

# Tests automatisés côté client

- Tests unitaires et javascript
- Intégration continue et javascript
- Tests et identification des composants HTML
- Tests fonctionnels web avec les Robots
- Tests fonctionnels web avec les bibliothèques d'automatisation
- Tests fonctionnels utilisateurs et Rich Internet Application
- Spécifications par l'exemple

Révision : Septembre 2016

# Tests unitaires et javascript

- Utiles lorsqu'on développe beaucoup de javascript
- xUnits appliqués au langage **javascript** qui tend à se généraliser ces dernières années
- Quelques outils phares :
  - **jsUnit** : framework de tests unitaires pour javascript
  - **QUnit** : initialement utilisé pour tester **JQuery unitairement**
  - **jsNUnit** : ASP.Net
  - **jsTestDriver** pour tester du code javascript sur plusieurs navigateurs et lancer de l'analyse de couverture du code par les tests.
  - **Jasmine, Mocha** : vers le BDD et l'expression plus naturelle du comportement attendu.

# Exemple avec JsUnit

- **JsUnit s'exécute DANS le moteur javascript du navigateur.**
- On écrit les tests dans une page HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JsUnit Factorial Tests - Chapter 13</title>
  <link rel="stylesheet" type="text/css" href="../../jsunit/css/jsUnitStyle.css">
  <script type="text/javascript" src="../../jsunit/app/jsUnitCore.js"></script>
  <script type="text/javascript" src="../../src/main/webapp/factorial.js"></script>
  <script type="text/javascript">
    function test15() {
      assertEquals(factorial(15), 1307674368000);
    }

    function testRegEx() {
      var actual = "JSUnit en Action";
      assertTrue(actual, /en/.test(actual));
      assertFalse(actual, /out/.test(actual));
    }
  </script>
</head>
<body>

    <h1>JsUnit Example</h1>
    <p>This page contains jsUnit tests</p>
    <p>To see the tests, view the source for this page.</p>

</body>
</html>
```

```
function featureASuite() {
  var result = new JsUnitTestSuite();
  result.addTestPage("../tests/featureA/Test1.html");
  result.addTestPage("../tests/featureA/Test2.html");
  return result;
}
```

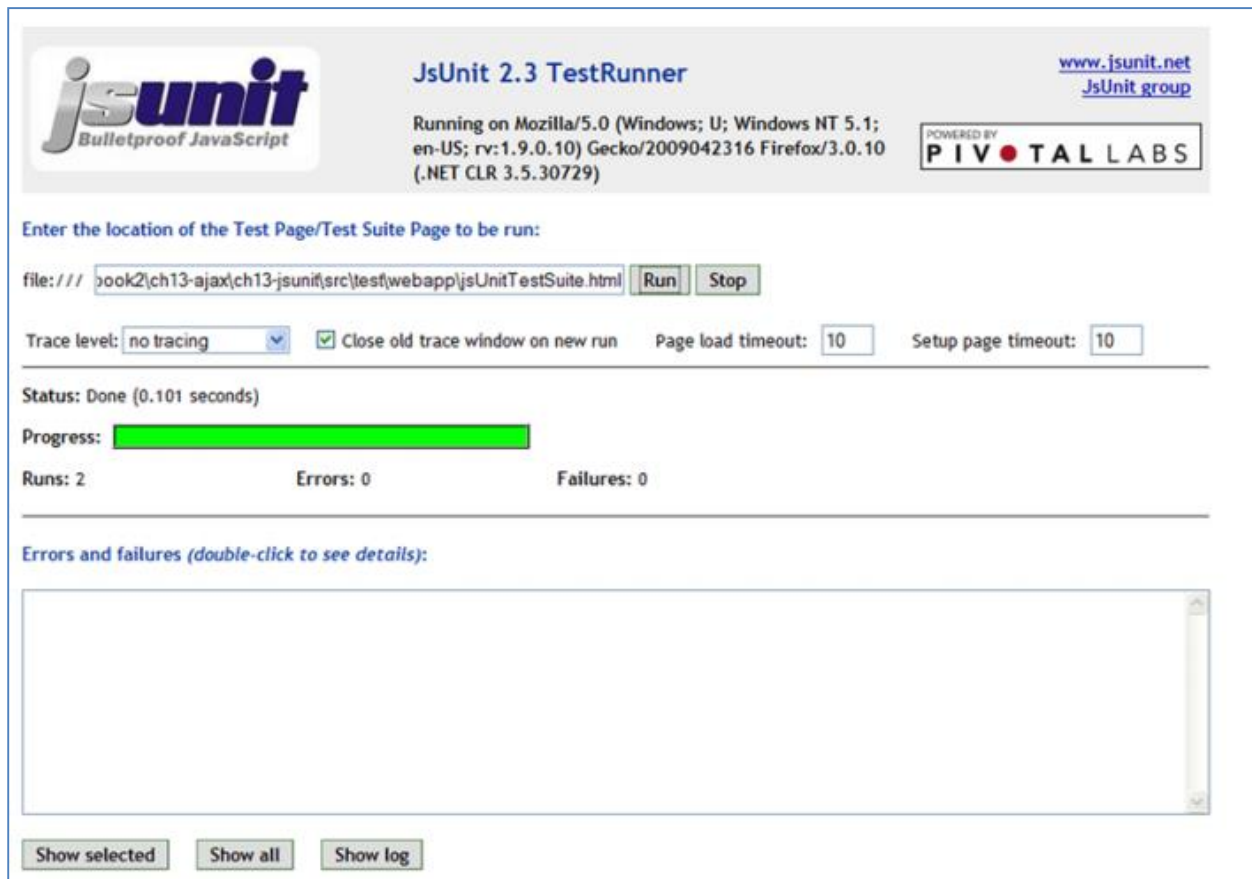
## Système d'assertions

- `assert([message], booleanValue)`
- `assertTrue([message], booleanValue)`
- `assertFalse([message], booleanValue)`
- `assertEquals([message], expectedValue, actualValue)`
- `assertNotEquals([message], expectedValue, actualValue)`
- `assertNull([message], value)`
- `assertNotNull([message], value)`
- `assertUndefined([message], value)`
- `assertNotUndefined([message], value)`
- `assertNaN([message], value)`
- `assertNotNaN([message], value)`
- `fail(message)`

# Lancements des tests jsUnit

Utilisation du runner : `jsunit/testRunner.html`

Utilisation de ant + java



The screenshot displays the JsUnit 2.3 TestRunner web interface. At the top left is the 'jsunit' logo with the tagline 'Bulletproof JavaScript'. To its right, the title 'JsUnit 2.3 TestRunner' is shown, followed by the version information: 'Running on Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.10) Gecko/2009042316 Firefox/3.0.10 (.NET CLR 3.5.30729)'. A link to 'www.jsunit.net JsUnit group' is in the top right. Below the header, a text input field contains the file path 'file:///book2/ch13-ajax/ch13-jsunit/src/test/webapp/jsUnitTestSuite.html', with 'Run' and 'Stop' buttons to its right. Below the input field are controls for 'Trace level' (set to 'no tracing'), a checked checkbox for 'Close old trace window on new run', and two timeout fields: 'Page load timeout: 10' and 'Setup page timeout: 10'. The status bar shows 'Status: Done (0.101 seconds)'. A green progress bar is labeled 'Progress:'. Below the progress bar, the results are summarized: 'Runs: 2', 'Errors: 0', and 'Failures: 0'. A section titled 'Errors and failures (double-click to see details):' contains an empty text area. At the bottom, there are three buttons: 'Show selected', 'Show all', and 'Show log'.

**jsunit**  
Bulletproof JavaScript

**JsUnit 2.3 TestRunner**

Running on Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.10) Gecko/2009042316 Firefox/3.0.10 (.NET CLR 3.5.30729)

www.jsunit.net  
JsUnit group

POWERED BY  
**PIVOTAL LABS**

Enter the location of the Test Page/Test Suite Page to be run:

file:///book2/ch13-ajax/ch13-jsunit/src/test/webapp/jsUnitTestSuite.html

Trace level:  ☒ Close old trace window on new run Page load timeout:  Setup page timeout:

Status: Done (0.101 seconds)

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and failures (double-click to see details):

# Exemple avec QUnit

