



Automatisation des applications web avec Selenium: Approfondissement

Objectifs

- Mettre en place des tests automatisés pour des applications Web





Agenda

- **Chapitre 1**
 - Principe de l'automatisation
 - Introduction à Selenium: IDE
 - Gestion des objets HTML
 - Fonctionnalités avancées
- **Chapitre 2**
 - Introduction à Selenium WebDriver
 - Selenium Grid
 - Exécution à partir de Jenkins
- **Chapitre 3**
 - Framework d'automatisation

Chap. 1: Introduction à Selenium

- Principe de l'automatisation des tests
- Premiers pas avec Selenium: Katalon Recorder
- Gestion des objets HTML
- Fonctionnalités avancées

1.1 Automatisation des tests

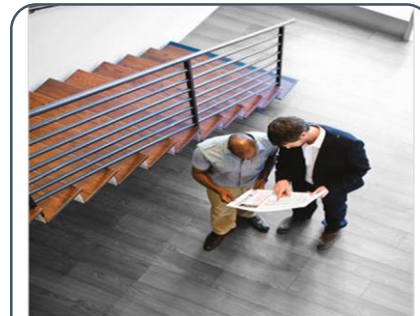
- Le Test fonctionnel permet de vérifier les caractéristiques qualité suivantes:



Précision



Interopérabilité

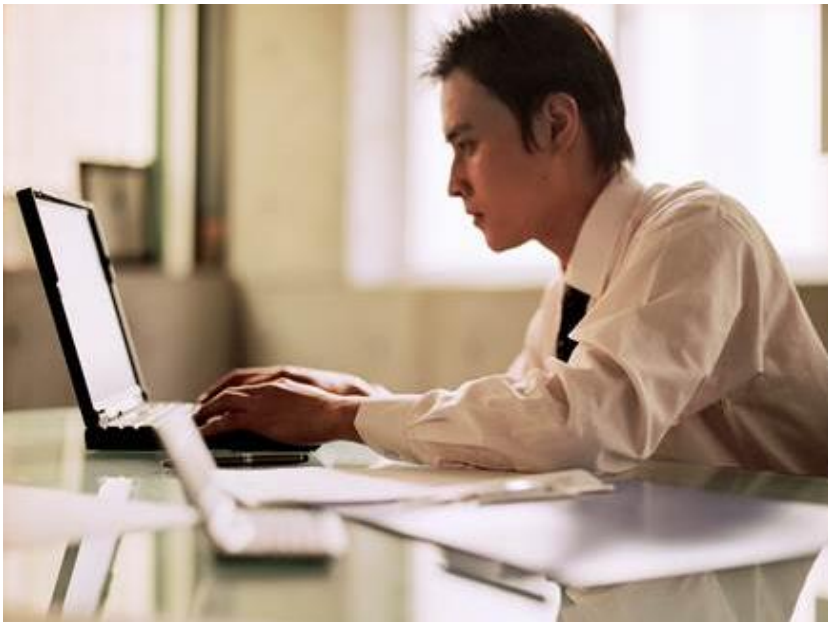


Prédictabilité



1.1 Automatisation des tests

Le Test fonctionnel peut être:



Manuel



Automatique

1.1 Automatisation des tests

Inconvénients du test manuel



Chronophage
Fastidieux/pénible
Contraignant
Coûteux
Incertain

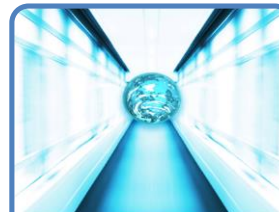
Manuel

1.1 Automatisation des tests

Avantage du test automatisé



Automatique



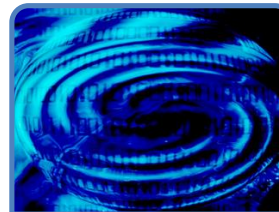
Rapide



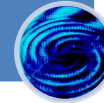
Fiable



Répétable



Compréhensible



Programmable



Réutilisable



1.1 Automatisation des tests

Les meilleurs candidats à l'automatisation

Tests de régression
des parcours-clés

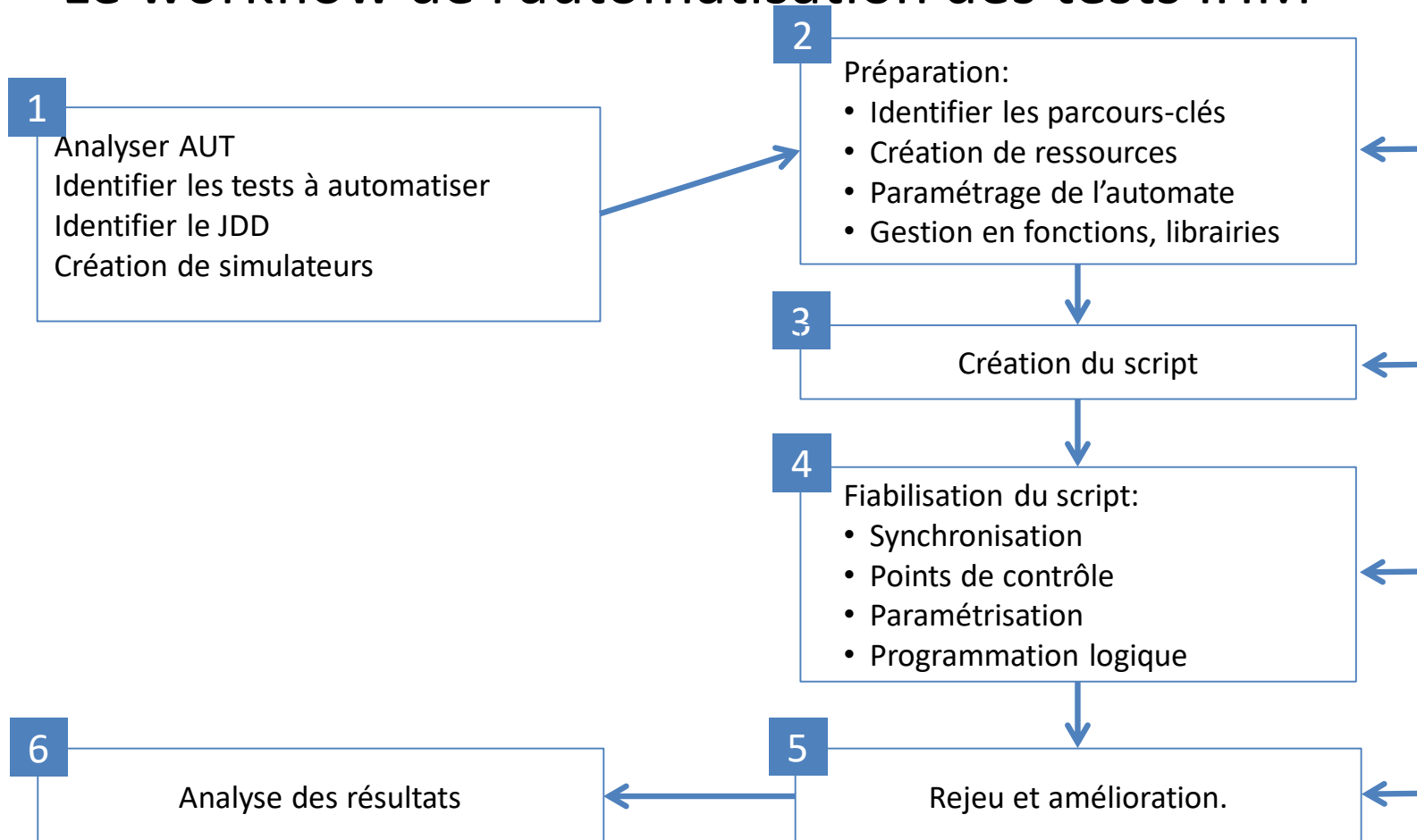
Smoke tests:
tests de confiance

Tests pilotés par les
données

Data Sanity Check:
initialisation des
données

1.1 Automatisation des tests

Le workflow de l'automatisation des tests IHM



1.1 Automatisation des tests

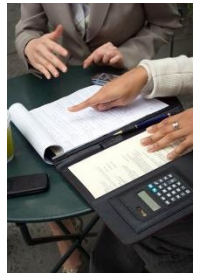
Analyse

Avant d'automatiser un test fonctionnel, il est opportun de:

- ➔ Revoir les étapes manuelles du test pour comprendre le processus métier
- ➔ Identifier par priorité les processus métiers les plus intéressants à automatiser
- ➔ Collecter les besoins en données de test
- ➔ Standardiser les termes avec des règles de nommage



1.1 Automatisation des tests



Documentation des processus métiers des parcours clés

Les processus métiers sont les blocs des scénarios de test.

Une bonne pratique d'automatisation consiste à créer des **petits blocs modulaires, réutilisables**, qu'on assemble pour concevoir le test.

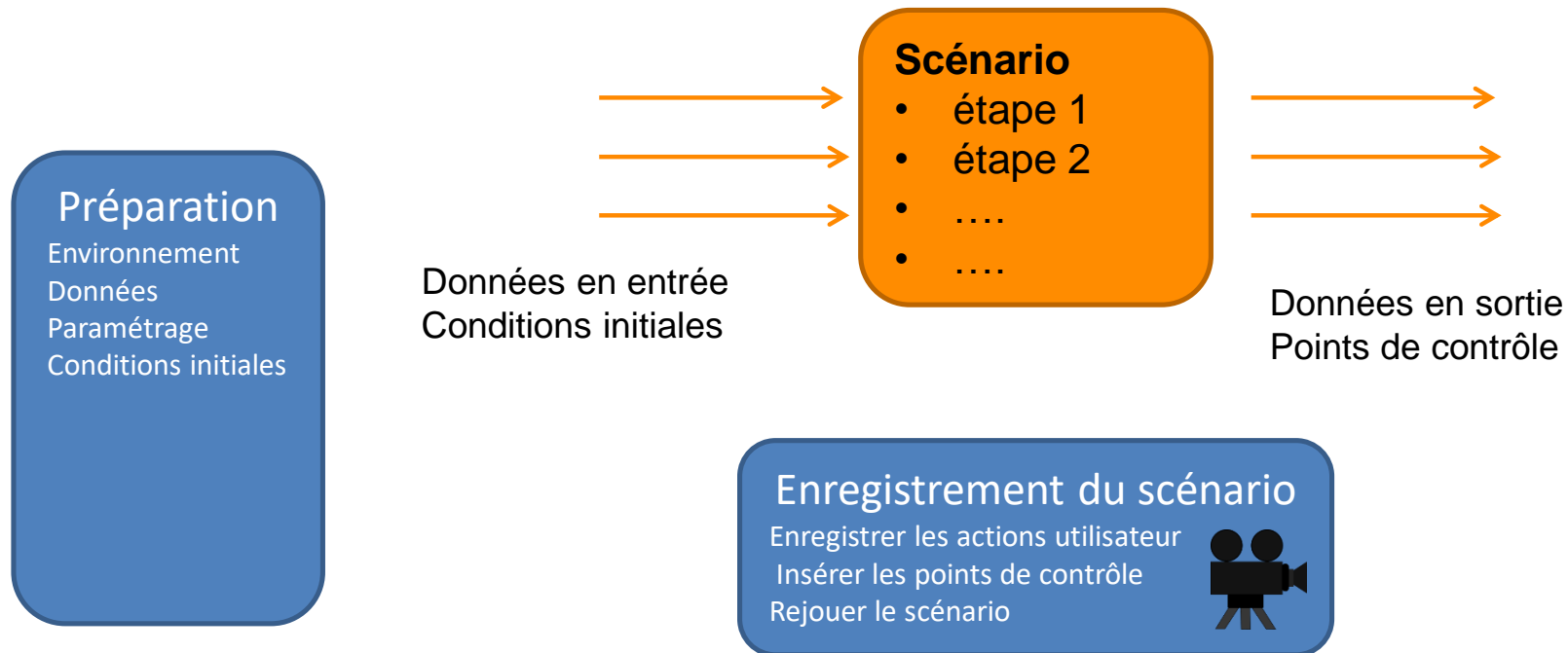
Une bonne connaissance et analyse des processus métiers permet d'optimiser l'effort de spécification des tests.

