

Selenium

Tests fonctionnels d'une application web

Simon Brandhof
16.01.2007



sous contrat Creative Commons
Paternité – Pas d'utilisation commerciale

Agenda

- **le monde merveilleux des tests**
tests unitaires, fonctionnels et de performance
- **selenium**
principes, démo
- **cycle de développement**
environnements, maven 2, intégration continue
- **conclusion**

Le monde merveilleux des tests

« le test est au développement ce que le fromage est à la fondue », Jacques Couvreur

Les tests unitaires

→ **Contrat d'une classe**

- Plus petite partie testable de l'application

→ **Principe fondamental du développement agile**

- Valide le bon fonctionnement d'une classe
- Vérifie la non régression
- Facilite le refactoring
- Facilite le développement collaboratif (« collective code ownership »)
- Fait office de documentation (fiable et à jour si tests automatisés)
- Séparation de l'interface et de l'implémentation (mock objects)

→ **Test Driven Development**

Les tests fonctionnels

→ ~ « integration test » « acceptance test »

→ **Valider**

- le fonctionnement de l'application
- la non régression d'une release
- la correction d'un defect

→ **Principe de boîte noire**

→ **Une des pratiques de l'eXtreme Programming**

- créés à partir des user stories
- Automatisation dans l'intégration continue
- Par le développeur, avec l'aide du client ou du service Qualité
- Calcul de la vélocité
- Tested Feature Metric de Ron Jeffries pour calculer l'avancée du projet

Les tests fonctionnels

→ **Les outils open source**

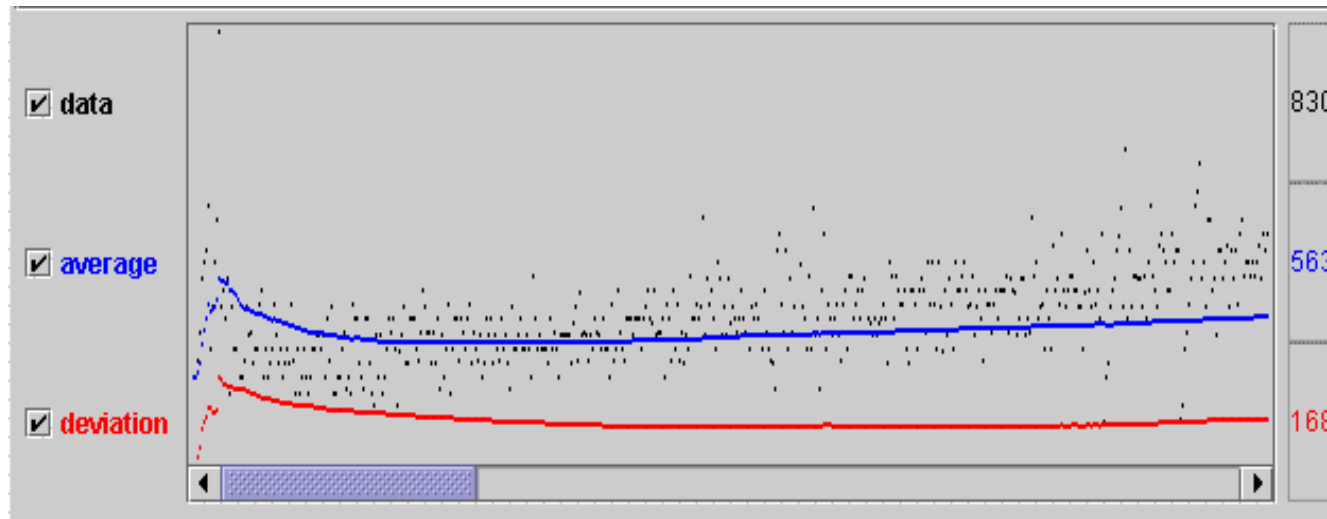
- Junit, HttpUnit, Cactus
- Jmeter
- **Selenium**, FIT, **FITnesse**, Watir, Sahi

→ **Les autres...**

- La suite Mercury
- Automated QA TestComplte

Les tests de performance

- **Mesurer la vitesse d'exécution et la montée en charge**
- **Outils open source**
 - Apache Jmeter (HTTP, FTP, JDBC, JMC, ...)