```
/* unitTest.js */
QUnit.test("currency conversion example", function( assert ) {
    assert.equal(convertINR.currencyConversion(100,1/63),'1.59', "100 INR is equal to 1.59 USD" );
});
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit basic example</title>
  <link rel="stylesheet" href="qunit-1.18.0.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="qunit-1.18.0.js"></script>
  <script src="unitTests.js"></script>
  <script src="currencyConversionTest.js"></script>
</body>
</html>
```

### QUnit basic example

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Filter:

QUnit 1.18.0; Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0) Gecko/20100101 Firefox/38.0

Tests completed in 9 milliseconds.  
1 assertions of 1 passed, 0 failed.

1. currency conversion example (1) Rerun 2 ms

1. 100 INR is equal to 1.59 USD

@ 1 ms

# Assertions QUnit

**async():** Instruct QUnit to wait for an asynchronous operation

**equal(actual, expected, message):** A non-strict comparison, roughly equivalent to JUnit's `assertEquals`

**expect(asserts):** Specify how many assertions are expected to run within a test

**notEqual(actual, expected, message):** A non-strict comparison, checking for inequality

**deepEqual(actual, expected, message):** A deep recursive comparison, working on primitive types, arrays, objects ...

**notDeepEqual(actual, expected, message):** An inverted deep recursive comparison, working on primitive types, arrays ...

**ok(result, message):** A Boolean check, equivalent to CommonJS's `assert.ok()` and JUnit's `assertTrue()`.

**notOk(result, message):** A Boolean check, inverse of `ok()` and CommonJS's `assert.ok()`, and equivalent to JUnit's

**assertFalse().** Passes if the first argument is falsy

**strictEqual(actual, expected, message):** A strict type and value comparison

**notStrictEqual(actual, expected, message):** A strict comparison, checking for inequality

**propEqual(actual, expected, message):** A strict type and value comparison of an object's own properties

**notPropEqual(actual, expected, message):** A strict comparison of an object's own properties, checking for inequality

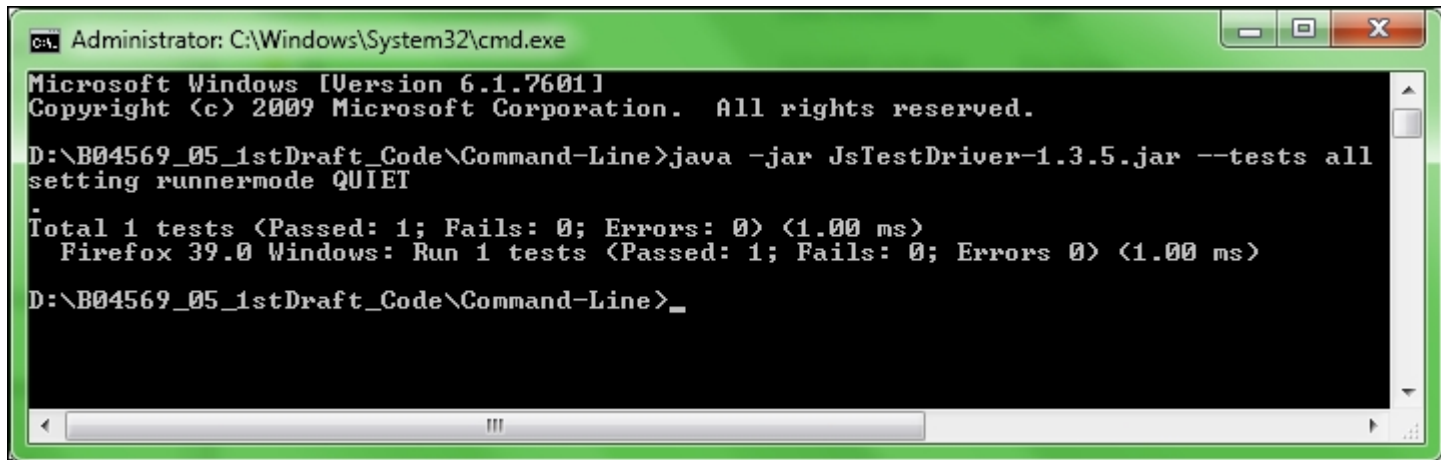
**push():** Report the result of a custom assertion

**throws():** Test if a callback throws an exception, and optionally compare the thrown error

# Exemple avec jsWebDriver

```
HelloWorldTest = TestCase("HelloWorldTest");
HelloWorldTest.prototype.testSay = function() {
    var helloWorldApp = new myapp.HelloWorldApp();
    assertEquals("Hello Reader!",
        helloWorldApp.say("Reader"));
};
```