La spécification des tests se décompose comme suit:

- ➔ Détermination des conditions initiales et finales de chaque processus métiers
- ➔ Spécification des étapes de traitement du processus (saisie, validation,...)
- ➔ Vérification de la réponse de l'application
- ➔ Détermination des critères d'acceptation et d'échec

1.1 Automatisation des tests

Spécification d'un script



1.1 Automatisation des tests

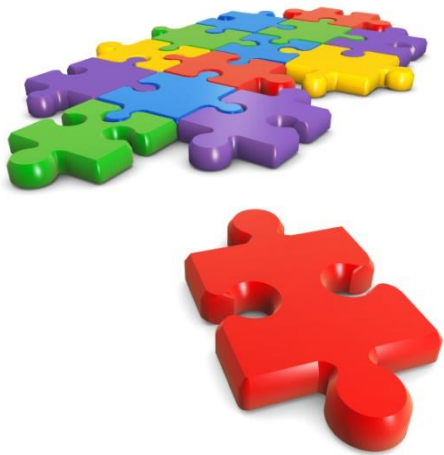
Améliorer le script



- Résoudre les problèmes synchronisation
- Paramétrer les données en entrée et sortie
- Insérer des points de contrôle complexes
- Utiliser si besoin des fonctions utilisateurs
- Insérer des scénarii de reprise

1.1 Automatisation des tests

Exploiter le script dans une usine



- Intégrer le script dans les scénarios existants
- Finaliser l'utilisation du framework si nécessaire
- Compléter la documentation du référentiel

1.1 Automatisation des tests

Outils d'automatisation des tests Web

Open source:



Commercial:



1.2 Premiers pas avec Selenium

Histoire

2004- Jason Huggins a développé une bibliothèque Javascript qui interagit avec des pages web → Selenium Core

2006- Simon Stewart (Ingénieur chez Google) a commencé à travailler sur un projet qu'il a appelé WebDriver. Simon voulait un outil de test qui communique directement avec le navigateur. → WebDriver

2008- Fusion de Selenium Core et WebDriver.

- Grande communauté
- support commercial
- Solution gratuite venue concurrencer les leaders commerciaux du marché

1.2 Framework Selenium

Plusieurs outils

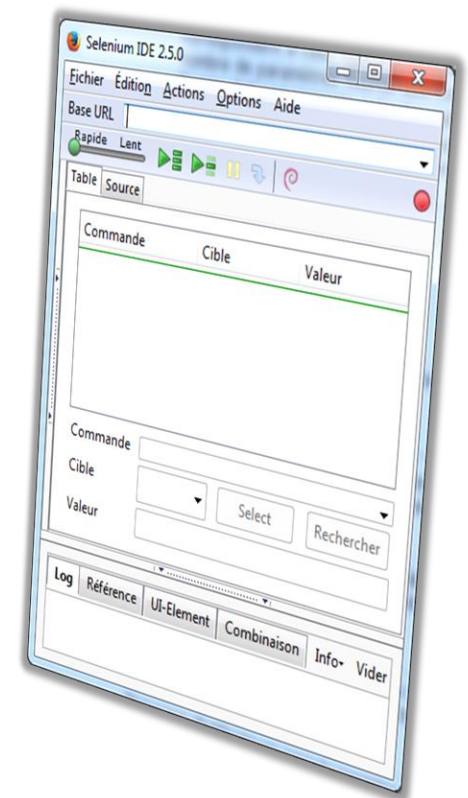
- **Selenium IDE**
- **Selenium WebDriver**
- **Selenium Grid**



IDE: Testeur non développeur

WebDriver: Testeur développeur

1.2 Selenium IDE



Katalon Recorder 3.0

the only extension that supports
all Selenese commands



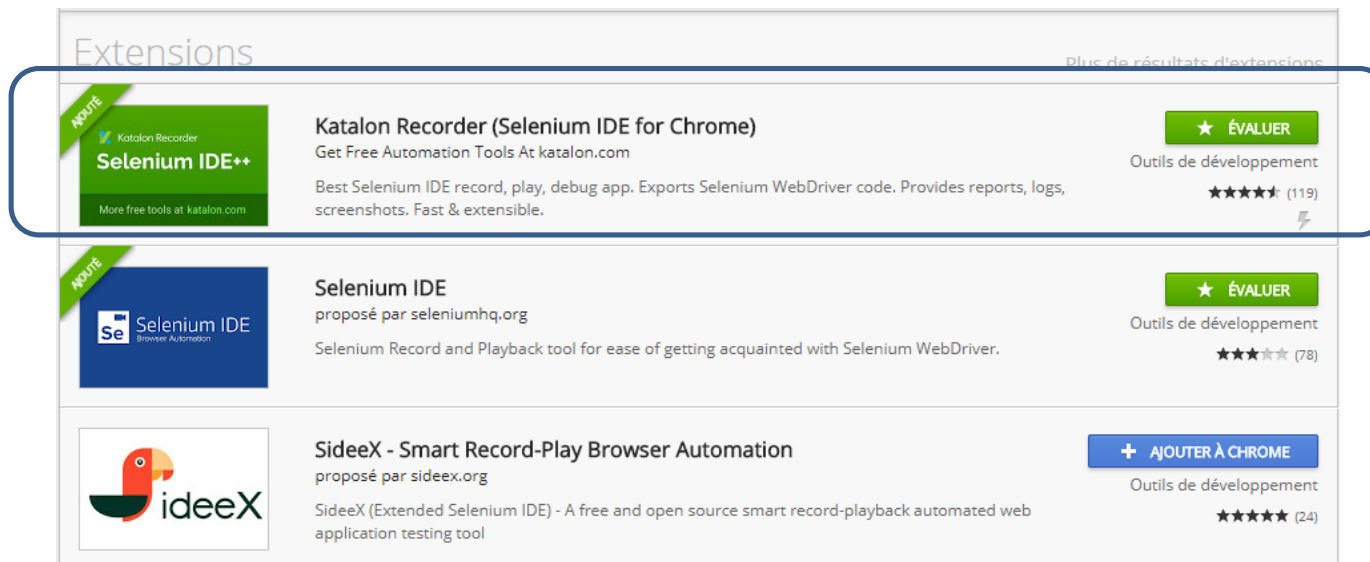
Chrome
web store



Firefox
Add-ons

1.2 Selenium IDE

- Selenium IDE (Integrated Development Environment): outil à utiliser pour développer des cas de test Selenium.
- Plugin ou extension de navigateur
- Permet de faire de l'automatisation sans scripting
- Plusieurs solutions Selenium IDE:

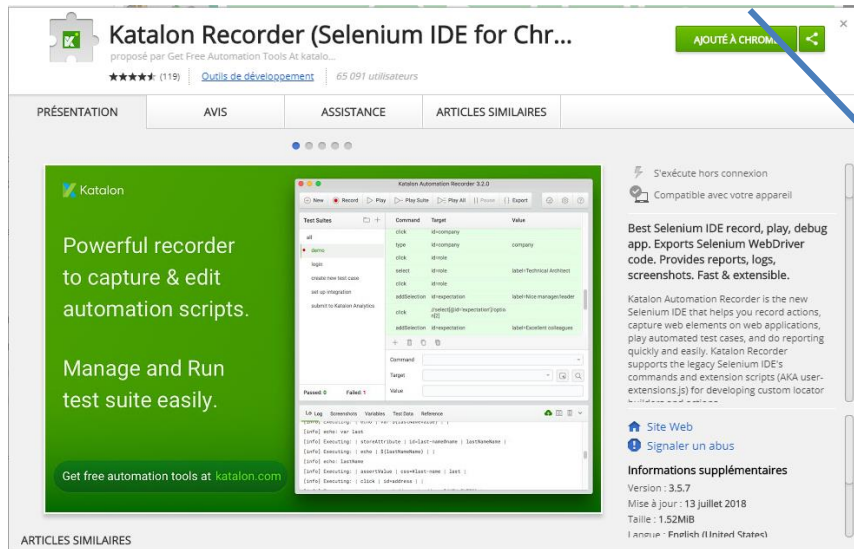


The screenshot displays the 'Extensions' page of the Chrome Web Store. A blue rounded rectangle highlights the first extension, 'Katalon Recorder (Selenium IDE for Chrome)'. Below it are two other extensions: 'Selenium IDE' and 'SideeX - Smart Record-Play Browser Automation'. Each extension card includes a logo, name, description, category, and user ratings.

Extension Name	Developer	Description	Category	Rating
Katalon Recorder (Selenium IDE for Chrome)	Katalon Recorder	Get Free Automation Tools At katalon.com. Best Selenium IDE record, play, debug app. Exports Selenium WebDriver code. Provides reports, logs, screenshots. Fast & extensible.	Outils de développement	★★★★★ (119)
Selenium IDE	seleniumhq.org	Selenium Record and Playback tool for ease of getting acquainted with Selenium WebDriver.	Outils de développement	★★★★★ (78)
SideeX - Smart Record-Play Browser Automation	sideex.org	SideeX (Extended Selenium IDE) - A free and open source smart record-playback automated web application testing tool	Outils de développement	★★★★★ (24)

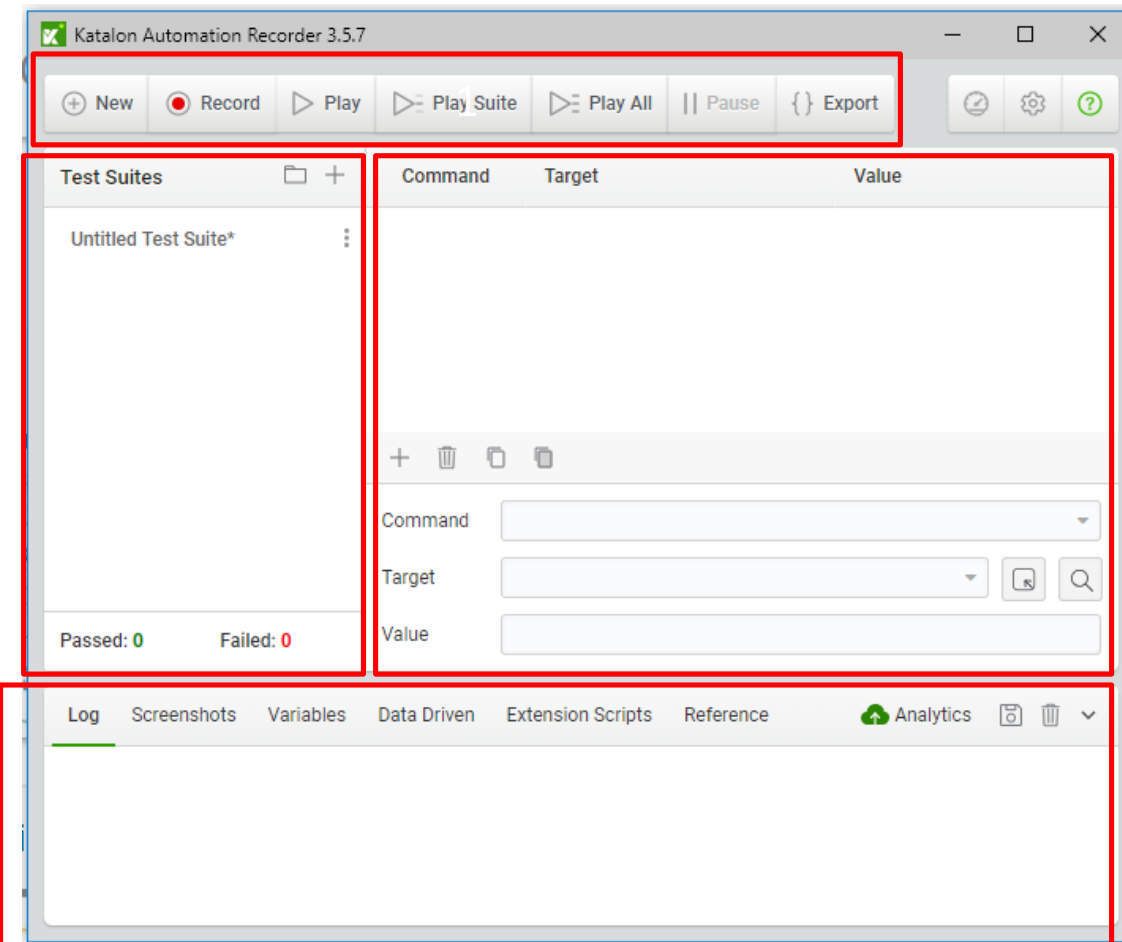
1.2 Katalon Recorder

Installation à partir de Chrome Extension



1.2 Katalon Recorder

Interface intuitif

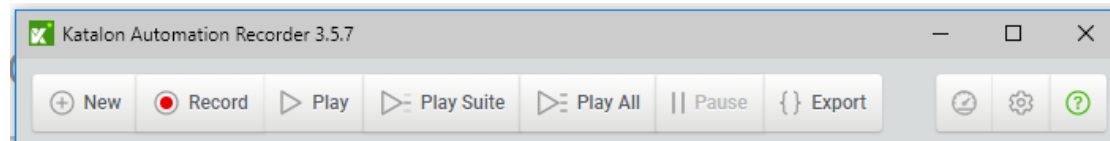


Main Toolbar
Test Case/ Suite Explorer
Test Case Details View
Log/Reference/Variable

1.2 Katalon Recorder

- **Approche par enregistrement**
 - Utilisent des scripts de test automatisés et nécessitent un effort important pour obtenir des bénéfices significatifs.
 - Enregistre les actions d'un testeur manuel.
- **LIMITATION**
 - L'approche ne peut pas être appliquée quand le nombre de tests automatisés est important,
 - Le script capturé peut être instable lors de l'occurrence d'événements inattendus car c'est une représentation linéaire avec des données et actions spécifiques appartenant à chaque script.