- JunitPerf pour les tests unitaires

Selenium

« comment ça marche ? », Jacques Couvreur

Selenium

→ Initié par ThoughtWorks

- 750 collaborateurs dans 6 pays
- Pionniers de l'agilité (Martin Fowler) et de l'open source (cruisecontrol)

→ Objectifs

- Tests fonctionnels d'applications web (HTML ou XML sur HTTP)
- Tests de compatibilité entre navigateurs

→ Open source (license Apache 2.0)

→ Projet actif, bien documenté (forum, wiki)

→ Trois modules

- Selenium core
- Selenium remote control
- Selenium IDE, plugin Firefox

Selenium core

→ Moteur en Javascript/HTML

- Les tests s'exécutent directement dans le navigateur
- Compatible avec Firefox, IE, Opera, Safari...

→ Interface d'administration

- Visualisation des scénarii
- Exécution des tests
- Débogueur (pas à pas, breakpoints)
- Affichage des logs
- Affichage du DOM de la page testée

Selenium core

The screenshot displays the Selenium TestRunner interface within a browser window. The browser address bar shows the URL: `chrome://selenium-ide/content/selenium/TestRunner.html?test=/conten`. The interface is divided into several sections:

- Test Suite:** A sidebar on the left with a "Playback" button.
- Test Player:** A table listing test steps and their corresponding actions.
- Selenium TestRunner:** A control panel on the right with execution buttons, a speed slider (Fast to Slow), a "Highlight elements" checkbox, and a status summary.

Test Player	Action
open	/valeurs.php
verifyTextPresent	Bienvenue
clickAndWait	agi
assertTextPresent	Individuals and interaction
click	dev
assertTextPresent	Nous maîtrisons les technologies suivantes
click	cons
verifyTitle	Hortis - Conseil

The Selenium TestRunner control panel includes the following details:

- Execute Tests:** Play, Stop, Refresh, and Reload buttons.
- Speed:** A slider between "Fast" and "Slow".
- Highlight elements:** A checkbox that is currently unchecked.
- Elapsed:** 00:02
- Tests:** 0 run, 0 failed, 0 incomplete.
- Commands:** 1 passed, 0 failed, 0 incomplete.
- Tools:** "View DOM" and "Show Log" buttons.

The web page being tested is the "hortis" website. It features a logo with the tagline "La rencontre de l'AGILITÉ et de l'OPEN-SOURCE". A navigation menu includes: "Nos valeurs", "Méthodologies Agiles", "Développement", "Conseil", "Offres d'emploi", and "Contacts". The main content area displays "Bienvenue" and a paragraph: "Hortis est spécialisé dans le développement d'applications J2EE par la mise en oeuvre de Méthodologies Agiles (Scrum / eXtreme Programming) et l'intégration de produits Open-Source." A decorative graphic of a tree is visible on the left side of the page.

Selenium core

- Exécute des actions comme cliquer ou saisir du texte
- Vérifie le contenu de la page (assertions)
- Le scénario est écrit en « Selenese »
 - Syntaxe très simple : tableau HTML à 3 colonnes (commande + 2 arguments)

MyTest		
open	/mypage	
type	nameField	John Smith
click	submitButton	True
verifyText	name	John Smith

Selenium core – les actions

→ **Tout ce que peut faire l'utilisateur**

- Presser/relâcher une touche, saisir du texte
- Simple/double clic de la souris
- Drag and drop (Ajax)
- Mouvement de souris
- Sélectionner une pop-up ou une frame
- Edition/suppression de cookies

→ **Notion d'attente d'événements (avec gestion d'un timeout)**

- `waitForXXX()`, `waitForCondition(javascript)`
- Pratique pour les applications Ajax et le chargement des frames

Selenium core – les assertions

→ Les assertions vérifient l'état de l'application

- « le titre de la page est X »
- « la checkbox Y est cochée »

→ Deux niveaux

- assertXXX (arrête le test)
- verifyXXX (log et continue le test)

→ Les accessors mémorisent la valeur d'une expression

- pour être réutilisée plus tard dans une action ou une assertion :

```
storeValue nameField firstName  
storeEval 'Mr' title  
assertTextPresent ${title} ${firstName}
```

Selenium core – les assertions

→ Une expression peut être

- Un « element locator » pour retrouver un élément de la page
- Une expression régulière (de type glob ou javascript)

Selenium core – les assertions

- **id=x**
 - Sélectionne l'élément avec l'attribut @id.
- **name=x**
 - Sélectionne le premier élément avec l'attribut @name.
- **identifiant=x**
 - id=x sinon name=x
- **dom=javascriptExpression**
 - dom=document.forms['myForm'].myDropdown
- **xpath=xpathExpression**
 - xpath=//table[@id='table1']//tr[4]/td[2]
- **link=textPattern**
 - Sélectionne le lien qui contient le texte spécifié
- **css=cssSelectorSyntax**
 - css=a[href="#id3"]