```
myapp = {};
myapp.HelloWorldApp = function() { };
myapp.HelloWorldApp.prototype.say =
function(name) {
    return "Hello " + name + "!";
};
```



A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\System32\cmd.exe". The window shows the output of running the JsTestDriver 1.3.5 jar file. The command executed is `java -jar JsTestDriver-1.3.5.jar --tests all setting runnermode QUIET`. The output indicates that 1 test passed successfully in 1.00 ms. The test is for Firefox 39.0 Windows, running 1 test which also passed successfully in 1.00 ms. The command prompt is at the directory `D:\B04569_05_1stDraft_Code\Command-Line`.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

D:\B04569_05_1stDraft_Code\Command-Line>java -jar JsTestDriver-1.3.5.jar --tests all
setting runnermode QUIET

Total 1 tests <Passed: 1; Fails: 0; Errors: 0> <1.00 ms>
  Firefox 39.0 Windows: Run 1 tests <Passed: 1; Fails: 0; Errors 0> <1.00 ms>

D:\B04569_05_1stDraft_Code\Command-Line>
```

# Analyse statique et javascript

- Analyse de style
  - jsLint en ligne sur [www.jslint.com](http://www.jslint.com)
  - Node.js : jsHint et jsLint
- Analyse de couverture
  - LCOV , jsCoverage
- Exemple avec jsWebDriver + Lcov + perl + genhtml

LCOV - code coverage report

Current view: top level

Test: jsTestDriver.conf-coverage.dat

Date: 2015-07-11 12:16:22

Hit

Total

Coverage

Lines: 709123357.5 %

Functions: 00-

Directory	Line Coverage ↕		Functions ↕	
lib	<div><div></div></div>	33.3 %	1 / 3	- 0 / 0
lib/adapter	<div><div></div></div>	64.8 %	59 / 91	- 0 / 0
lib/jasmine-1.1.0	<div><div></div></div>	54.6 %	578 / 1058	- 0 / 0
src	<div><div></div></div>	77.3 %	34 / 44	- 0 / 0
src-test	<div><div></div></div>	100.0 %	37 / 37	- 0 / 0

Generated by: LCOV version 1.0



# Intégration continue et javascript

- Lanceurs de tâches pour javascript
  - Grunt (Le plus célèbre sous nodeJS)
  - Cake (CoffeeScript)
  - Gulp (Plus simple que Grunt)



- Il est facile de coupler un serveur IC et un lanceur javascript
  - Jenkins + Grunt
  - Jenkins + Gulp

- Exemple
  - Jenkins + Grunt + jsWebDriver



# Tester les interfaces graphiques web réelles

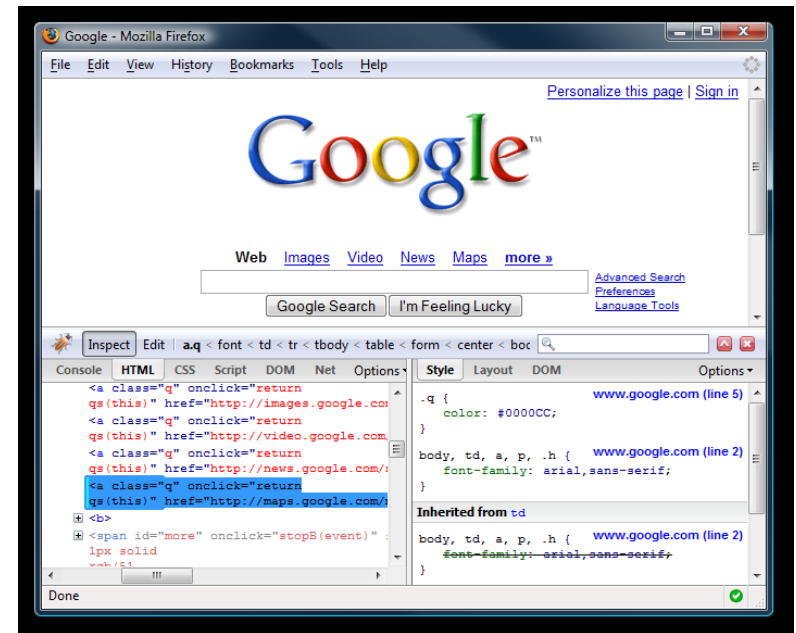
- Nécessite de piloter le navigateur !
- Plusieurs possibilités :
  - **Robot externe + plugins navigateurs**
    - Ranorex + Ranorex Firefox Plugin
    - TestComplete + TestComplete Chrome Plugin
  - **Robot interne + javascript**
    - Selenium IDE + Selenium Core
  - **Code + Driver (WebDriver) pour chaque navigateur**
    - ChromeDriver pilotable par l'API Web Driver
    - CHAI + Web Driver + ChromeDriver
    - Jasmine + WebDriverIO + ChromeDriver
  - **Framework javascript spécialisé**
    - DalekJS
  - **Faire son propre framework en javascript ou avec un langage supportant l'API Web Driver**



- **IMPORTANT** : API **Selenium Web Driver** est actuellement examinée par le W3C pour devenir l'**API officielle d'automatisation des navigateurs**.

# La problématique : l'identification des composants

- Comment identifier les composants HTML pour les retrouver sur une page web ?
  - Attribut id
  - Attribut name
  - Attribut class
  - Type de tag HTML
  - Position dans le DOM (XPath et CSS)
  - Valeurs d'attributs (XPath et CSS)
  - Texte d'un hyperlien
- Utiliser les outils de développement des navigateurs, firebug, firexpath pour examiner les composants et tester les requêtes CSS et XPath



# Exemples XPath

- `driver.findElement(By.xpath("/html/body/div/div/form/input"));`
- `driver.findElement(By.xpath("//input[2]"));`
- `driver.findElement(By.xpath("//input[@id='username']"));`
- `driver.findElement(By.xpath("//img[@alt='Previous']"));`
- `driver.findElement(By.xpath("//input[@type='submit' and @value='Login']"));`
- `driver.findElements(By.xpath ("//img[not(@alt)]"));`
- `driver.findElement(By.xpath("//td[contains(text(),'Item 1')]"));`

