda page will be defined here
11     By title = By.id("com.xebia.eventsapp:id/action_bar_custom_title");
12     By AgendaList = By.id("referenceOfAgendaList");
13
14     // All the behavior of Agenda page will be defined here in functions
15
16     public void verifyAgendaPage()
17     {
18         waitVar.until(ExpectedConditions.presenceOfElementLocated(title));
19
20         assertEquals("Agenda",driver.findElement(title).getText());
21     }
22
23     public void clickonAgenda()
24     {
25         //code goes here
26     }
27
28 }
```

Third page object is Speakers page. (SpeakersPage.java)

```
1  package com.pages;
2
3  import static org.junit.Assert.assertEquals;
6
7  public class SpeakersPage extends DriverFactory{
8
9
10
11     // All the locators for Speaker page will be defined here
12     By title = By.id("com.xebia.eventsapp:id/action_bar_custom_title");
13     By SpeakersList = By.id("referenceOfSpeakersList");
14
15     // All the behavior of Speaker page will be defined here in functions
16     public void verifySpeakersPage()
17     {
18         waitVar.until(ExpectedConditions.presenceOfElementLocated(title));
19
20         assertEquals("Speakers",driver.findElement(title).getText());
21     }
22
23
24     public void clickonSpeaker()
25     {
26         //code goes here
27     }
28
29 }
30
```

Change 4: Step Definitions file also be updated to use the Page objects. Here is the snapshot of the Step Definitions file:

```
1  package com.stepDefinitions;
2
3  import java.net.MalformedURLException;
15
16 public class AgileNcrAppPageObject {
17
18     DriverFactory df = new DriverFactory();
19     HomePage hp = new HomePage();
20     AgendaPage ag = new AgendaPage();
21     SpeakersPage sp = new SpeakersPage();
22
23     @Given("^user is on Application Home Page$")
24     public void user_is_on_Application_Home_Page() throws MalformedURLException  {
25         hp.verifyHomePage();
26     }
27     @Then("^user gets an option to choose Agenda, Speakers, Locaton and My sechedule$")
28     public void user_gets_an_option_to_choose_Agenda_Speakers_Locaton_and_My_sechedule() {
29         hp.verifyHomePageOptions();
30     }
31
32     @When("^user selects Agenda$")
33     public void user_selects_Agenda() {
34         hp.clickAgenda();
35     }
36     @Then("^user is on Agenda Screen$")
37     public void user_is_on_Agenda_Screen() {
38         ag.verifyAgendaPage();
39     }
40     @When("^user chooses to go back$")
41     public void user_chooses_to_go_back() {
42         hp.clickBack();
43     }
44     @When("^user selects Speakers$")
45     public void user_selects_Speakers() {
```

Now, you can see yourself that if someone has to make a change, how easy it becomes to change and at the same time how easy it is to add new functionality.

After integrating Cucumber and Appium with Page Objects, let see from our list of desired feature which are achieved and which are not still:

1. Even Non-Technical people like PO and Managers can help in Test Automation (Cucumber)

2. Requirements are directly converted into Test cases (Cucumber)

3. Automation tool supports multiple platforms i.e. iOS, Android etc. (Appium)

4. Inter portability of same test cases on Multiple platform (Same test case can be run for iOS and Android) (Appium)

5. Maintainability (Page-Objects)

6. Re-usability (Page-Objects)

I really hope after reading this article, you will have a better understanding of what are the features required from a good automation framework and how to use different tools/design approaches to achieve those features.

Shankar is an Agile enthusiast with expertise in Automation Testing. He has worked in all phases of the software development life cycle in the capacity of Automation Tester, Automation Test Architect and Automation Lead. He has excellent understanding of Cucumber-JVM, Selenium RC, Selenium Webdriver, Appium, Robotium, Core Java, TestNG and Junit. He also has good knowledge of Agile methodologies such as Scrum and Software Development processes such as TDD and BDD.

He started his journey in the IT industry as a Java developer and then he moved into the automation world. Since then, he has been working with different clients for Website and Mobile Automation using different flavors of Selenium.

He is a Certified Scrum Master (CSM), a Certified Tester (ISTQB), and a Certified Programmer for Java (SCJP 5.0) and Oracle 9i (OCA).

So you've decided you need a test management tool....

When a company decides it is time for a new test management system they have a variety of tools to choose from.

If you look into how people manage their testing and QA process you will see that most teams usually start out documenting their process in excel spreadsheets. This is a simple and free solution, however, it is only fitting for small scale companies, or single project endeavors. Once there are more than 3-4 testers in the mix, excel becomes a much less friendly solution.

Other companies might start working with the large scale "Enterprise Solutions". These often look attractive at first glance, with all their features and options, but when you try to start working with them you realize that all their bells and whistles are not really needed. In fact, the software is so complicated to use and requires so many resources from your company to install and manage it, than in many cases these solutions become shelfware on some QA Manager's office.

How to choose a test management tool?

Selecting the right QA Management System is not a trivial task but it's not rocket science either. The most common problem is that test engineers and managers start evaluating tools, only to abandon the tasks because they did not sit to plan this process correctly in advance.

Following are 5 simple steps to help you evaluate and select the appropriate QA

Management System for your team and your company's needs:

Step 1 - Project lead selection

Assign the task to one person. If the team leader is not going to be doing the evaluation himself then there should be at least be one person from the team with enough experience and knowledge to lead this project. The chosen evaluator should be allocated with time for the task, and not leave it as a "side project".

Step 2 - Create an evaluation checklist

Create a requirements list that describes your needs and the expectations from the new tool. Do this by thinking about the reasons that made you start looking for a new tool and write them down.

It is also important to include any company restrictions such as: budget, SaaS, security etc. so you don't waste time on something that turns out to be irrelevant for your team.