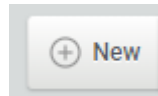
1.2 Katalon Recorder: Toolbar



Button	Description
New	Créer un test ou une suite de test
Record	Enregistrer un script
Play	Exécuter le script sélectionné
Play Suite	Exécuter la suite sélectionnée
Play All	Tout exécuter
Pause/Resume	Faire pause/reprise d'une exécution
Stop	Stopper une exécution
Export	Exporter le script dans un langage de développement
Speed	Ajuster la vitesse d'exécution
Setting	Paramètres
Help icon	Guide

1.2 Katalon Recorder: Test

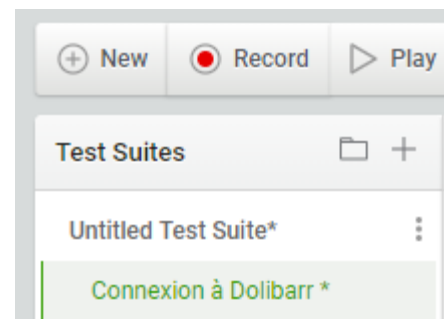
- Créer un test



Katalon Recorder (Selenium IDE for Chrome)

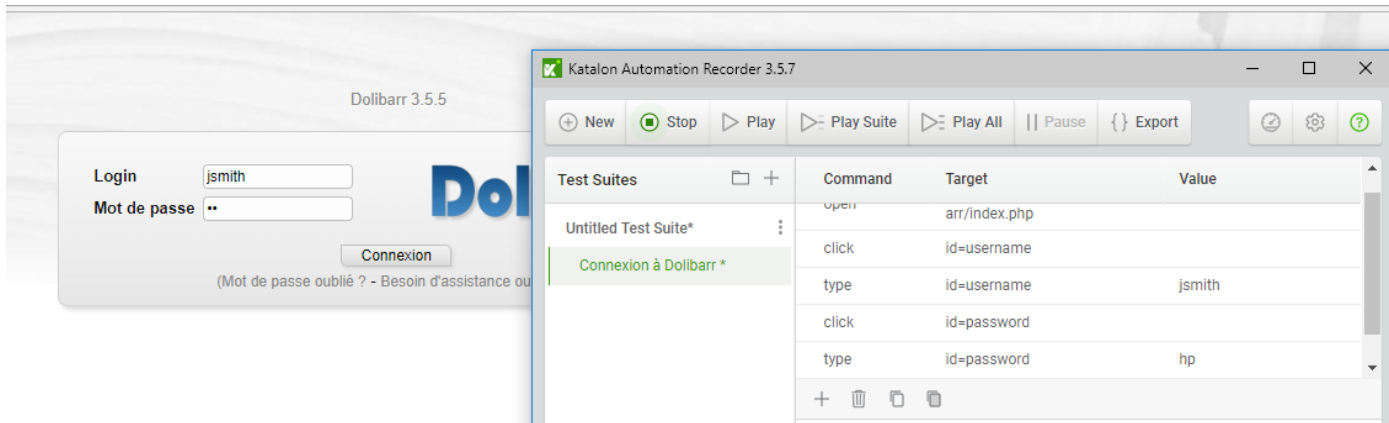
Please enter the Test Case's name

OK Annuler



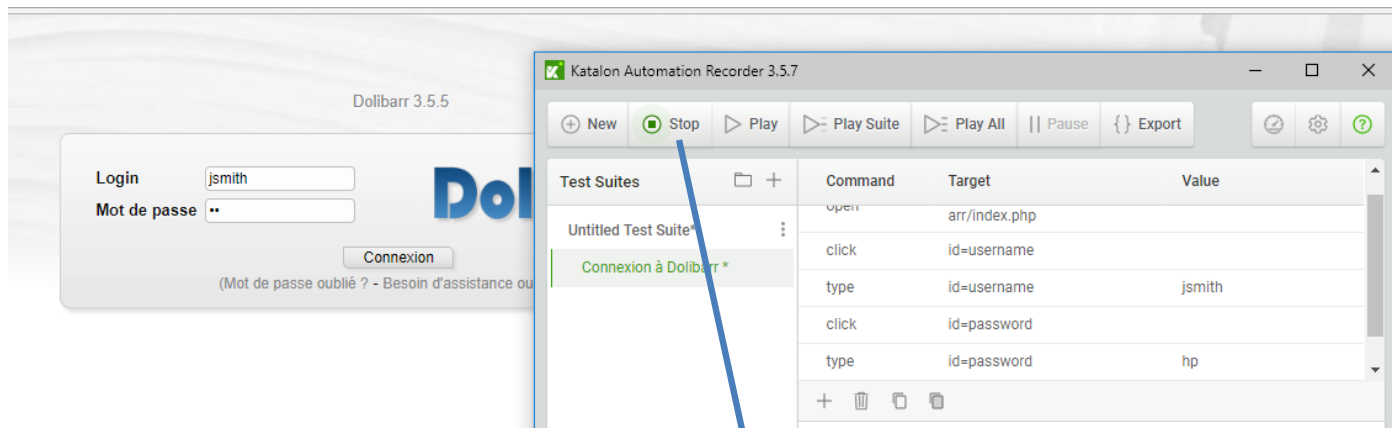
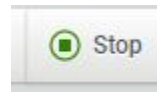
1.2 Katalon Recorder: Test

- Enregistrer un scénario



1.2 Katalon Recorder: Test

- Stopper l'enregistrement



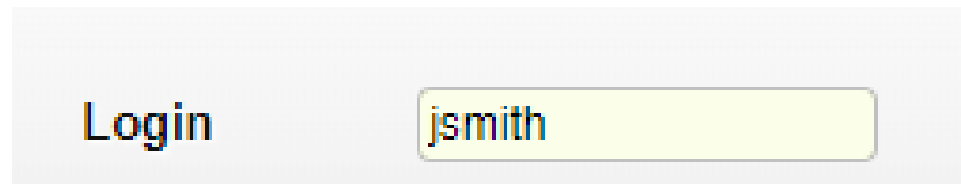
Command	Target	Value
open	http://demo.testlogiciel.pro/dolibarr/index.php	
click	id=username	
type	id=username	jsmith
click	id=password	
type	id=password	hp
click	//input[@value=' Connexion ']	
click	//img[@alt='Déconnexion']	

1.2 Katalon Recorder: Test

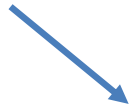
- Script Selenium

- Command: action sur le navigateur
- Target: objet ciblé par l'action
- Value: valeur d'entrée de l'action

click	id=username	
type	id=username	jsmith



1.2 Katalon Recorder: Exécution



Login

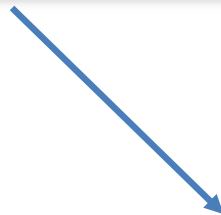
Mot de passe

Connexion

(Mot de passe oublié ? - Besoin d'assistance ou aide ?)

Dolibarr ERP/CRM

Command	Target	Value
open	http://demo.testlogiciel.pro/dolibarr/index.php	
click	id=username	
type	id=username	jsmith
click	id=password	
type	id=password	hp
click	css=input.button	
click	//img[@alt='Déconnexion']	







Passed: 1 Failed: 0

Target

Value

Log Screenshots Variables Data Driven Extension Scripts Reference

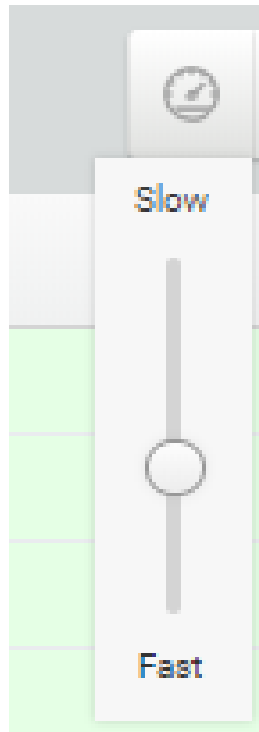
Analytics    

```
[info] Executing: | click | id=username | |
[info] Executing: | type | id=username | jsmith |
[info] Executing: | click | id=password | |
[info] Executing: | type | id=password | hp |
[info] Executing: | click | css=input.button | |
[info] Executing: | click | //img[@alt='Déconnexion'] | |
[info] Time: Sat Jul 14 2018 13:35:13 GMT+0200 (heure d'été d'Europe centrale) Timestamp: 1531568113543
[info] Test case passed
```

Selenium: Automatisation des applications

1.2 Katalon Recorder: Exécution

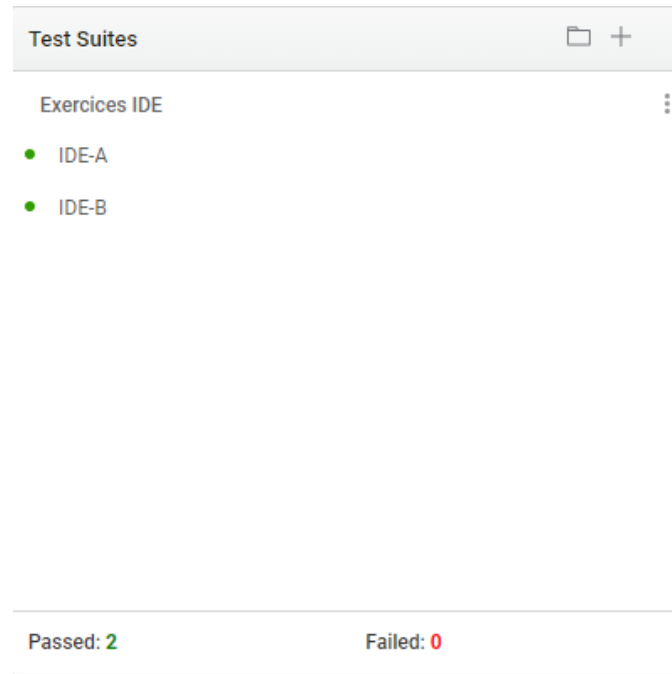
- Vitesse



1.2 Katalon Recorder: Exécution

- Résultats

Statut
d'exécution

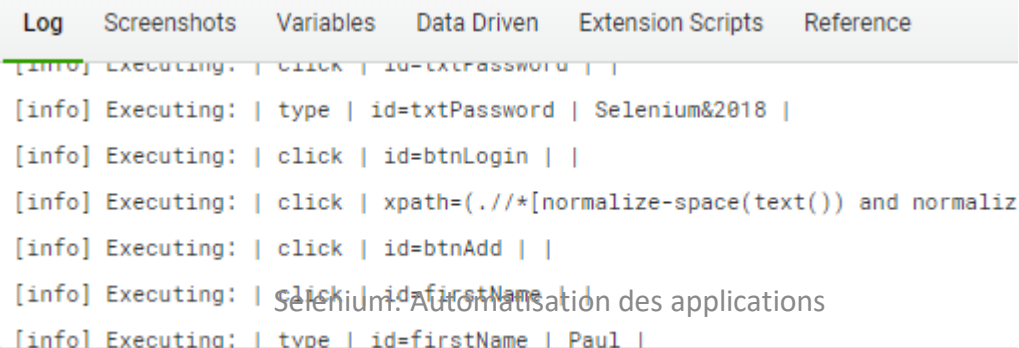


The screenshot shows the 'Test Suites' panel in Katalon Recorder. It lists 'Exercices IDE' with two sub-items, 'IDE-A' and 'IDE-B', both marked with green dots indicating successful execution. At the bottom, a summary bar shows 'Passed: 2' in green and 'Failed: 0' in red.