## Fonctions XPath

- `input[starts-with(@id,'ctrl')]`
- `input[ends-with(@id,'_userName')]`
- `Input[contains(@id,'userName')]`
- `/table/tr[1]` , `/table/tr[last()]` , `/table/tr[last()-1]` , `/table/tr[position()>4]` , `//tr[td>40]`
- \* n'importe quel élément, @ n'importe quel attribut , `node()` noeuds

# Exemples CSS

- `driver.findElement(By.cssSelector("input"));`
  - `driver.findElement(By.cssSelector("input.login"));` // class login
  - `driver.findElement(By.cssSelector("input#username"));`
  - `driver.findElement(By.cssSelector("input[name=username]"));`
  - `driver.findElement(By.cssSelector("input[type='submit'][value='Login']"));`
  - `driver.findElements(By.cssSelector("img:not([alt])"));`
  - `driver.findElements(By.cssSelector("div, p"));`
  - `driver.findElement(By.cssSelector("td:contains('Item 1')"));`
- 
- `input[id^='ctrl']` – commence par
  - `input[id$='_userName']` – se termine par
  - `Input[id*='userName']` – contient
  - `form#loginForm:first-child` , `form#loginForm:last-child` , `:nth-child(2)`

# Problème des IDs dynamiques

- A chaque chargement d'une page HTML, certaines bibliothèques de composants graphiques écrits en javascript (Dojo, JQuery UI, Sencha Ext JS, ...) génèrent des ids dynamiquement (ext-gen-123, treepicker-1038, ...)
- 2 Stratégies
  - Ignorer les ID dynamiques et opter pour un autre mode de reconnaissance de composants
  - Traiter les ID dynamiques en reconnaissant une partie stable
- Les robots ou les frameworks proposent des assistants
  - Règles de reconnaissance des IDs
  - Expressions régulières ou traitement de chaînes (début, fin, milieu)
  - Système interne de reconnaissance de composants
  - Exemple : XPath pour `<input id=« text-12345 »/>` -> `//*[@id, 'text-']`

# Tests fonctionnels côté client

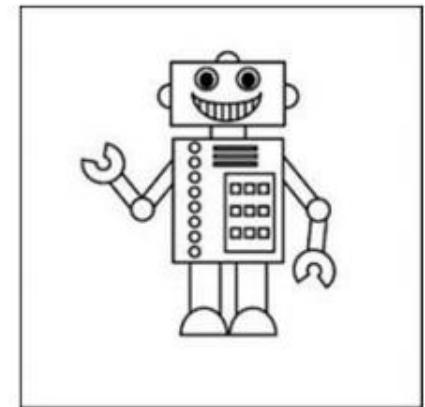
- Le test fonctionnel dans les navigateurs avec le paramétrage prévu en production
  - **Reproduction exacte des actions des utilisateurs**
  - **Système d'injection rapide** de données dans les composants
  - **Outils :**
    - Robot de tests
    - Bibliothèque de pilotage des navigateurs, comme Selenium, couplée à une bibliothèque xUnit ou équivalent
    - Eventuellement un **outil présentant un plus haut niveau d'abstraction**, permettant de piloter les scripts de tests dans un **environnement métier**

# Principaux robots de tests

- Produits commerciaux les plus connus

Complets pour le test fonctionnel et faisant partie d'une suite intégrée :

- IBM Rational Functional Tester
- HP Unified Functional Testing (QTP)
- Microfocus/Compuware TestPartner
- Microfocus/Borland Silk Test
- SmartBear – AutomatedQA TestComplete
- Ranorex – Ranorex Software
- EggPlant – Reconnaissance d'images



- Produits open source

- Web : Watir ( ruby ) , Watij ( java ) , Selenium IDE , Sahi
- Sikuli : technique de reconnaissance d'images rapide
- Auto-It : reconnaissance de composants Windows





# Fonctions proposées par les robots

- **Enregistrement des actions** : scripts ou listes de mots clé
- Enregistrement des **oracles de validations**
- Environnement d'organisation des tests
- Environnement de mise au point des tests et **développement**
- Lancement des tests : machine locale, machine réseau ou cloud
- Système de log et de rapport de test
- Assistants divers pour injecter les données et « variabiliser » les constantes, reconnaître les composants
- **Demande en général de bonnes compétences en développement**



# Exemple : Ranorex Actions

The screenshot displays the Ranorex Studio interface with a test script titled "AddEntry.rxrec". The script is in RECORD mode and contains 9 actions. The first action is a Mouse Click on the "EditViewEntry" repository item. The subsequent actions are Mouse Clicks on "AddEntry", "Text", "MBtnIcon", "ListItem1", "MBtnOK", and "UserName", followed by a Key Sequence for "\$varUsername".

#	Action	Button	Action Spot	Repository Item	
1	Mouse	Click	Left	66;7	EditViewEntry
2	Mouse	Click	Left	Relative	AddEntry
3	Mouse	Click	Left	Relative	Text
4	Key Sequence			\$varTitle	Text
5	Mouse	Click	Left	Relative	MBtnIcon
6	Mouse	Click	Left	Relative	ListItem1
7	Mouse	Click	Left	Relative	MBtnOK
8	Mouse	Click	Left	Relative	UserName
9	Key Sequence			\$varUsername	UserName

The bottom pane shows the "Item" tree with the following structure:

- KeyPromptForm
- MainForm
  - EditViewEntry (selected)
  - WordPressDemo
  - List
  - Row0
  - Save
  - Close
- KeePass

The "Path" column for the selected "EditViewEntry" item shows the following path:

```
Base: /form[@controlname='KeyPromptForm']  
Base: /form[@controlname='MainForm']  
??/menuItem[@accessiblename='Edit']/menuItem[@acces...  
container[@controlname='m_splitHorizontal']/?/?/containe...  
container[@controlname='m_splitHorizontal']//table[@con...  
container[@controlname='m_splitHorizontal']//table[@con...  
??/button[@accessiblename='Save']  
??/button[@accessiblename='Close']  
Base: /contextmenu[@processname='KeePass']
```

The right pane shows a screenshot of the KeePass Password Safe application window, highlighting the "File", "Edit", "View", "Tools", and "Help" menu items.

# Example : Ranorex Keyword Driven Testing

Ranorex Module Browser

Search... (F3)

- Keyword Driven Testing
  - Groups
  - Modules
    - CloseSUT
    - Login
      - varPassword
    - StartSUT
      - varExecutionPath

Keyword Driven Testing.rxtst

NEW RUN TestRun MANAGE DATASOURCES...