Step 3 - Vendor research

Use the features and limitations checklist you created as your guide to make sure you don't drown in the sea of options. Then look up external recommendations or complaints in sources like QA groups (on LinkedIn for example) or QA related forums. End this step with a list of about 4-6 options.

Step 4 - Vendor shortlist

Narrow your vendor list further to 2-3 options with the help of your evaluation checklist. Then proceed to register for demos and sign-up to work with an actual trial project to begin the real evaluation of each of the tools in your shortlist. At this point it is advised to get the rest of the team involved to help assess the tools you are evaluating and figure out the pros and cons of each solution.

Step 5 - Live trial

To really understand the abilities, the limitations and the general flow, you will need to work with the tool.

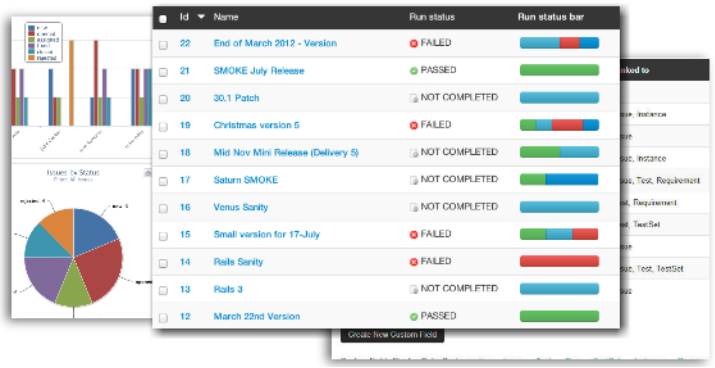
Probably the best way is to take one small project or feature from your everyday testing tasks and use the tools that are relevant in a limited but realistic end-to-end evaluation.

Bonus Step - choose a vendor a not a tool

It is important to remember that when you choose a tool you are also choosing a Vendor that will support your needs and answer (or not!) any questions you may have.

Take this into account and make sure that the people you are working with and the people who will support you (not only the ones who are trying to sell to you) are responsive, knowledgeable, and that you feel good working with them.

The six things you need to know about PractiTest:



On the spectrum between free options to Enterprise Solutions, PractiTest is a test management system positioned perfectly in the middle. With just the right amount of functionality to help you lead your process, but without all the rest of the features that make many system too complicated and hard to work with. PractiTest was built and is continuously updated based on real testers needs.

PractiTest's vision is that testing tools are meant to make software QA simple. The test management tool should feel natural to one's process, require little to no effort to utilize, while enabling successful and speedy product releases.

1. A complete solution

PractiTest is a complete test management solution that can manage every step of the QA process from requirements to tests, runs and bug tracking.

"...We were finally free from the never-ending task of sending excel sheets with the issues back and forth to the supplier." - ***Dutch ministry of foreign affairs***

2. Seamless integrations

PractiTest offers seamless integration with popular bug tracking tools such as Jira, Redmine, Bugzilla, and more; as well as automation solutions such as Selenium, QTP, TestComplete, etc. You can even use PractiTest's API to connect external frameworks such as Jenkins.

This makes PractiTest an ideal choice for those searching for a test management solution that will work with their existing systems without having to shift over completely.

3. Start working and migrate your data quickly

Migrating from Excel to PractiTest is done by simply importing your existing data using built-in functionality.

The UI is very intuitive so that your team can start working almost immediately after a short introduction.

"Deployment was very easy. Due to its simplicity and friendly UI everybody loves using PractiTest, even those who were used to using other, better known solutions. Additionally, the API is simple and easy to understand and maintain" - ***Udi Vered: QA Director , Zerto.***

4. Elevate the quality of your QA

PractiTest will help improve your testing process, because you will be working with a system that was designed to manage and match your methodological needs.

"Practitest provides the fluency and visibility needed to manage complex multi-platform testing, organizing the work of their Scrum process, while keeping up with the fast pace of product releases." - ***David Elimelech: Director of QA, Aternity.***

"With PractiTest, we can access test information from a central location that provides a clear view of testing processes and status. This assists me in making more informed QA decisions, and in reporting to management". - ***Stephen Musal: Lead QA Tester, Freeman***

5. Customize to your needs.

PractiTest is a fully customizable tool so you can make it fit your individual testing and process needs. You can define the filters to organize your information, the fields you require in your test sets and runs, the data to display in your reports and all the way to customizing your advanced integrations using the API.

6. Professional Support that is with you all the way

One of the biggest advantages of PractiTest is not a technical feature, but rather its outstanding customer support.

Our team of experts offers methodological human support throughout the life of your project. This personalized and professional approach really sets PractiTest apart from the rest of the QA management solutions:

"The PractiTest team has always been great to work with. If I have found an issue they respond back quickly and very professionally. Without spending hundreds of thousands of dollars a year, we are getting a great product with great support that fits the needs of our department." - ***Stephen Musal: Lead QA Tester, Freeman***

"Last but not least, one of the advantages of Practitest is its customer service: It is simply excellent". - ***Udi Vered: QA Director , Zerto.***

About

H.S. PractiTest is a cloud based Innovative test management tool. A technology and methodology leader in the field of Application Life Management (ALM), PractiTest provides its customers with the best in class End-to-End system to meet their Testing and QA needs. It is easy and affordable, but very flexible and methodological.

PractiTest enables organizations to ensure visibility and communication at all levels. As well as helping project development teams streamline and manage their testing processes, while providing management with a clear and simple view of their project status at all times.