Test Suites
Exercices IDE
• IDE-A
• IDE-B

Passed: 2 Failed: 0

Log du
rapport
d'exécution

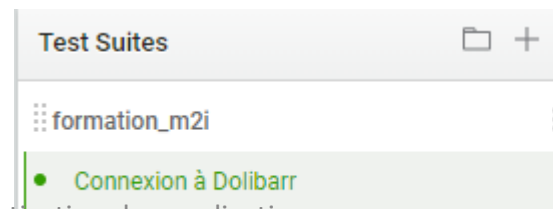
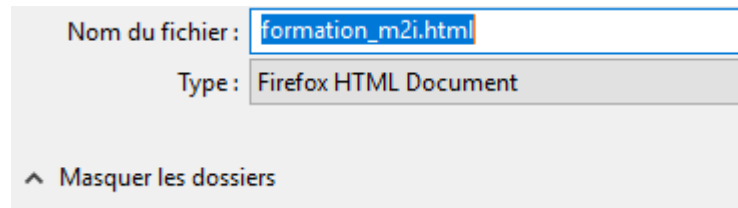
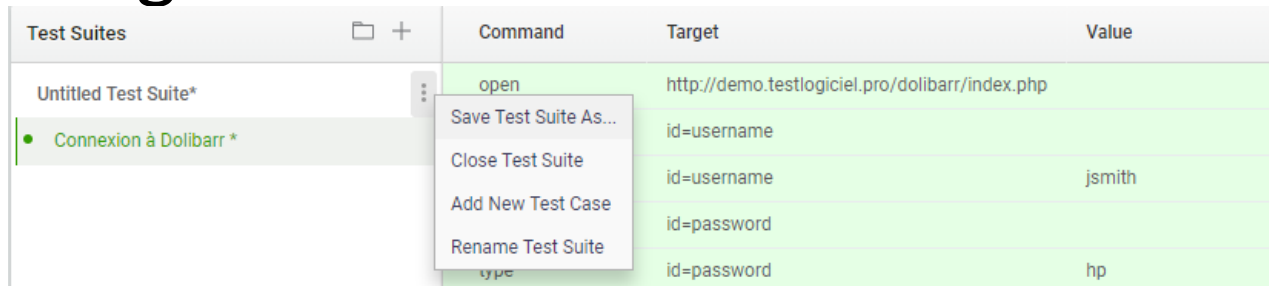


The screenshot shows the 'Log' tab in Katalon Recorder. It displays a series of log entries for an execution, including commands like 'click', 'type', and 'click' with their respective parameters. The text 'Selenium: Automatisation des applications' is visible in the background.

Log	Screenshots	Variables	Data Driven	Extension Scripts	Reference
[info] Executing: click id=txtPassword					
[info] Executing: type id=txtPassword Selenium&2018					
[info] Executing: click id=btnLogin					
[info] Executing: click xpath=(.//*[normalize-space(text()) and normalize-space(.)='Selenium: Automatisation des applications'])					
[info] Executing: click id=btnAdd					
[info] Executing: click id=firstName					
[info] Executing: type id=firstName Paul					

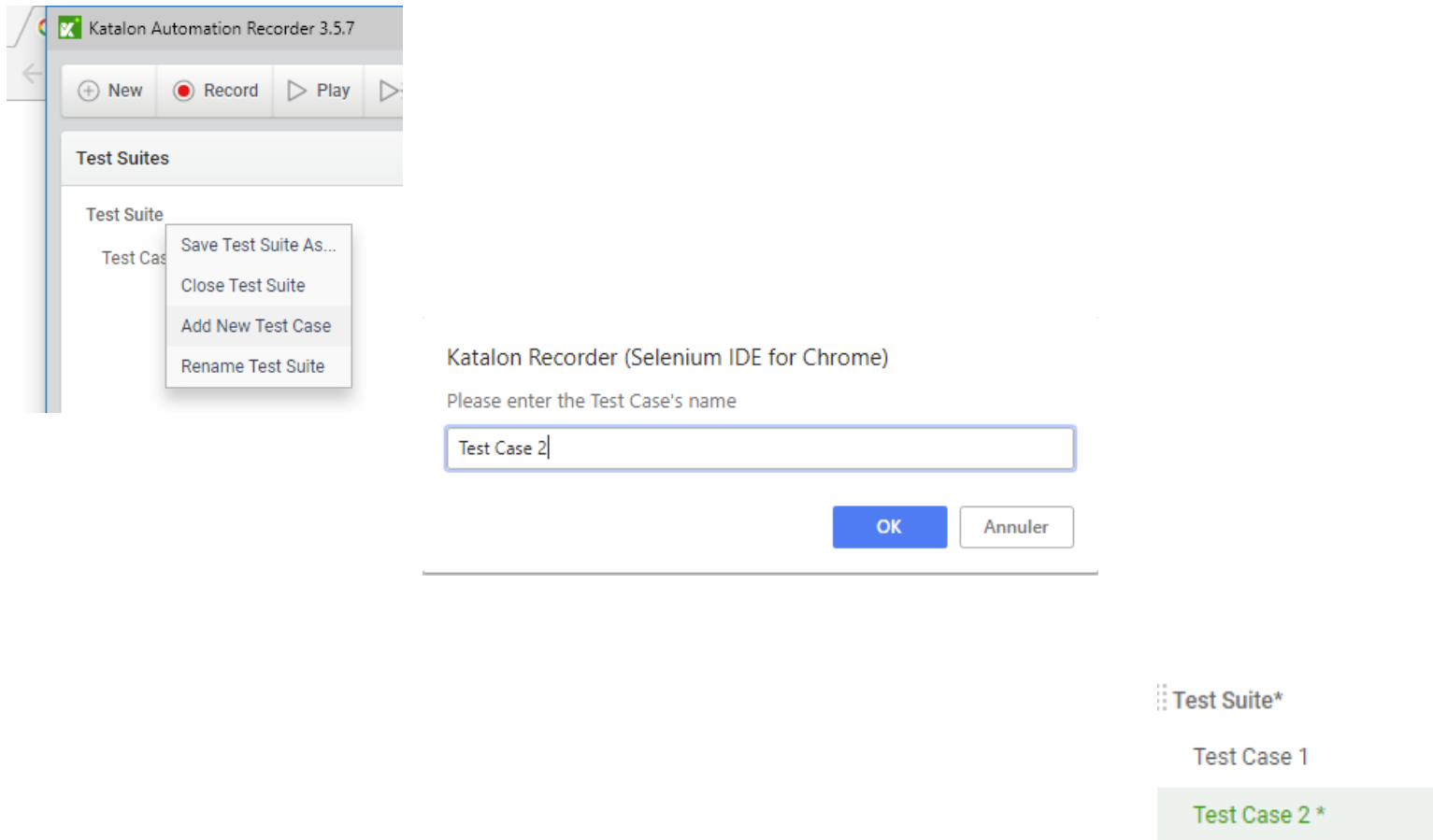
1.2 Katalon Recorder: Suite

- Sauvegarder



1.2 Katalon Recorder: Suite

Ajouter un cas de test dans la suite de test



1.2 Katalon Recorder: TP

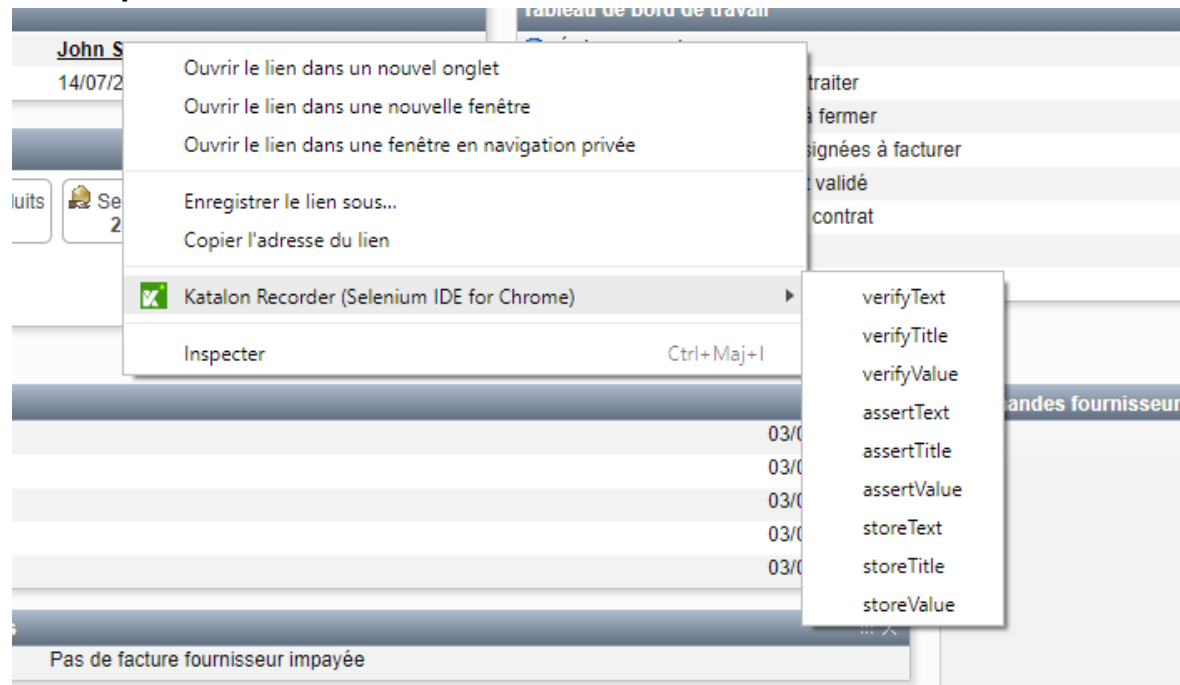
Exercice IDE-A: Enregistrer un Test case

- Scénario: Ajouter un nouveau salarié
 - Lancer Chrome: <http://orangehrm.selenium-formation.org/symfony/web/index.php/auth/login>
 - Saisir **Username**=admin
 - Saisir **Password**=Selenium&2018
 - Cliquer sur le bouton **Login**
 - Cliquer sur **PIM**
 - Cliquer sur le bouton **Ajouter**
 - Saisir les informations suivantes:
 - **Prénom**=....
 - **Nom du milieu**=....
 - **Nom de famille**=....
 - Cliquer sur **Sauver**
 - Cliquer sur **Editer**
 - Saisir les informations suivantes:
 - **Numéro du permis**=XXXXXX
 - **Sexe**=Masculin
 - **Etat marital**=Marié
 - **Nationalité**=Français
 - **Date de naissance**=2000-01-01
 - Cliquer sur **Sauver**
 - Cliquer **Welcome Admin**
 - Cliquer sur **Déconnexion**

Exécuter le test case en vitesse medium

1.2 Katalon Recorder: Checkpoint

Ajouter un point de vérification avec le menu contextuel

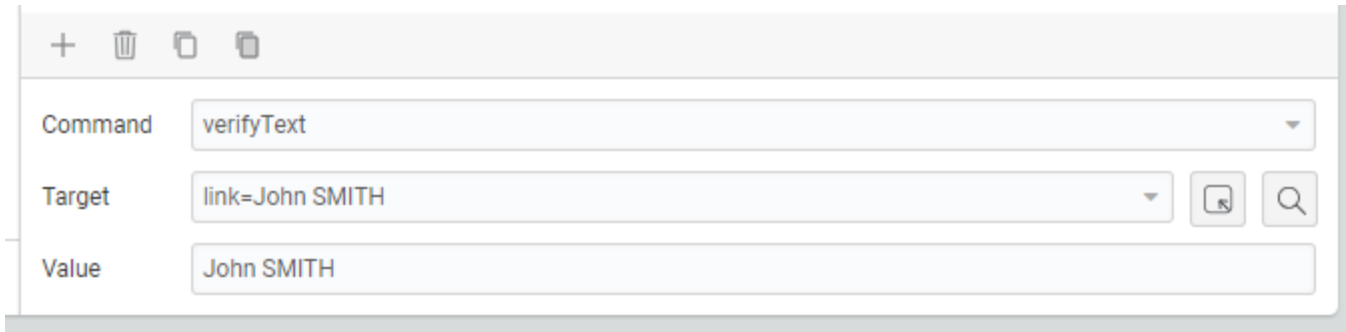


Quelles sont les vérifications possibles pour une application Web?