Selenium core – les inconvénients

→ **Cross-Site Scripting (XSS)**

- Selenium (HTML + JS) doit être déployé sur le même domaine que l'application testée

→ **Les navigateurs doivent être correctement configurés**

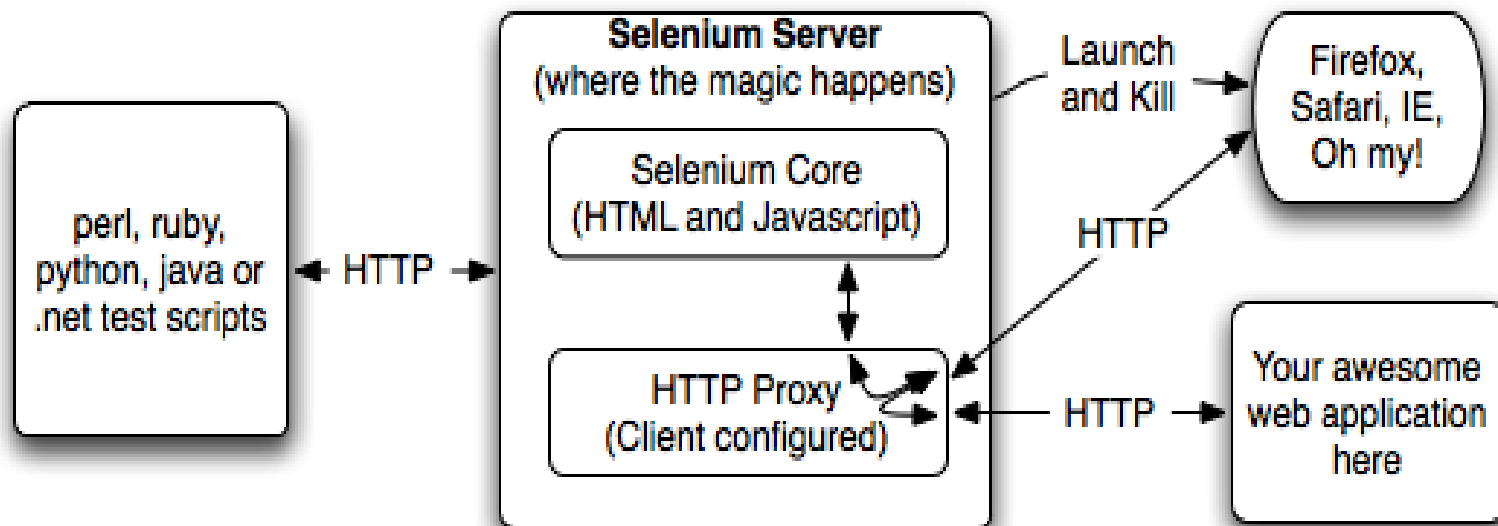
- Profil potentiellement différent avant chaque exécution de test
- cookies existants
- mots de passe ou données de formulaires sauvegardés

→ **Difficile à automatiser**

- Lancement du navigateur
- Rapport d'exécution non exportable

Selenium remote control

→ Application Java 5 (« Selenium Server ») qui contrôle le navigateur



→ Deux modes

- automatique : lecture de scénarii au format Selenese
- interactif : scripting des scénarii en Java, Junit, Ruby, Python, .NET...

Selenium remote control

- **Démarrage du navigateur avec un profil vierge**
- **Pas de problème de XSS (proxy HTTP)**
- **Ant / maven 2**
 - Génération de rapports HTML
 - Pas d'installation de selenium-core dans le serveur applicatif
- **Automatisable pour l'intégration continue**
- **Les tests en Java ou Ruby plus complets que ceux en Selenese**
 - Réutilisation de code
 - Accès à des données inexistantes dans le navigateur (EJB, JDBC, ...)

Selenium remote control – maven 2

→ **src/test/selenium**

- Tests au format Selenese (ex. AdminRoleTest.html)
- Regroupement des tests avec le TestSuite (ex WebmailTestSuite.html)

→ **Profils maven2**

- Configuration (URL du serveur de test, ...)
- Activation des tests fonctionnels dans le cycle de vie : mvn install
- Sinon mvn integration-test

→ **Nécessite Java 5**

Selenium IDE

- **Plugin Firefox**
- **Enregistrement d'un scénario pouvant être exécuté dans n'importe quel autre navigateur**
- **Exécution d'un test directement dans firefox**
 - Moteur selenium dans le noyau de firefox (chrome)
 - Pas de problème de Cross-site Scripting
 - Debugger
- **Export des tests en Selenese (par défaut), Java, Ruby et autres**

Selenium IDE

The screenshot displays the Selenium IDE interface for a test script titled "TestPopUp.html". The Base URL is set to "http://baobab". The interface includes a menu bar (File, Edit, Options, Help), a toolbar with Run, Walk, Step, and other execution controls, and a table of test commands. Below the table is a configuration panel for the selected "type" command, and a Log/Reference section at the bottom.