Contact Us

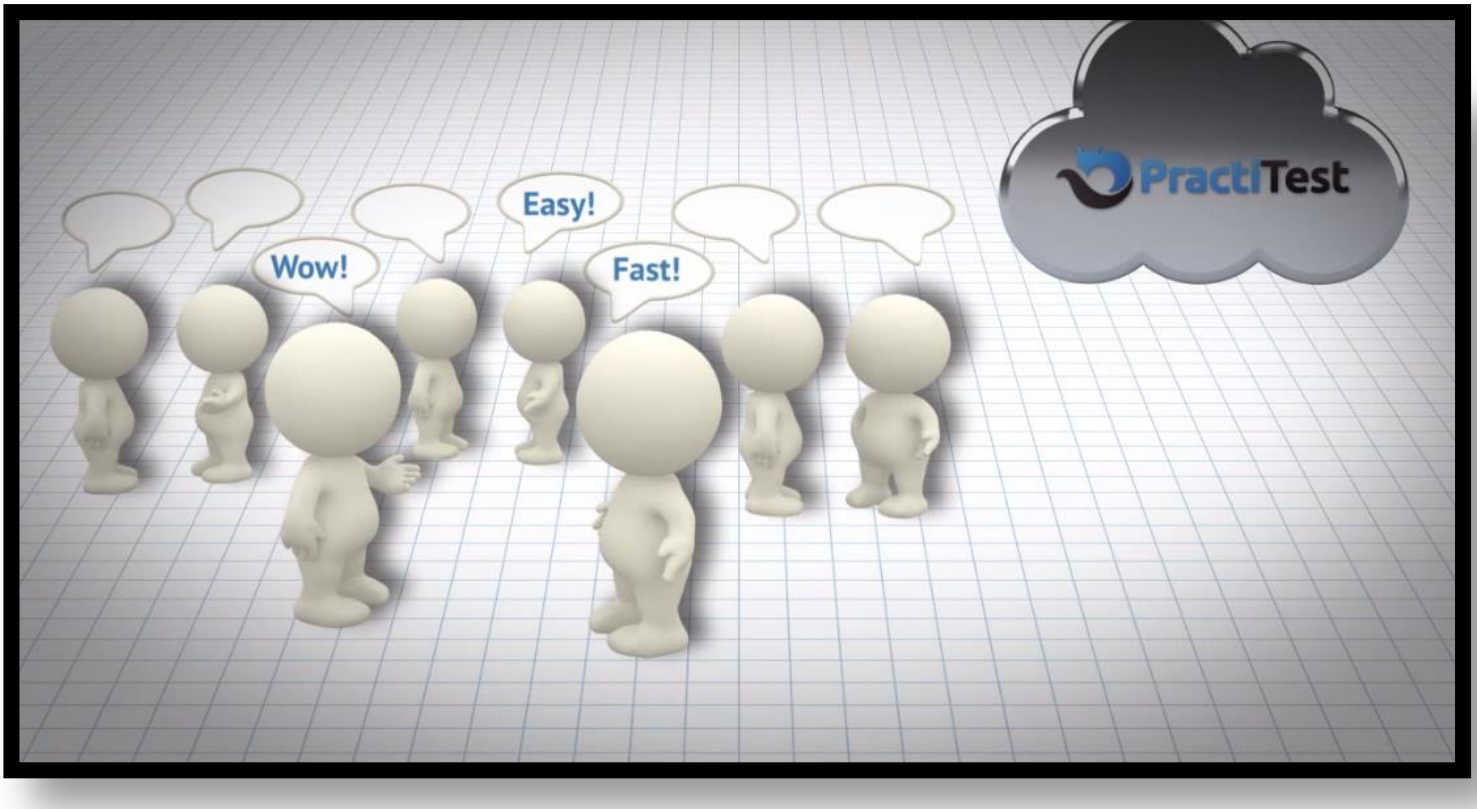
Phone: +1-847-993-3064

email: info@practitest.com

website:www.practitest.com

And don't forget to watch PractiTest Introdcution video exclusively on TV for Testers.

CLICK HERE!



Accelerate Load Testing Cycles with LoadComplete 3.0

The SmartBear team is thrilled to announce new and rebranded version of LoadUIWeb – LoadComplete 3.0. The focus of this release is to help organizations drastically reduce the time required to test and optimize application performance.

Software teams are under increasing pressure to deliver quality applications in shorter release cycles. More often than not, load testing is often left to the last minute by many organizations since new revenue enhancing features takes precedence over basic performance testing. Teams are therefore frequently left with a short amount of time before the deployment of an application, to identify, uncover, and resolve serious performance issues.

While these challenges affect all applications, they only get worsen with mobile application development because of shorter release cycles. A recent Forrester Research report captures this trend by stating, "Performance is not a priority for today's mobile development teams….Development teams typically want to focus on mobile app performance, but all too often business sponsors prioritize new features over sustained performance engineering."

As a result of these trends, the need for a tool that generates actionable insights in these shortened performance testing cycles becomes even more critical. LoadComplete 3.0 has thus been focused on providing features that accelerate load testing cycles and thereby significantly decrease performance testing time. With new comparison reporting feature available in LoadComplete 3.0, testers have the ability to analyze server and browser side metrics across various load tests in a single report. As a result, there is no need to do any manual analysis. LoadComplete 3.0 further reduces the time required for analyzing and debugging test scripts. Through a single click, testers can identify request dependencies among different pages of an application, which further minimizes the need for time consuming manual analysis that would otherwise be required to develop and debug load scripts.

But wait, there is more! LoadComplete 3.0 now supports 2,000 virtual users per agent, an increase from a previous limit of 250 virtual user limit for each agent. This means testers now need fewer agents to run a load test. The hardware investment required to run a load test is drastically reduced, thereby significantly decreasing the costs associated with running a load test. Additionally, with LoadComplete 3.0, SmartBear has introduced support for 250 & 10,000 VUs.

Try these awesome features yourself by clicking here.

LoadComplete

by SMARTBEAR

Did you

ever wish to have software testing mindmaps repository?

If you did, your wish has come true.

Visit http://apps.testinsane.com/mindmaps/ and explore testing mindmaps.

www.testinsane.com | sales@testinsane.com | .com/testinsane | @testinsane

You can't ignore this.