1.2 Katalon Recorder: Checkpoint

Ajouter un point de vérification avec le menu contextuel

- Verify: permet de vérifier un élément sans stopper le test en cas d'échec
- Assert: permet de vérifier un élément et stopper le test en cas d'échec
- Store: permet de stocker le résultat dans



The screenshot shows the Katalon Recorder interface for adding a checkpoint. At the top, there is a toolbar with icons for adding (+), deleting (trash), copying, and pasting. Below this, there are three input fields: 'Command' with a dropdown menu set to 'verifyText', 'Target' with a dropdown menu set to 'link=John SMITH' and two icons (a square with a cursor and a magnifying glass), and 'Value' with a text box containing 'John SMITH'.

1.2 Katalon Recorder: Variable

Stocker une valeur dans une variable

Pour gérer les données dynamiques créées par le système et les utiliser plus tard dans le script

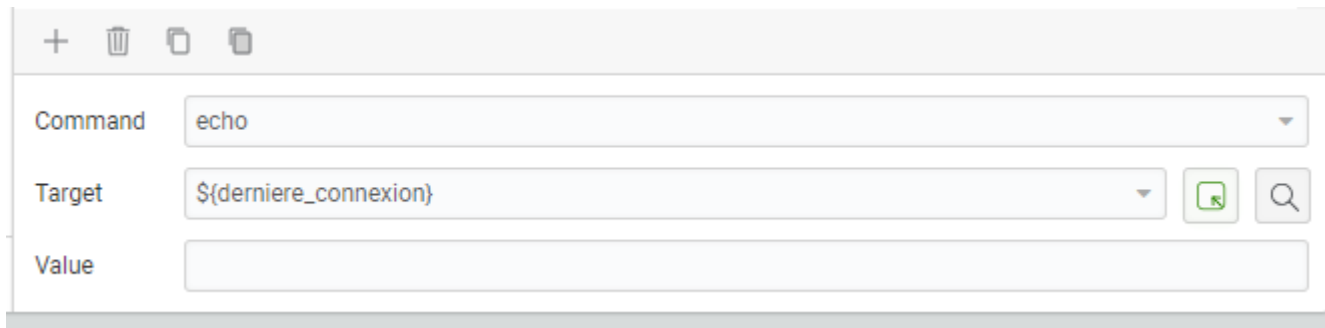
Informations	
Utilisateur	John SMITH
Connexion précédente	15/07/2018 06:02

Command	storeText
Target	//tr[3]/td[2]
Value	derniere_connexion

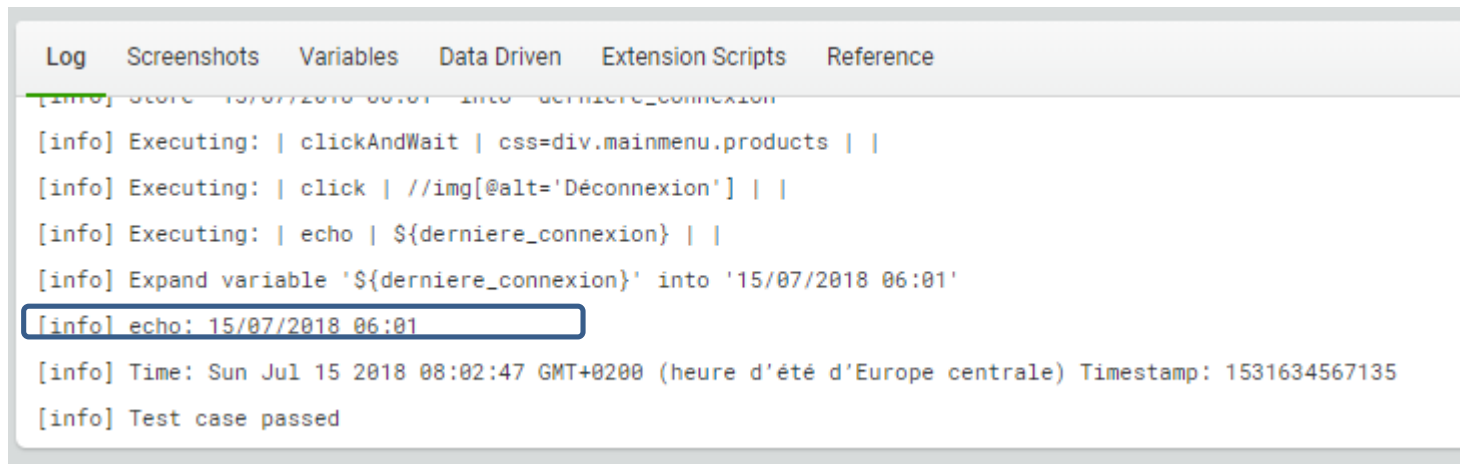
Passed: 1	Failed: 0	Value	
Log	Screenshots	Variables	Data Driven
			Extension Scripts
			Reference
derniere_connexion		string	15/07/2018 06:01

1.2 Katalon Recorder: Variable

Utiliser la variable



The screenshot shows the Katalon Recorder's command configuration window. It has a toolbar at the top with icons for adding, deleting, and copying commands. Below the toolbar, there are three fields: 'Command' with a dropdown menu showing 'echo', 'Target' with a text input containing '\${derniere_connexion}' and a search icon, and 'Value' with an empty text input.



The screenshot shows the Katalon Recorder's log window. It has a tabbed interface with 'Log' selected. The log entries show the execution of a test case. The entry '[info] echo: 15/07/2018 06:01' is highlighted with a blue box. The log also shows the expansion of the variable '\${derniere_connexion}' into '15/07/2018 06:01' and the final test case passed message.

```
Log Screenshots Variables Data Driven Extension Scripts Reference
[info] Store 15/07/2018 06:01 into derniere_connexion
[info] Executing: | clickAndWait | css=div.mainmenu.products | |
[info] Executing: | click | //img[@alt='Déconnexion'] | |
[info] Executing: | echo | ${derniere_connexion} | |
[info] Expand variable '${derniere_connexion}' into '15/07/2018 06:01'
[info] echo: 15/07/2018 06:01
[info] Time: Sun Jul 15 2018 08:02:47 GMT+0200 (heure d'été d'Europe centrale) Timestamp: 1531634567135
[info] Test case passed
```

1.2 Katalon Recorder: TP

Exercice IDE-B: Vérifier la connexion

- Scénario: Test de la connexion
 - Lancer Chrome: <http://orangehrm.selenium-formation.org/>
 - Cliquer sur **LOGIN**
 - Vérifier l'apparition du message: **Informations d'identification valides**
 - Saisir **Username**=admin
 - Cliquer sur **LOGIN**
 - Vérifier que **Username** est vide
 - Vérifier l'apparition du message: **Informations d'identification valides**
 - Saisir **Username**=admin
 - Saisir **Password**=fauxmotdepasse
 - Cliquer sur **LOGIN**
 - Vérifier que **Username** est vide
 - Vérifier que **Password** est vide
 - Vérifier l'apparition du message: **Informations d'identification valides**
 - Saisir Username=admin
 - Saisir Password=Selenium&2018
 - Cliquer sur **LOGIN**
 - Vérifier que **Dashboard** est affiché
 - Cliquer sur Welcome Admin
 - Cliquer sur Déconnexion

Exécuter le test case

1.2 Katalon Recorder: TP

Exercice IDE-A: Vérifier la création du salarié

- Reprendre le scénario IDE-A
 - Stocker l'id du salarié dans une variable avant de cliquer sur Sauver
 - Avant la déconnexion, rajouter les étapes suivantes
 - Cliquer sur **Liste des salariés**
 - Saisir **Id**=variable stockée
 - Cliquer sur **Rechercher**
 - Cliquer sur le lien du **numéro du salarié**
 - **Vérifier le nom, et prénom**

Exécuter le test case

1.3 Gestion des objets



Manipulation des objets avec Selenium IDE



1.3 Gestion des objets

Plusieurs techniques d'enregistrement de script de test

- Par la reconnaissance des objets graphiques
- Par la reconnaissance d'images
- Par la reconnaissance analogique (Clavier, Souris, reconnaissance de texte à l'écran)

Selenium utilise principalement la reconnaissance des objets

- Identification des objets HTML avec les propriétés DOM:
 - Document Object Model

The screenshot shows a web form titled "Nouveau tiers (prospect, client, fournisseur)". The form includes several input fields and dropdown menus. A blue box highlights the "Type du société:" section, which contains two radio buttons: "Créer tiers" (selected) and "Créer un tiers + un contact/adresse fils". Another blue box highlights the "Nom du tiers" text input field. A third blue box highlights the "Code client" text input field, which contains the value "CU1904-1842". A blue arrow points from the "Nom du tiers" input field to a callout box containing the DOM selector: `<input type="text" class="minwidth300" maxlength="128" name="firstname" id="firstname" value> == $0`. The form also includes sections for "Prospect / Client", "Fournisseur", and "État".

1.3 Gestion des objets

Le Target: identification des éléments

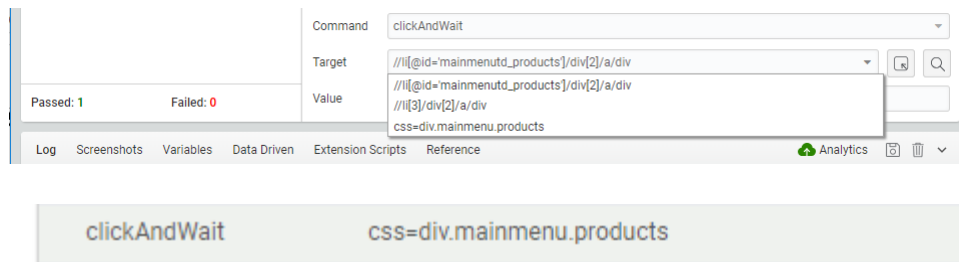
Un target est un élément graphique de l'IHM de l'application à tester.