Search... (F3)

Item	Data Binding	Description
<input checked="" type="checkbox"/> <b>Keyword Driven Testing - Test Suite</b>		
<input checked="" type="checkbox"/> <b>TestCase</b>	Credentials - rows: 3	
<input checked="" type="checkbox"/> StartSUT	Bound variable: 1	
<input checked="" type="checkbox"/> Login	Bound variable: 1	
<input checked="" type="checkbox"/> CloseSUT		

Projects

KeywordLibrary.cs

NEW RUN

Keyword\_Driven\_Testing.KeywordLibrary

CloseSUT()

```
50 public void StartSUT(string executionPath)
51 {
52     Host.Local.RunApplication(executionPath);
53 }
54
55 public void Login(string password)
56 {
57     Text mTbPassword = "/form[@controlname='KeyPromptForm']//text[@controlname='m_tbPassword']";
58     mTbPassword.TextValue = password;
59     Button mBtnOK = "/form[@controlname='KeyPromptForm']/button[@controlname='m_btnOK']";
60     mBtnOK.Click();
61 }
62
63 public void CloseSUT()
64 {
65     Form mainForm = "/form[@controlname='MainForm']";
66     Host.Local.CloseApplication(mainForm);
67 }
```

# Exemple : Ranorex User Code, Variables, Data Sources

The screenshot illustrates the configuration of a Ranorex test. The main window shows a test action table with two actions: 'User Code' and 'User Code'. The second 'User Code' action is selected, and a context menu is open, showing the method 'Login(String password)' highlighted. The 'password' variable is linked to the 'password' column in the table.

The 'Manage Data Sources' dialog is open, showing the 'Excel Connector...' option selected. The 'Data Source Config' section is empty.

The Excel data source is 'Passwords.xlsx - Excel', showing a table with the following data:

	A	B	C	D
1	excel_Title	excel_Username	excel_Password	excel_URL
2	WordPressDemo	admin	demo123	<a href="http://bitly.com/wp_demo">http://bitly.com/wp_demo</a>
3	Gmail	<a href="mailto:test@gmail.com">test@gmail.com</a>	password	<a href="https://mail.google.com">https://mail.google.com</a>

# Les bibliothèques d'automatisation des navigateurs

- Objectifs

- Fournir une API de pilotage des navigateurs
- **API de reconnaissance des composants en lecture/écriture**
- API de déclenchement d'évènement utilisateurs
- API de réglage des propriétés du navigateur
- API de gestion de la synchronisation des actions de tests
- API de gestion de javascript, ajax, cookie, boîte de dialogue

- Les principales bibliothèques d'automatisation

- Selenium Web Driver (Java, Python, Ruby, C#, Javascript)
- Webdriverio + selenium web driver
- DalekJS – pilotage natif en javascript
- Bibliothèque de pilotage des composants prévue dès la conception
  - Exemple avec Sencha Ext JS et SenchaTest



# Les surcouches

- Objectifs des surcouches
  - WebDriverJS (Selenium)
  - Protractor (Selenium)
  - Capybara (Selenium)
  - Google Robot Frameworks (Selenium, ...)
- Couplage éventuel avec une bibliothèque de test, d'assertions
  - CHAI.js
  - Should.js
  - Assert.js
- Frameworks de test de plus haut niveau pour exprimer le comportement attendu
  - Karma
  - Mocha
  - Jasmine
- **Exemples courants :**
  - Mocha + CHAI.js + Protractor
  - Jasmine + WebDriverIO



# Un exemple avec Jasmine

- Utile pour les applications modernes « javascript centric »
- Possibilité de tester toutes les parties des applications côtés client ET serveur
- Applications : unitaires, intégrations et validation

```
var webdriver = require('selenium-webdriver');

var driver = new webdriver.Builder()
    .withCapabilities(webdriver.Capabilities.chrome())
    .build();

describe('basic test', function () {
    it('should be on correct page', function (done) {
        driver.get('http://www.wingify.com');
        driver.getTitle().then(function(title) {
            expect(title).toBe('Wingify');
            // Jasmine waits for the done callback to be called before proceeding to next
            // specification.
            done();
        });
    });
});
```

Exemple d'utilisation de  
Jasmine avec WebDriverJS  
pour Selenium

# Les surcouches

- Objectifs des surcouches : simplifier les API, abstraire les commandes

- WebDriverJS (Selenium)
- Protractor (Selenium)
- Capybara (Selenium)
- Google Robot Frameworks

```
it 'should reject birthdates of users under 21 years old' do
  birth_date = '20000123'
  fields(:birth_date => birth_date).valid_fields?(:birth_date).should be_false
  fields(:birth_date => birth_date).errors[:birth_date] == ["under-21"]
end

it 'should reject birthdates before 1900' do
  birth_date = '18990123'
  fields(:birth_date => birth_date).valid_fields?(:birth_date).should be_false
  fields(:birth_date => birth_date).errors[:birth_date] == ["before-1900"]
end
```

- Couplage éventuel avec une bibliothèque de test, d'assertions

- CHAI.js
- Should.js
- Assert.js
- Expect.js

- Frameworks de test de plus haut niveau pour exprimer le comportement attendu

- Karma
- Mocha
- Jasmine

- **Exemples courants :**

- Mocha + CHAI + Protractor
- Jasmine + Protractor

Create expressive integration tests with `chai` and `selenium-webdriver`.