HOW HCE & RTE HELP TESTERS?

There could be testers who write their own tiny tools which could help in solving a specific problem or helping them to fasten the process of some task or help them to test better in a given context. At Test Insane, we have a dedicated team for developing these utilities in-house which could help our testers to add value to testing activity. And also, we share the same utilities / programs / tools with the testing community so that we all can move towards a better testing world.

Recently, we developed HCE and RTE for testing two different things. Without much ado, let us speak about what are these exactly. HCE stands for HTML comment extractor and RTE stands for RestFul Test Endpoints.

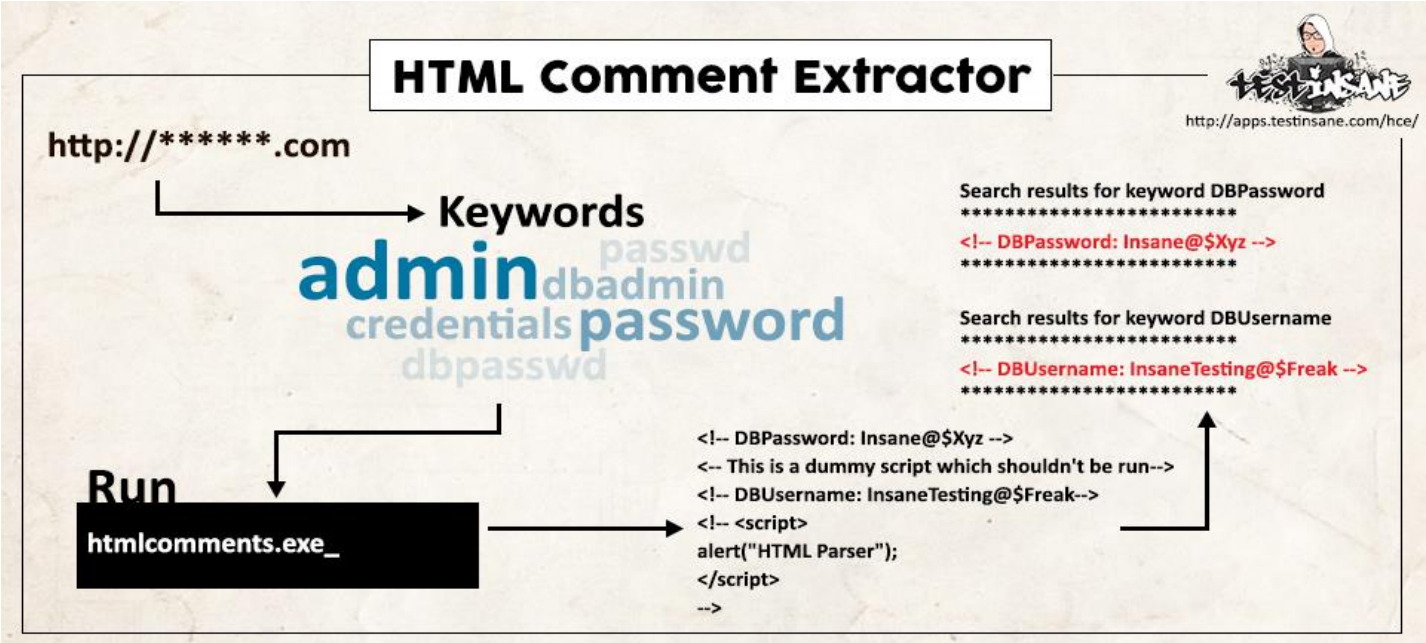
How does HCE help?

Usually, you see the HTML and JS style comments in the client side code. Now, there could be a possibility where your comments could carry some sensitive information like admin credentials, analytics sensitive data, database table names, database name / username / password etc. Most of the testers just go through the black-box approach and this test could be missed.

What problem did we see in order to develop this tiny utility?

We saw that testers need to go through every page and scroll to see the comments. And this we found it cumbersome and tedious way / monotonous way! We thought, we could build a utility which could extract all the HTML and JS style comments and write it to a log file. And then testers could go through all the comments at once place. And then we added an extra feature which was keyword based extraction, that is; once all the comments are extracted we run it through keywords like admin, passwd, database, db name, password etc. which acts like sensitive data. And there is one more log file created which extracts comments with sensitive data if found.

How does it work? [visit our HCE page at http://apps.testinsane.com/hce/]



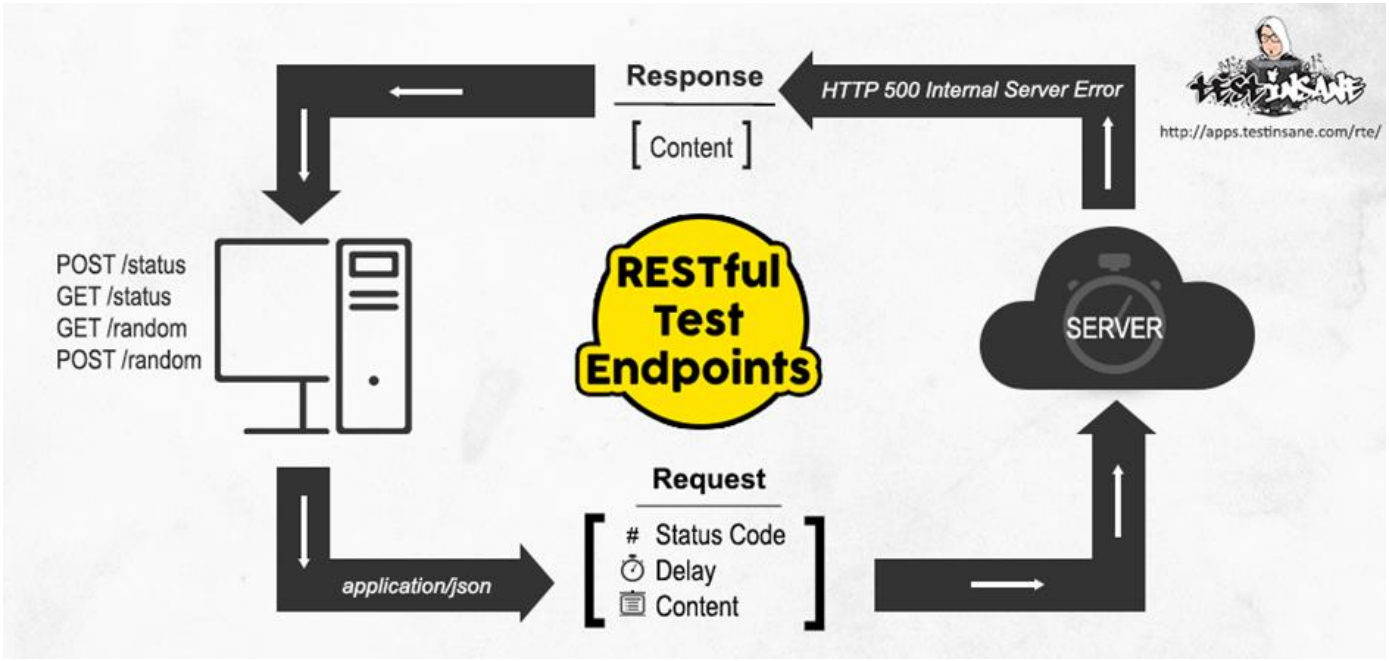
How does RTE help?