Target

//li[@id='mainmenutd_products']/div[2]/a/div

Identification doit être fiable



1.3 Gestion des objets

Localisation par identifiant

id=username

```
<input type="text" id="username" name="username" class="flat" size="15" maxlength="40" value tabindex="1">
```

Localisation par name

name=username

```
<input type="text" id="username" name="username" class="flat" size="15" maxlength="40" value tabindex="1">
```

Localisation par nom de lien

link=John SMITH

Informations	
Utilisateur	John SMITH
Connexion précédente	16/07/2018 09:01

1.3 Gestion des objets

Localisation par CSS Selector

Plusieurs formats de sélecteurs CSS pour identifier à partir du tag en se basant sur les propriétés:

- id: #
- class : .
- attribute : [attribute='value']
- Innertext: :contains('texte')

```
<input type="text" class="minwidth300" maxlength="128"
name="name" id="name" value autofocus="autofocus"> == $0
```

- css=input#username
- css=input#name.minwidth300
- css=input[autofocus='autofocus']

Nom du tiers

test

- css=label:contains('Nom du tiers')

```
▼<span id="TypeName" class="fieldrequired">
  <label for="name">Nom du tiers</label> == $0
</span>
```

1.3 Gestion des objets

Localisation par xpath

XPath est le langage utilisé pour localiser des nœuds dans un document XML, Comme HTML peut être une implémentation de XML (XHTML), les utilisateurs de Selenium peuvent utiliser ce langage puissant pour cibler des éléments dans leurs applications Web.

```
<html>
<body>
  <form id="loginForm">
    <input name="username" type="text" />
    <input name="password" type="password" />
    <input name="continue" type="submit" value="Login" />
    <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

- `/html/body/form[1]`

Chemin absolu, premier form sous body sous html

- `//form[1]`

Premier formulaire dans le code HTML

- `//form[@id='loginForm']`

form dont id= 'loginForm'

- `//input[@name='username']`

input dont name='username'

- `//form[@id='loginForm']/input[1]`

premier input situé sous le form dont id 'LoginForm'

- `//input[@name='continue'][@type='button']`

input dont name='continue et type ='button'

1.3 Gestion des objets

Localisation par xpath

```
<input type="submit" class="button" value="&nbsp; Identifiant &nbsp;" tabindex="5"
xpath="1"> == $0
```

- `//input[contains(@value,'Identifiant')]`

input pour lequel l'attribut value contient 'Identifiant'



- `//input[@id=(//label[contains(., "Pour:")]/@for)]`

Sélection d'un champ texte dont l'id est la valeur de l'attribut for du label contenant tableau ayant un titre

`//label[.='Pour']`

`//label[text()='Pour']`

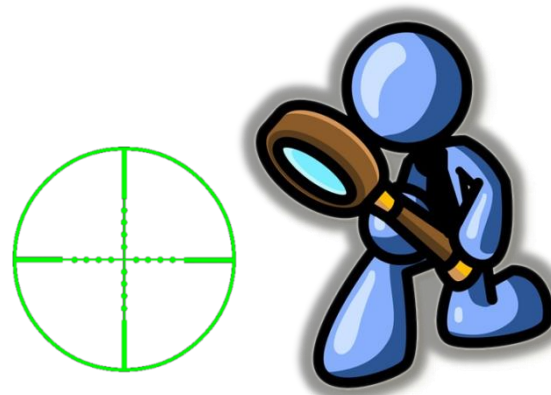
- `//table[.//th='Réf.']/tr[4]/td[2]`

Sélection de la 4è ligne et 2è colonne du tableau dans lequel on trouve un th='Réf'

1.3 Gestion des objets

La meilleure stratégie pour identifier

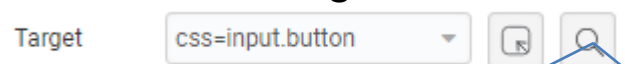
- 1) id
- 2) name
- 3) xpath
- 4) css



1.3 Gestion des objets

Tester le target d'un élément

- Mettre un point d'arrêt: « Toggle Breakpoint »
- Exécuter à partir de la ligne sélectionnée: « Play from here »
- Exécuter seulement la ligne sélectionnée: « Play this command »



Log Screenshots Variables Data Driven Extension Scripts

```
[info] Executing: | open | http://demo.testlogiciel.pro/
[info] Executing: | type | id=username | ${username} |
[info] Expand variable '${username}' into 'jsmith'
[info] Executing: | type | id=password | ${password} |
[info] Expand variable '${password}' into 'hp'
[info] Breakpoint: Stop.
[info] Pausing
[info] Stop executing
[info] Element is found in top frame.
```

Log Screenshots Variables Data Driven

```
[info] Pausing
[info] Stop executing
[info] Element is found in top frame.
[info] Element is found in top frame.
[info] Element is found in top frame.
[info] Element is found in top frame.
[info] Element is found in top frame.
[error] Element is not found.
```

1.3 Gestion des objets

Exercice IDE-C: Utiliser les targets

Créer un nouveau test Exercice IDE-C

- Copier les steps à partir de l'exercice A
- Nettoyer le script: Enlever les steps de clic inutile
- Remplacer tous les targets par une identification par xpath
- Exécuter le test

1.3 Gestion des objets

Quelques commandes typiques de Selenium

open: ouvre une page à l'aide d'une URL.

click/clickAndWait: effectue une opération de clic, et éventuellement attend une nouvelle page à charger

verifyTitle/assertTitle: vérifie un titre prévu de la page.

verifyTextPresent: vérifie un texte quelque part sur la page.

1.3 Gestion des objets

Quelques commandes typiques de Selenium

verifyElementPresent: vérifie qu'un élément de la page attendu, tel qu'il est défini par la balise HTML, est présent sur la page.

verifyText: vérifie que texte prévu et sa balise HTML correspondante sont présents sur la page.

verifyTable: vérifie les contenus attendus d'une table.



waitForPageToLoad: interrompt l'exécution jusqu'à ce qu'une nouvelle page se charge. Appelé automatiquement lorsque la commande clickAndWait est utilisé.

waitForElementPresent: interrompt l'exécution jusqu'à ce qu'un élément de la page attendu, tel qu'il est défini par la balise HTML, est présent sur la page.

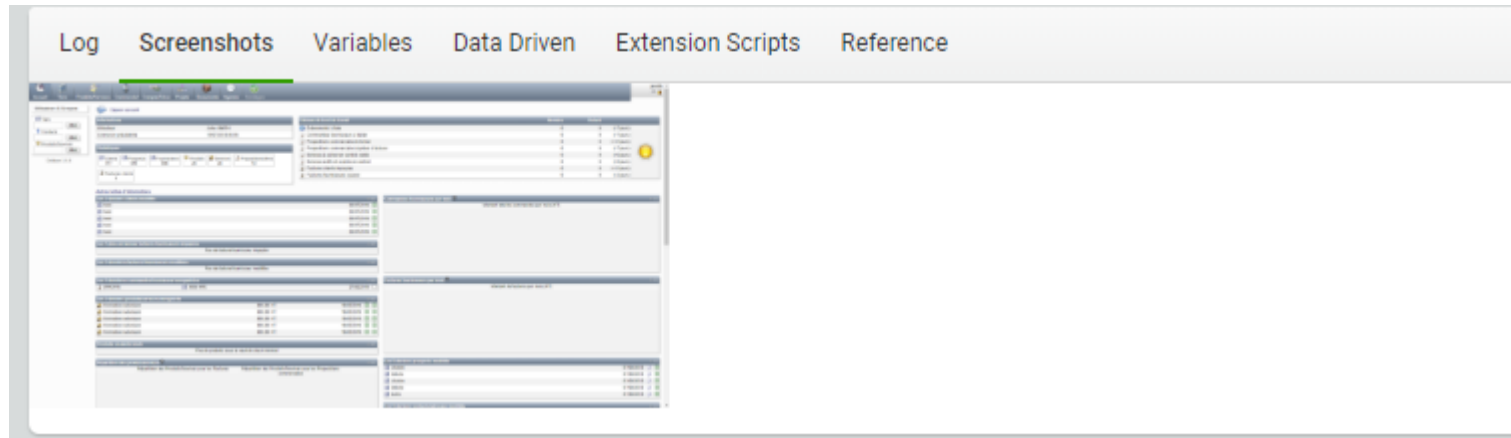
1.3 Gestion des objets

Prendre une capture d'écran

Command

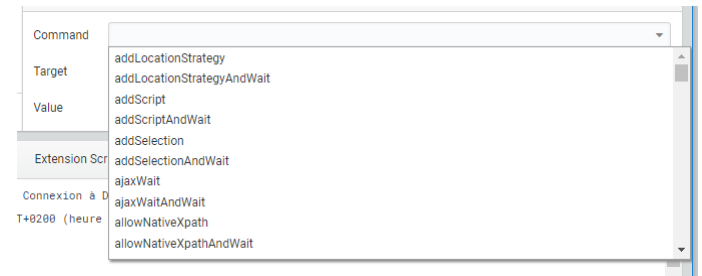
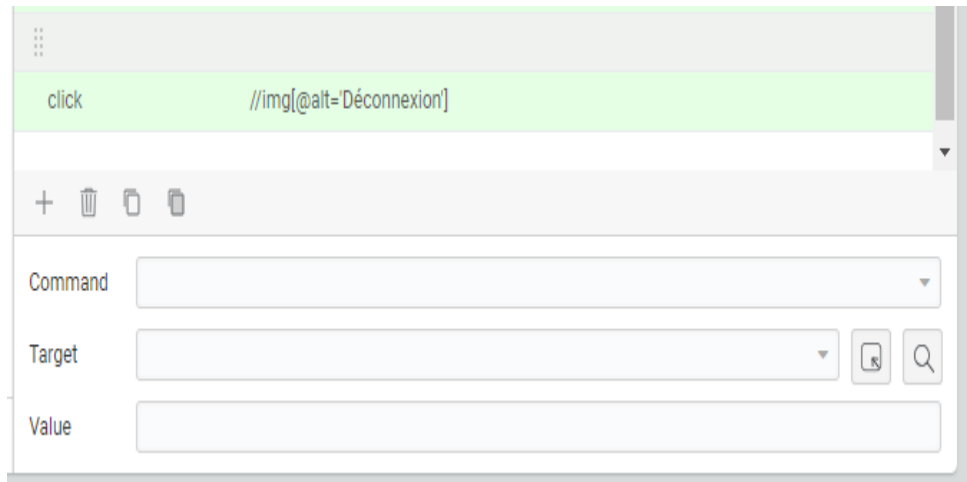
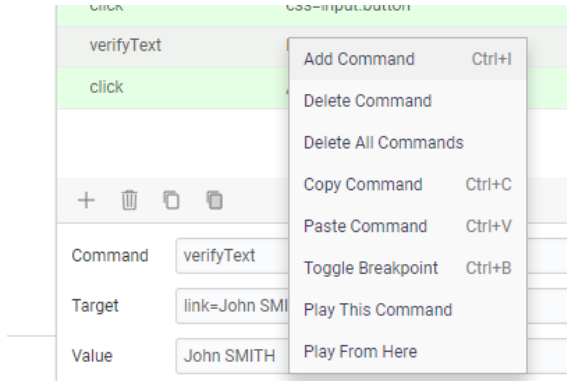
Target  

Value



1.3 Gestion des objets

Insérer une commande sans enregistrement



1.3 Gestion des objets

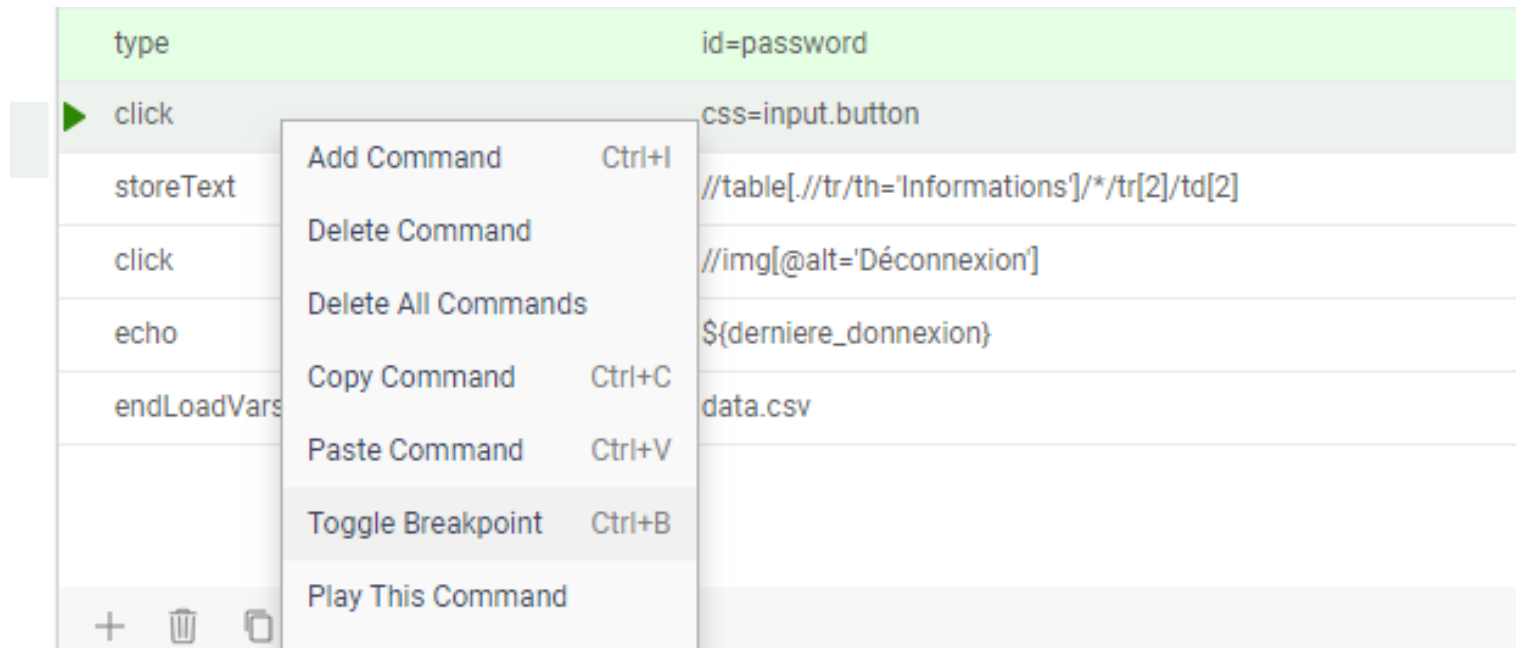
Les commandes selenese



1.3 Gestion des objets

Fonctionnalité de débogage

- Mettre un point d'arrêt: « Toggle Breakpoint »
- Exécuter à partir de la ligne sélectionnée: « Play from here »
- Exécuter seulement la ligne sélectionnée: « Play this command »



1.3 Gestion des objets

Exercice IDE-D1: Créer un test en utilisant le recorder

- Scénario: Créer un poste
- Lancer l'application à l'adresse : <http://orangehrm.selenium-formation.org/>
- Vérifier la présence du champ « Nom d'utilisateur »
- Saisir **Nom d'utilisateur**=admin
- Saisir **Mot de passe**=Selenium&2018
- Cliquer sur **Connexion**
 - Vérifier que vous êtes sur le Tableau de bord
- Cliquer sur **Recrutement**
- Cliquer sur **Offres d'emploi**
- Cliquer sur le bouton **Ajouter**
 - Vérifier que la valeur par défaut de Titre du poste est= « --Sélectionner --»
 - Vérifier sur Actif est coché
- Sélectionner un **Titre de poste**
- Saisir **Nom du poste vacant** : ...
- Saisir **Manager qui recrute** : John SMITH
- Nombre de postes : 3
- Décoché Publier dans Flux RSS
- Cliquer sur **Sauver**
 - Vérifier que vous êtes sur la page : Modifier les postes vacants
- Cliquer sur **Retour**
 - Vérifier que le bouton rechercher est présent
- Sélectionner Vacant=nom du poste
- Cliquer sur Rechercher
 - Vérifier que dans le tableau, la valeur de la 2è ligne 2è colonne est le nom du poste
- Prendre une capture d'écran
- Cliquer sur Déconnexion

1.3 Gestion des objets

Exercice IDE-D2: Créer un Test case sans recorder

- Scénario: Créer une note de frais
 - Lancer Chrome: <http://dolibarr.selenium-formation.org/>
 - Saisir **Login**=jsmith
 - Saisir **Mot de passe**=dolibarrhp
 - Cliquer sur le bouton **Identifiant**
 - Cliquer sur **GRH**
 - Cliquer sur **Nouveau** pour ajouter une note de frais
 - Saisir **Date début** =22/10/2018
 - Saisir **Date de fin** =22/10/2018
 - Sélectionner **Utilisateur**=John SMITH
 - Sélectionner **Utilisateur Approbateur**=SuperAdmin
 - Cliquer sur le bouton **Créer note de frais**
 - Ajouter une ligne de frais de transport à 150€
 - Ajouter une ligne de frais de Autres à 30€
 - Vérifier que le Montant TTC est 180,00 €
 - Cliquer sur enregistrer
 - Se déconnecter

1.3 Gestion des objets

Utiliser des expressions régulières

- Regexp

Command	Target	Value
clickAndWait	link=search	
verifyValue	id=name	regexp:[Tt]ax ([Yy]ear)

- « ^ » = commence par
- « \$ » = se termine par
- « . » = remplace tout caractère, le « joker ».
- « * » = correspond avec 0 ou plusieurs caractères précédents
- « + » = correspond avec 1 ou plusieurs caractères précédents
- « ? » = correspond avec 0 ou 1 caractère précédent.
- « [] » = correspond à un caractère dans une liste
- « [-] » = correspond à un caractère dans une série
- « [^abc] » = ne correspond à aucun caractère de la série
- « {} » = répète le précédent caractère
-

1.3 Gestion des objets

Exercice IDE-E2: Utiliser une expression régulière

- Créer le scénario suivant:
- Lancer Chrome: <http://dolibarr.selenium-formation.org/>
- Saisir **Username**=jsmith
- Saisir **Password**=dolibarrhp
- Cliquer sur **Tiers**
- Cliquer sur **Nouveau Tiers**
- Vérifier la valeur du champ **code client** respecte bien le format prédéfini
- Stocker cette valeur dans une variable code_client
- Sélectionner **Fournisseur**=Non
- Cliquer sur **créer le tiers**
- Cliquer sur Liste Clients
- Rechercher en utilisant le code_client précédent
- Vérifier que la recherche a ramené le résultat

1.4 Fonctionnalités avancées

Pilotage par les données

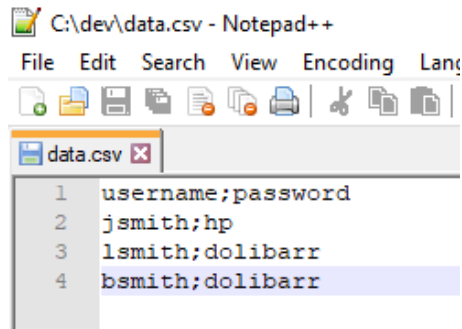
- Cette approche isole les données (entrées de test) à l'aide d'un tableur et utilise un script plus générique qui peut les lire et effectuer le même test avec des données différentes.
- Même les testeurs non familiarisés avec le langage de scripts peuvent entrer des données de tests pour ces scripts prédéfinis.



1.4 Fonctionnalités avancées

Pilotage par les données

- Ajout du csv

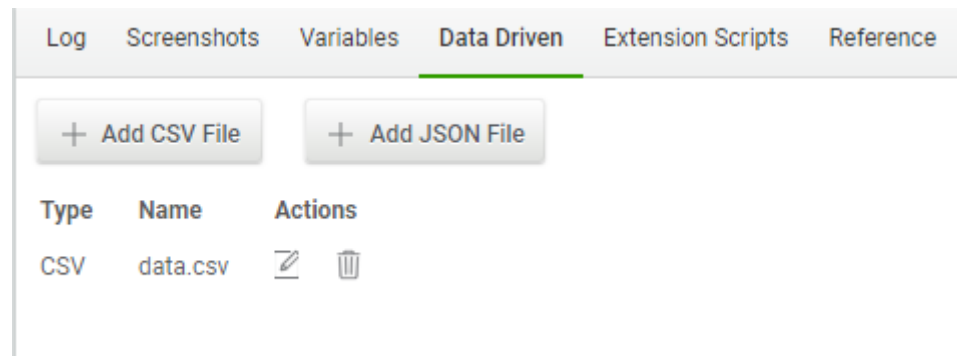


C:\dev\data.csv - Notepad++

File Edit Search View Encoding Lang



data.csv