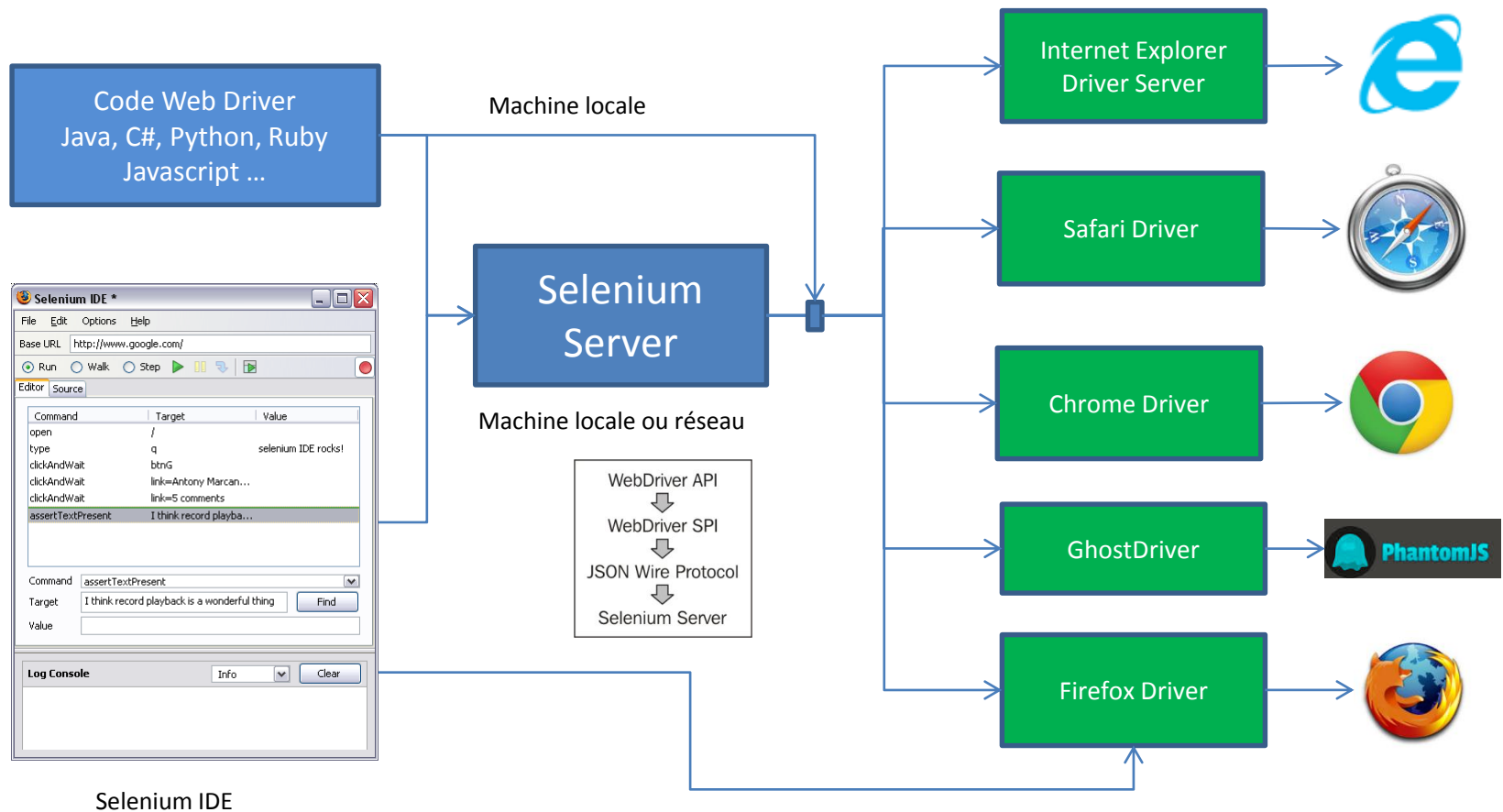
```
chai.use(chaiWebdriver(driver));
driver.get('http://chaijs.com/');
expect('nav h1').dom.to.contain.text('Chai');
expect('#node .button').dom.to.have.style('float', 'left');
```



# Maturité de l'automatisation

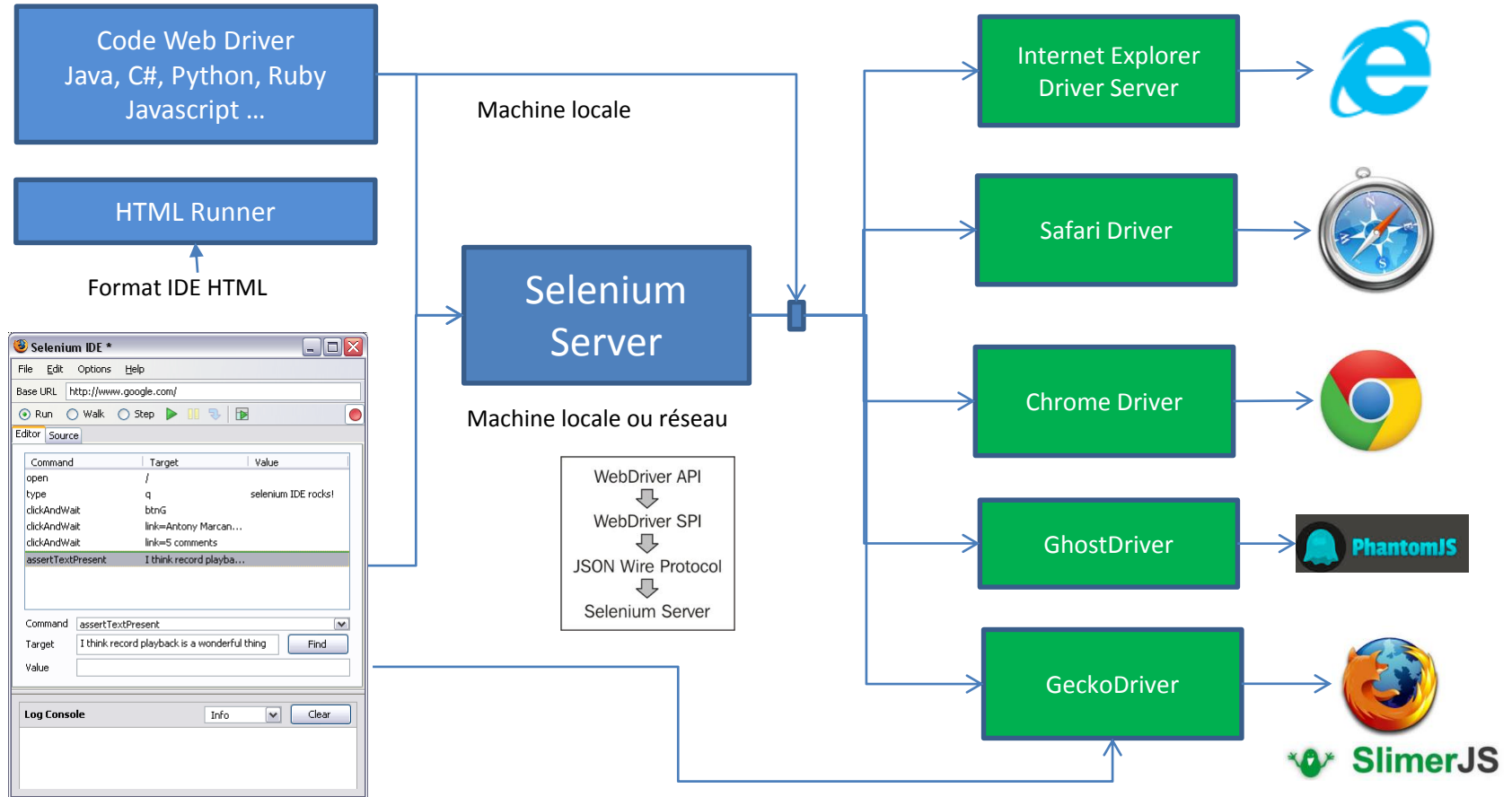
- **Couplage avec les xUnits ou des systèmes d'assertion**
- **Reproduction exacte des actions utilisateurs**
- **Niveau d'abstraction**
  - 1 : Code linéaire
  - 2 : Code linéaire et organisé dirigé par des données externes
  - 3 : Mots clés indépendants du code permettant de piloter les tests au niveau métier :
    - Couche d'abstraction
    - Couche intermédiaire
    - Couche technique
  - 4 : Code organisé et factorisé