Do you test your application for HTTP status codes like 404, 500, 301, 302 and others? Well, 404 is easy, but how about simulating other responses from the server and watching how does your application code handle such responses.

RTE is like a micro-service where we simulate various responses from the server with attributes like delay etc.

RTE helps in prototyping, unit testing, check automation where you could get test coverage across various server responses.

How does RTE work?



Visit http://apps.testinsane.com/rte/ for more help and documentation!

About Test Insane:

Test Insane Software Testing Private Limited was founded in 2014 with its headquarters at Bengaluru, Karnataka (India) with the intention of delivering high value with highly skilled software testers and software developers who work together to test better. Test Insane specializes in Exploratory Testing approach.

Test Insane offers exploratory testing services, test automation services, and test consulting across web and mobile platforms.

What if I tell you that I know of a techie who is crazy about life and about everything he does between getting up and hitting the bed at night? What if I tell you that I know of a tester who has grown playing with Apple products can speak for hours about how to break them? Guess I must be in school when he was busy building his expertise around mobile testing.

Meet **Rahul Mirakhur**! Rahul is currently heading QA department for Atimi Software Inc. (www.atimi.com), a leader in mobile and cross-platform development with core strengths in developing for the new generation of smartphones.

A passionate tester since his development days in late-1990's, Rahul moved into Software Testing way early in his IT professional career and since been involved in leading, mentoring and creating highly-skilled test teams in disparate domains like Desktop publishing, Security,