```
1 username;password
2 jsmith;hp
3 lsmith;dolibarr
4 bsmith;dolibarr
```



Log Screenshots Variables **Data Driven** Extension Scripts Reference

+ Add CSV File + Add JSON File

Type	Name	Actions
CSV	data.csv	 

1.4 Fonctionnalités avancées

Pilotage par les données

- Chargement des données

Command	Target	Value
loadVars	data.csv	
open	http://demo.testlogiciel.pro/dolibarr/index.php	
type	id=username	\$(username)
type	id=password	\$(password)
click	css=input.button	
click	//img[@alt='Déconnexion']	
endLoadVars	data.csv	

Chargement du fichier

Utilisation des variables=colonne du fichier csv

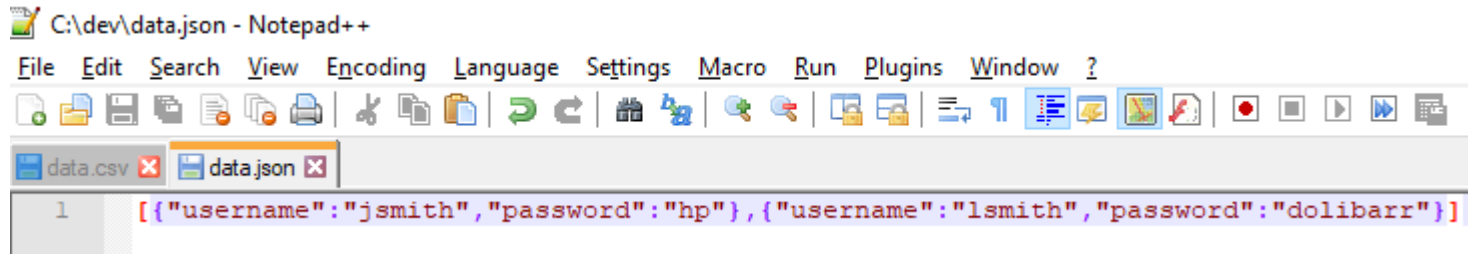
```
1 username;password
2 jsmith;hp
3 lsmith;dolibarr
4 bsmith;dolibarr
```

Libération du fichier

1.4 Fonctionnalités avancées

Pilotage par les données

- Ajout json



```
C:\dev\data.json - Notepad++  
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?  
data.csv data.json  
1 [{"username": "jsmith", "password": "hp"}, {"username": "lsmith", "password": "dolibarr"}]
```

Command	Target	Value
loadVars	data.json	
open	http://demo.testlogiciel.pro/dolibarr/index.php	
type	id=username	\${username}
type	id=password	\${password}
click	css=input.button	
click	//img[@alt="Déconnexion"]	

1.4 Fonctionnalités avancées

Utilisation des expressions régulières

- regexp

Command	Target	Value
clickAndWait	link=search	
verifyValue	id=name	regexp:[Tt]ax ([Yy]ear)

- « ^ » = commence par
- « \$ » = se termine par
- « . » = remplace tout caractère, le « joker ».
- « * » = correspond avec 0 ou plusieurs caractères précédents
- « + » = correspond avec 1 ou plusieurs caractères précédents
- « ? » = correspond avec 0 ou 1 caractère précédent.
- « [] » = correspond à un caractère dans une liste
- « [-] » = correspond à un caractère dans une série
- « [^abc] » = ne correspond à aucun caractère de la série
- « {} » = répète le précédent caractère

1.4 Fonctionnalités avancées

Utilisation de fonctions **Javascript** pour traiter les variables

StoreEval: exécution de commandes javascript et stockage dans une variable

RunScript: exécution de commandes javascript

store	100	AAA
store	20	BBB
storeEval	\${AAA}-\${BBB}	CCC
storeEval	document.title = \${CCC};	
assertTitle	80	

1.4 Fonctionnalités avancées

Ajouter une extension Javascript

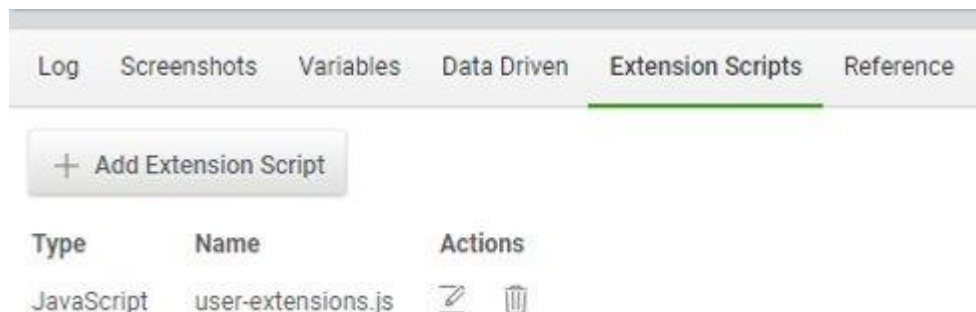
Ajout de nouvelle commande selenese

- Il est possible de coder des nouvelles commandes en javascript
- Création du fichier js: user-extensions.js