# Architecture Selenium 2.0



Les scripts selenium IDE peuvent être joués sur d'autres navigateurs que firefox.

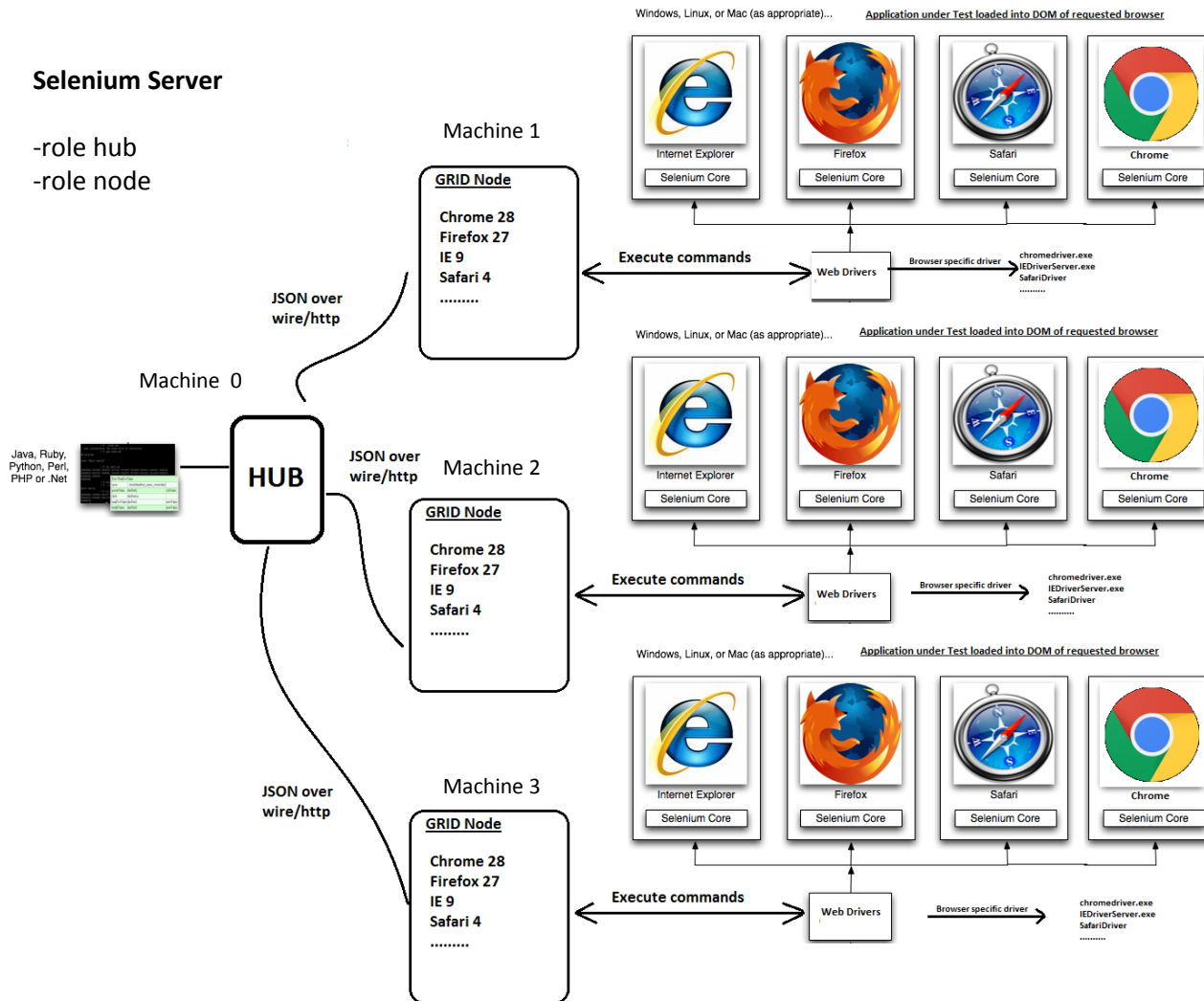
# Architecture Selenium 3.0



Selenium IDE

Les scripts selenium IDE peuvent être joués sur d'autres navigateurs que firefox via Selenium Server ou via les langages et l'API WebDriver 3.0.

# Architecture Selenium GRID



# Tests et RIA propriétaires

- La messe est dite : HTML 5 est vainqueur
- Nombre important d'applications RIA à maintenir :
  - Tous les robots commerciaux propose des extensions flash/flex silverlight et JavaFX
  - Quelques robots spécialisés
    - RIATest
  - Des produits spécialisés
    - FlexMonkey et RIATest
    - Gwt-tests-utils
    - Sahi Pro
  - Il existe des extensions pour Selenium
    - FlexMonkium, Flex-ui-selenium, flex-pilot-x