Anti-Virus, Games, Print media, Mobile, etc. He is well-known mobile testing expert in Indian testing community and has been spreading his knowledge through various conferences (like the ones hosted by STeP-IN and UNICOM), workshops and his blogs.

Being an inspiration to many who know of his work, Rahul feels that his un-satiated thirst for knowledge, self-criticism for constant improvement and passion for software testing field makes him what he is today. He also likes software testing field because it does not help him learn just about software but also about human psychology; a field he is ever so fascinated by.

One of the things that I personally like about Rahul is his vast experience in software testing that has helped build successful teams (that have tested numerous iOS apps which have been featured in Apple's AppStore). I also admire the way he delivers his talks.

Rahul is probably the first QA director in India I have met who is ardent proponent of Exploratory Testing and has advised his teams for investing time in AST's BBST curriculum; for enhancing their testing skills. Well, I appreciate his decision.

When I asked him what would be his advice for fresh testers, he said; *"There is no short-cut to hard work and sacrifices have to be made if one wants to grow professionally in today's razor-sharp competitive market".* And *"Stay hungry stay foolish is my mantra for learning",* he added.

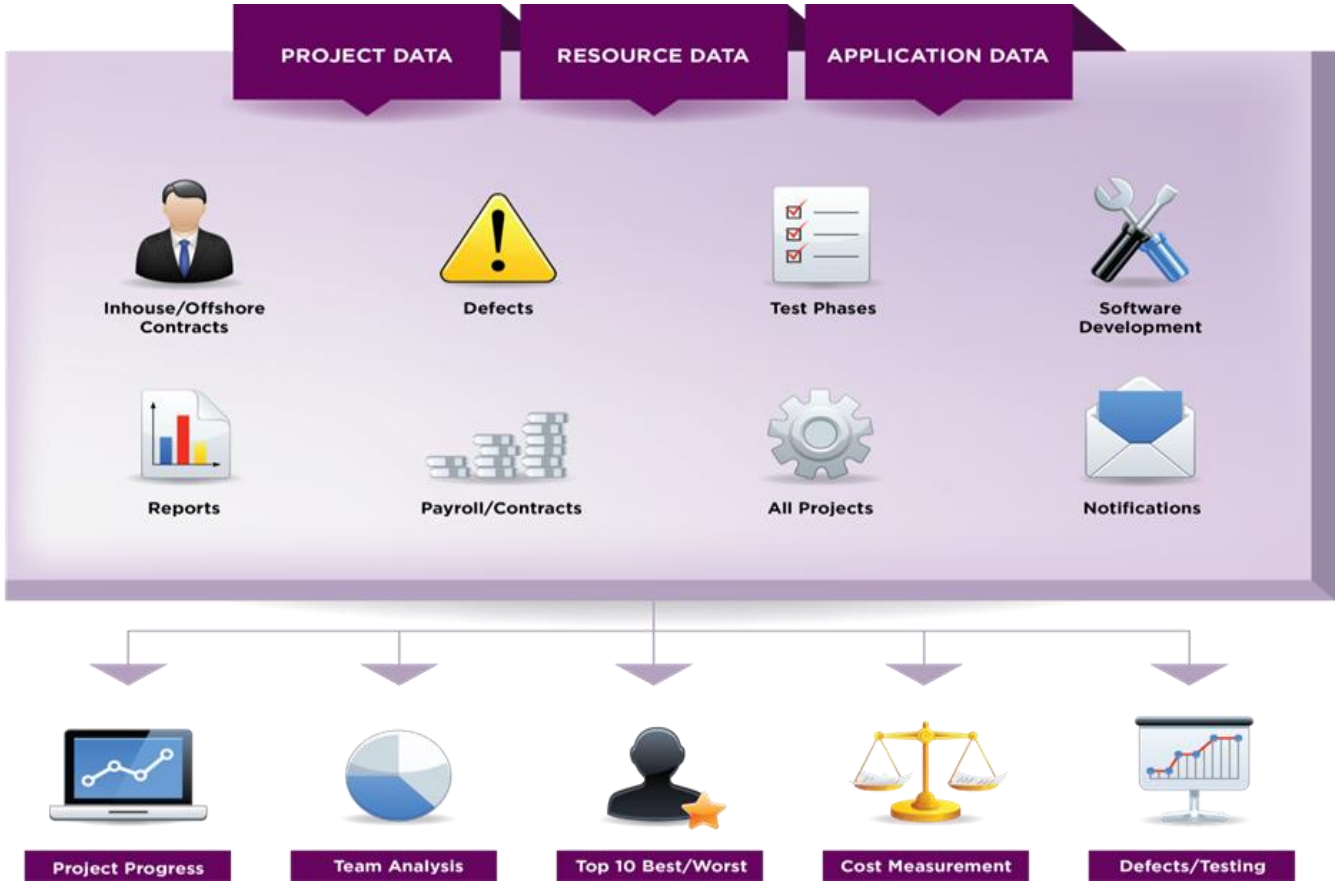
When he is not working with (and in) his teams, keeping abreast of iOS world, Rahul loves to spend every possible waking moment with his family.

If you care about knowing what really helps make great mobile apps in today's ever-changing world of software, want to understand various testing techniques that are employed by world-class mobile app test teams when delivering under high pressure of time, scope and resources, then Rahul is definitely the person who can help you. And I am sure that he will be more than happy to help.