Command	Target	Value
open	/portal/server.pt?space=Login&cached=tru...	
type	id	DBIZ_brand...
type	in_pw_userpass	password
clickAndWait	name=in_bu_Login	
open	/portal/server.pt?space=CommunityPage&c...	
open	/portal/server.pt?space=CommunityPage&c...	
assertElementPresent	link=OPEN MY MAIL BOX	
click	link=OPEN MY MAIL BOX	
waitForPopUp	webmail	30000
selectWindow	webmail	
selectFrame	webMailTop	
assertTextPresent		e-services

Command: type
Target: id
Value: DBIZ_brandhof

Log Reference
type(locator, value)
Arguments:

- ♦ locator - an element locator
- ♦ value - the value to type

Sets the value of an input field, as though you typed it in.

Cycle de vie logiciel

Environnements

→ Développement

- Propre au développeur
- Instable
- Etat inconsistant

→ Intégration

- Partagé par toute l'équipe
- Mis à jour par l'intégration continue
- Données consistentes
- Validé par les tests fonctionnels

→ Pré-production

→ Production

Maven 2

→ Plugin selenium

- Exécute le mode automatique de Selenium Remote Control
- `mvn integration-test`

```
<dependency>  
  <groupId>org.openqa.selenium.server</groupId>  
  <artifactId>selenium-server</artifactId>  
  <version>0.9.1-SNAPSHOT</version>  
  <scope>test</scope>  
</dependency>
```

→ Intégration du rapport d'exécution dans le site maven

Selenium on rails



→ Configuration (config.yml)

```
browsers:  
  firefox: 'c:\Program Files\Mozilla Firefox\firefox.exe'  
  ie: 'c:\Program Files\Internet Explorer\iexplore.exe'
```

→ Exécution

```
./script/server -e test  
rake test:acceptance
```

Conseils

- **Enregistrer/exécuter les tests sur l'environnement d'intégration**
- **KISS (Keep It Simple and Stupid)**
 - Enregistrer les tests avec Selenium IDE au format Selenese (HTML)
 - Nombreux tests simples et de petite taille, plutôt que quelques tests complexes
- **Dépendance minimale avec le design et les libellés**
- **Si possible, prendre le temps d'écrire un test lors de la correction d'un defect**

Conseils

- **Abuser des paramètres id et name dans le HTML**
 - Sur une portlet pour l'identifier dans une page du portail
 - Sur les liens, formulaires, boutons, ...
- **Utiliser les regexp si plusieurs valeurs possibles**
 - `verifyText | //div[@id='user'] | regexp:Jacques|Freddy|Simon`
- **Installer selenium-core en local**
 - Exemples complets et exhaustifs
- **S'aider de plugins Firefox pour l'enregistrement des tests**
 - XPath Checker
 - Xpather

Conseils

→ Générer des valeurs uniques pour certaines données de formulaire

```
<tr>
  <td>store</td>
  <td>javascript{'subject' + (new Date()).getTime()}</td>
  <td>subject</td>
</tr>
<tr>
  <td>type</td>
  <td>subject</td>
  <td>${subject}</td>
</tr> .....
<tr>
  <td>assertTextPresent</td>
  <td></td>
  <td>${subject}</td>
</tr>
```

Conclusion

Conclusion

→ **Ce n'est pas une perte de temps**

- Les tests sont de + en + rapides à écrire
- Vous validez qu'un bug reste corrigé
- Vous validez qu'une fonctionnalité/correction de bug est bien incluse dans une nouvelle release

→ **Mais...**

- Demande un effort initial (apprentissage, environnement d'intégration)
- Ne correspond pas à tous les besoins
- Couplage avec Junit, HttpUnit, ...

Documentation

→ **Open QA**

- <http://www.openqa.org/selenium/>
- Toutes les commandes sur <http://www.openqa.org/selenium-core/reference.html>
- Forum utilisateurs : <http://forums.openqa.org/forum.jspa?forumID=3>

→ **Exemples complets dans selenium-core**

- `%SELENIUM_CORE_HOME%/core/TestRunner.html`

→ **Hortis**

- Communauté des Pratiques ALM