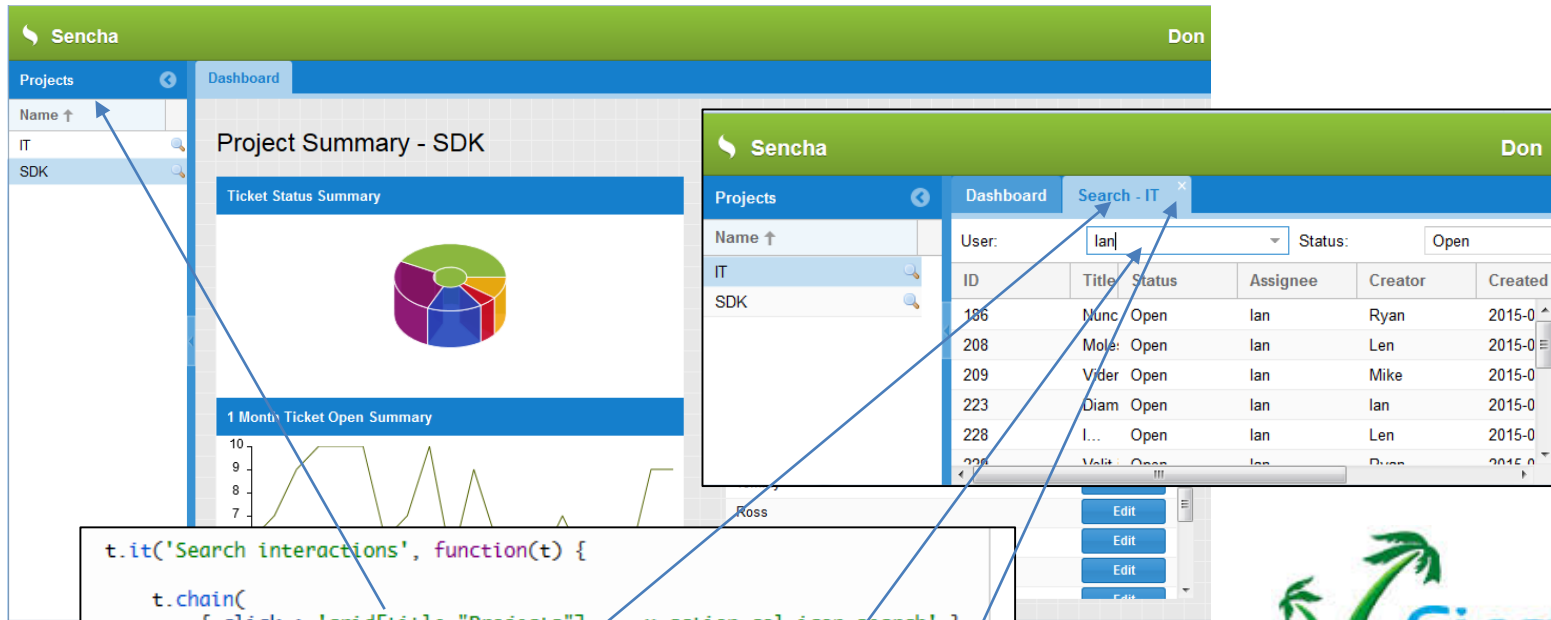


# Tests et bibliothèques javascript RIA

- Les composants riches sont écrits 100% en javascript + CSS + [HTML 5]
- Exemples de bibliothèques : **jQuery UI, Ext JS, YUI, Dojo**
- Les éditeurs sont conscients des problèmes pour tester efficacement leurs composants
  - Problèmes des id dynamiques et de la profondeur de l'arbre DOM.
- Les éditeurs **proposent des accesseurs intelligents** sur leurs composants pour pouvoir les tester :
  - Exemple avec **Ext JS** et **Ext.ComponentQuery**
  - Utilisation du **Web Driver javascriptExecutor** pour utiliser ces méthodes
- Certains « frameworks » de tests utilisent ces objets intelligents pour proposer des méthodes de test, d'attentes, et d'actions sur ces composants
- Exemple avec Siesta et Sencha Ext JS




# Tests de composants Ext-JS 5.0 avec Siesta



The screenshot shows a Sencha application with a dashboard. The dashboard has a 'Projects' menu on the left and a 'Search - IT' window on the right. The 'Search - IT' window displays a table of search results. The code in the foreground is a Sencha test script that interacts with the application components.

```
t.it('Search interactions', function(t) {  
    t.chain(  
        { click : 'grid[title="Projects"] => .x-action-col-icon.search' },  
        { waitForCQ : 'tabpanel grid[title^=Search]', desc : 'Should find Search grid' },  
        { waitForRowsVisible : '>>grid[title^=Search]', desc : 'Search grid' },  
        { click : '>>combobox[fieldLabel="User"]' },  
        function (next) {  
            var gridview = t.cq1('>>grid[title^=Search]').getView()  
            t.firesOnce(gridview, 'refresh');  
            t.cq1('combobox[fieldLabel="User"]').select(2)  
            next();  
        },  
        { click : 'tab[text^=Search] => .x-tab-close-btn' },  
        { waitForCQNotFound : 'tabpanel grid[title^=Search]', desc : 'Should not find Search grid' },  
    ),  
})
```



Siesta propose donc une série d'ordres et de contrôles sur les composants en utilisant l'objet ComponentQuery de Ext-JS

# Spécifications par l'exemple

- Prenons de la hauteur
  - Un exemple est un test potentiel
  - Un test est un exemple potentiel
  - **A specification is a test, a test is a specification**
- Pourquoi ne pas utiliser **le même outil pour décrire le fonctionnement de l'application et décrire ses tests** ?
  - Un wiki accessible à tous
  - Une documentation vivante
  - Des spécifications parfois exécutables si on développe une glue vers les robots ou les scripts d'automatisation
- Outils
  - Fitnesse, cucumber, concordion





⚠ **Test Pages:** 0 right, 0 wrong, 0 ignored, 1 exceptions **Assertions:** 6 right, 0 wrong, 0 ignored, 2 exceptions (1,466 seconds)

## Test System: slim:fitness.slim.SlimService

variable defined: TEST\_SYSTEM=slim

classpath: c:\plateforme-fitnessse\classes\

Les spécifications sont les suivantes :

- La note de cours est un **entier** compris entre 0 et 25
- La note d'examen est un **entier** compris entre 0 et 75

Le système calcule la **somme** des notes de cours et d'examen et renvoie une note parmi les lettres A B C D

- somme < 30 -> D
- somme >=30 et somme < 50 -> C
- somme >=50 et somme < 75 -> B
- somme >=70 -> A

Les tests doivent porter sur les limites et les cas négatifs

```
import
```

```
tests
```

### Test Examen

cours	examen	note?
20	49	B
20	50	A
20	30	B
20	29	C
20	9	D
20	10	C
26	10	Valeur hors bornes ⚠ Valeur hors bornes
20	AA	Mauvais format ⚠ Mauvais format