Contact Rahul on - thinkd1fferent@mac.com or follow him on Twitter @rahulmirakhur.

- Lalitkumar Bhamare

TeamQualityPro is a real-time integrated platform designed to evaluate the entire ecosystem of Application development with drill down investigation into specific team activity. The dashboards present on demand information to make informed assessments from any location at any time.



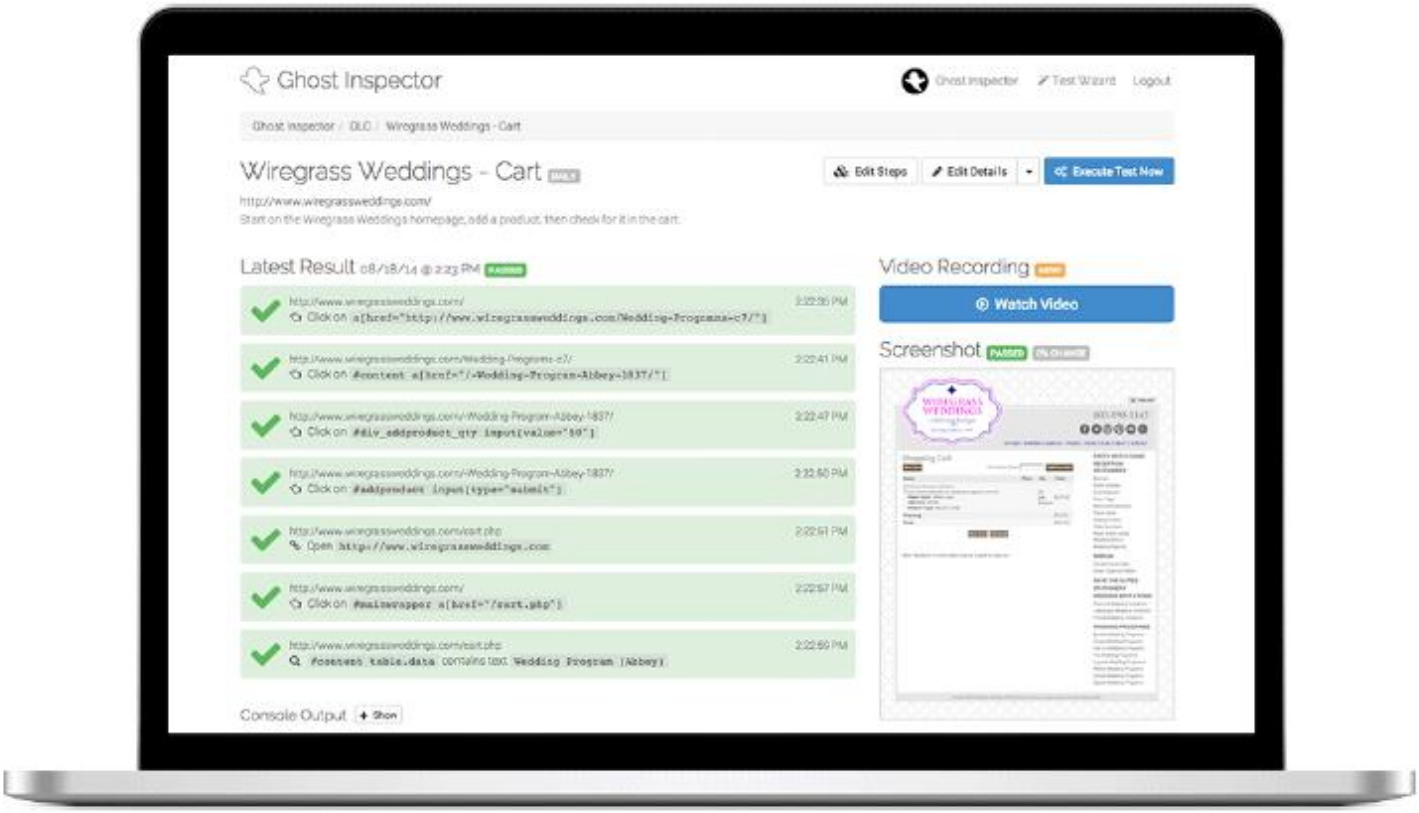
Take the first step in saying:

YES To:	NO To:
• Real time reporting • Instantaneous data gathering • Objective, vetted data • Comprehensive data • Data that mirrors your process • Data that is always current	• Ad-hoc reporting • Data warehouse projects • Subjective data • Missing data • Not aligned data • Wrong time frame data

Ghost Inspector

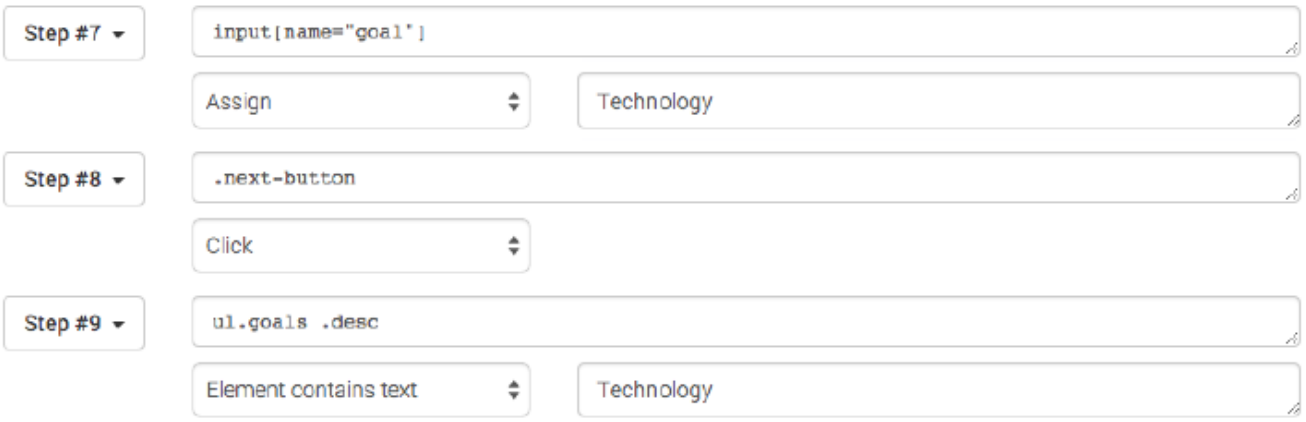
Automated UI testing made simple

Ghost Inspector is a new automated UI testing and monitoring platform. It allows you to build or record simple tests that check specific functionality in your website or application. These tests are immediately synced to the cloud and run continuously to catch any bugs, issues or regressions that occur.

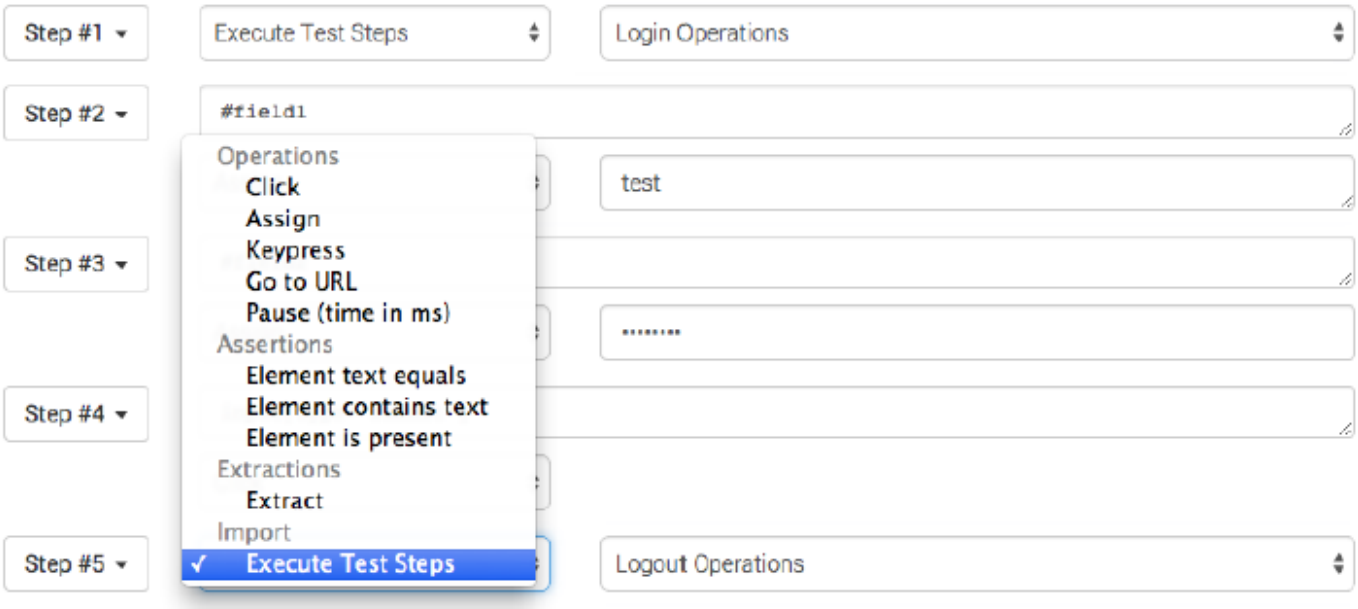


Ghost Inspector is taking a new approach to tackling an old problem: the unwieldy nature of automated browser testing. There have generally been two approaches to this problem. The first approach is to record tests, which are often fragile and expire quickly. The second approach is to code your own tests, which takes a good deal of effort and a high level of expertise. Ghost Inspector attempts to merge these approaches without losing the benefits of either.

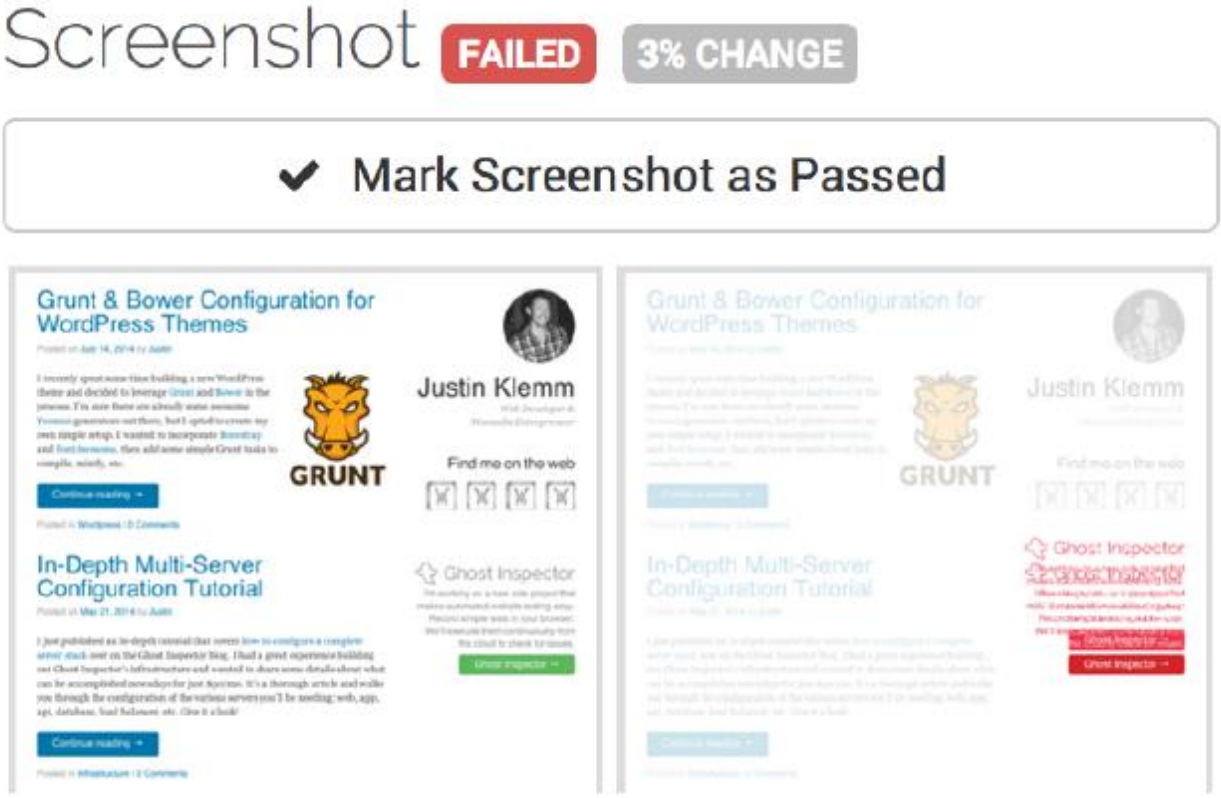
The system offers you the ability to record tests via a Chrome browser extension. However, instead of outputting rigid files, the recorder syncs your test steps to the Ghost Inspector cloud service so that it can be run continuously. A robust editor is provided to update and maintain your tests. You can also build tests from scratch in the editor, the same way you would if you were coding your own test.



Ghost Inspector is embracing concepts like reusability to help you get more mileage out of your tests. Tests can be nested within other tests as subroutines, so that specific actions can be segmented out and reused. For instance, you can create a "Login Operations" test that logs into your system, and then insert it as the first step of various other tests in the suite. This keeps your login code in one place and makes it easy to maintain if changes to your application occur.



In addition to the core features above, Ghost Inspector also provides a platform that stores results, records video, sends notifications, plugs into continuous integration frameworks and even does some fancy things like comparing screenshots to detect visual regressions.



Ghost Inspector is currently using PhantomJS to run its automated tests, but they hope to expand their browser ecosystem through the use of Selenium in the future. They offer a free tier with 2 tests so that everyone can try out the service. For more information, visit *https://ghostinspector.com/*

Gaining knowledge...

has

just become

more awesome!

Don't believe?

try

testomaton

Quality Assurance via Automation

Software testing startup specializing in test automation services, consulting & training for QA teams on how to build and evolve automation testing suites.

Services

Test System Analysis and work estimation

Review of client's system (either in development or on early stage) to produce a Test Plan document with needed test cases and scenarios for automation testing.

Tests creation and Automation

Development of test cases (in a test plan) and Test Scripts to execute the test cases.

Regression and Test evolution

Development of Regression test suites and New Features suites, including test plans and test scripts or maintenance of existing.

Architecture Consulting

Review of software architecture, components and integration with other systems (if applicable).

Trainings

Depending on the particular need, we can provide training services for: Quality Assurance theory and best practices, Java, Spring, Continuous Integration, Groovy, Selenium and frameworks for QA. For further information please check our web site.

We enjoy putting our experience to your service. Our know-how allows us to provide consultation services for projects at any stage, from small up to super-large. Due to our range of expertise we can assist in software architecture and COTS selection; review of software designs; creation of testing suites and test plans with focus on automation; help building quality assurance teams via our extensive training curriculum.

look us up: www.testomaton.com - twitter: @testomaton