```
Selenium.prototype.doTypeRandomEmail = function(locator) {  
  // All locator-strategies are automatically handled by "findElement"  
  var element = this.page().findElement(locator);  
  /* The following block generates a random email string */  
  var allowedChars = "abcdefghijklmnopqrstuvwxyz";  
  var stringLength = 8;  
  var randomstring = '';  
  
  for (var i=0; i<stringLength; i++) {  
    var rnum = Math.floor(Math.random() * allowedChars.length);  
    randomstring += allowedChars.substring(rnum,rnum+1);  
  }  
  // Append a domain name  
  randomstring += "@test.com"  
  // Replace the element text with the new text  
  this.browserbot.replaceText(element, randomstring);  
};
```

typeRandomEmail

id=username



1.4 Fonctionnalités avancées

Exercice IDE-E1: Pilotage par un fichier csv

- Créer le scénario suivant:
 - Lancer l'application à l'adresse: <http://orangehrm.selenium-formation.org>
 - Saisir **Nom d'utilisateur**=jsmith
 - Saisir **Mot de passe**=Selenium&2018
 - Cliquer sur **Connexion**
 - Vérifier que vous êtes sur le **Tableau de bord**
 - Cliquer sur **Recrutement**
 - Utiliser un fichier csv pour créer plusieurs candidats pour le poste Selenium Junior
 - Cliquer sur le bouton **Ajouter**
 - Saisir **prénom**
 - Saisir **Nom de famille**
 - Saisir **Email**
 - Cliquer sur **Sauvegarder**
 - Cliquer sur **Retour**
 - Dans le formulaire de recherche : Saisir le nom du candidat
 - Cliquer sur **Rechercher**
 - Vérifier dans le tableau résultat que le statut est Candidature soumise
 - Cliquer sur Déconnexion

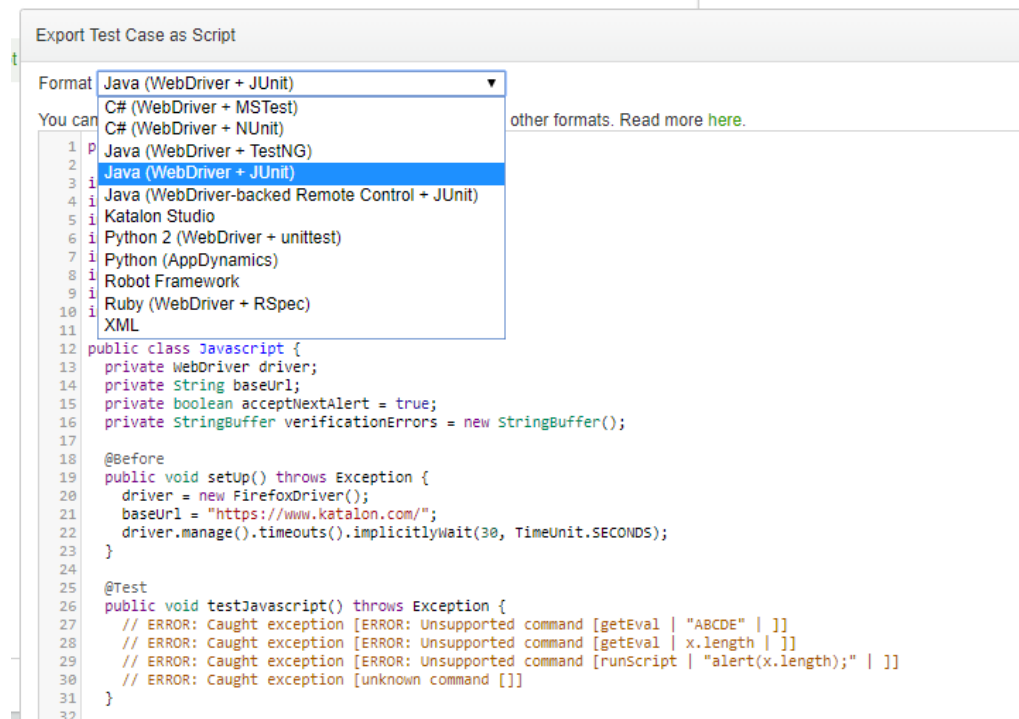
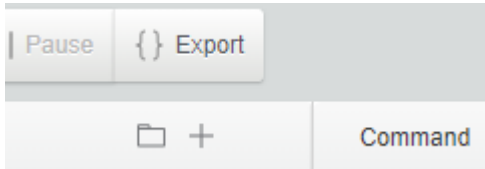
1.4 Fonctionnalités avancées

Exercice IDE-E2: Utiliser une expression régulière

- Créer le scénario suivant:
- Lancer Chrome: <http://dolibarr.selenium-formation.org/>
- Saisir **Username**=jsmith
- Saisir **Password**=dolibarrhp
- Cliquer sur **Tiers**
- Cliquer sur **Nouveau Tiers**
- Vérifier la valeur du champ **code client** respecte bien le format prédéfini
- Stocker cette valeur dans une variable `code_client`
- Sélectionner **Fournisseur**=Non
- Cliquer sur **créer le tiers**
- Cliquer sur Liste Clients
- Rechercher en utilisant le `code_client` précédent
- Vérifier que la recherche a ramené le résultat

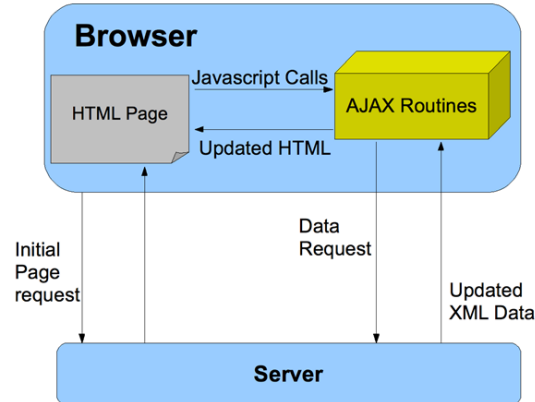
1.4 Fonctionnalités avancées

Il est possible d'exporter le script dans un langage de développement



1.4 Fonctionnalités avancées

- Asynchronous Javascript And XML
 - Ajax permet la mise à jour des pages HTML de façon asynchrone en échangeant des données en XML/JSON avec le serveur
 - AJAX permet de mettre à jour une page Web sans rechargement de toute la page



Problématiques AJAX pour Selenium

- Problème de synchronisation
- Problème d'exécution d'évènement Javascript



1.4 Fonctionnalités avancées

AJAX: Synchronisation WaitFor

`waitForSelectedValue`
`waitForSelectedValues`
`waitForSomethingSelected`
`waitForSpeed`
`waitForTable`
`waitForText`
`waitForTextNotPresent`
`waitForTextPresent`
`waitForTitle`
`waitForValue`

Chap. 1: Travaux pratiques

Mettre en pratique sur un scénario de test de votre application



Chap. 2: Selenium WebDriver

- Selenium WebDriver et Java
- Exécuter un test Junit
- Selenium Grid
- Exécution à partir de Jenkins

2.1 WebDriver

- WebDriver est conçu pour fournir une interface de programmation plus simple, plus concise en plus de répondre à certaines limitations dans l'API selenium-RC.
- Selenium-WebDriver a été développé afin de mieux gérer des problématiques difficilement gérable avec Selenium IDE
- Offre toutes les possibilités de programmation des langages de développement (Java, C#, Ruby, Python)



2.1 WebDriver

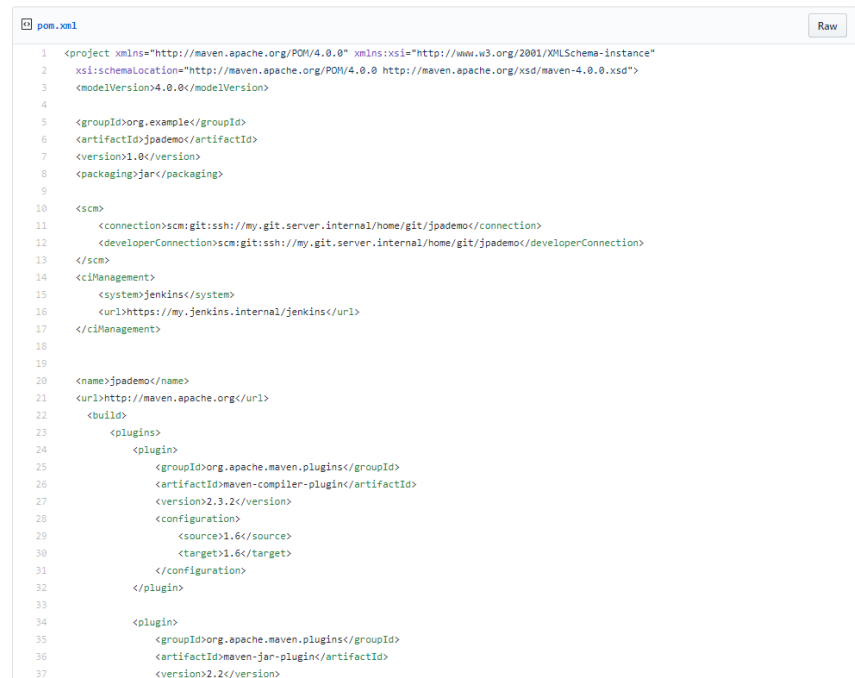
- Solution de gestion d'un build d'un projet Java
- Objectifs
 - Définir un standard de build d'un projet java
 - Documenter le projet
 - Partager les librairies entre les projets
 - Eviter de stocker dans les outils SCM les librairies

Un projet Maven

- Fichiers du code sources
- Fichiers de configuration
- Licences
- Fichiers de ressources
- Dépendances

POM: Project Object Model

- Fichier Maven
- Normalise et décrit le projet

A screenshot of a code editor showing a Maven Project Object Model (POM) file named 'pom.xml'. The file is displayed with line numbers from 1 to 37. The XML content defines a project with the following details: groupId 'org.example', artifactId 'jpademo', version '1.0', and packaging 'jar'. It includes a source control management (scm) section with a connection to a local git repository and a developer connection. The ciManagement section specifies a Jenkins system with a URL. The build section includes a plugin for the Maven Compiler Plugin, configured with groupId 'org.apache.maven.plugins', artifactId 'maven-compiler-plugin', version '2.3.2', and a configuration for source and target Java versions of 1.6. The file also includes a dependency on the Maven Jar Plugin with groupId 'org.apache.maven.plugins', artifactId 'maven-jar-plugin', and version '2.2'.

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>org.example</groupId>
6   <artifactId>jpademo</artifactId>
7   <version>1.0</version>
8   <packaging>jar</packaging>
9
10  <scm>
11    <connection>scm:git:ssh://my.git.server.internal/home/git/jpademo</connection>
12    <developerConnection>scm:git:ssh://my.git.server.internal/home/git/jpademo</developerConnection>
13  </scm>
14  <ciManagement>
15    <system>jenkins</system>
16    <url>https://my.jenkins.internal/jenkins</url>
17  </ciManagement>
18
19
20  <name>jpademo</name>
21  <url>http://maven.apache.org</url>
22  <build>
23    <plugins>
24      <plugin>
25        <groupId>org.apache.maven.plugins</groupId>
26        <artifactId>maven-compiler-plugin</artifactId>
27        <version>2.3.2</version>
28        <configuration>
29          <source>1.6</source>
30          <target>1.6</target>
31        </configuration>
32      </plugin>
33
34      <plugin>
35        <groupId>org.apache.maven.plugins</groupId>
36        <artifactId>maven-jar-plugin</artifactId>
37        <version>2.2</version>
```

2.1 WebDriver

- Maven dependencies

- groupId: **org.seleniumhq.selenium**
- artifactId: **selenium-java**
- Version: **3.14.0**

- Driver

- Exécutable: outil nécessaire pour communiquer les opérations Selenium aux différents browsers
- Les fichiers exécutables doivent être accessibles
 - IE: IEDriverServer.exe
 - Chrome: chromedriver.exe

- WebDriver

- Classe permettant de piloter le driver et les commandes Selenium
- Une classe WebDriver par browser
 - ChromeDriver
 - InternetExplorerDriver

```
@Before
public void setUp() throws Exception {
    driver = new ChromeDriver();
}
```

2.1 WebDriver: JUnit

- Framework de test unitaire

Outil fournissant un environnement pour tests unitaires ou de composant dans lequel un composant peut être testé de façon isolée ou avec des bouchons ou pilotes appropriés.

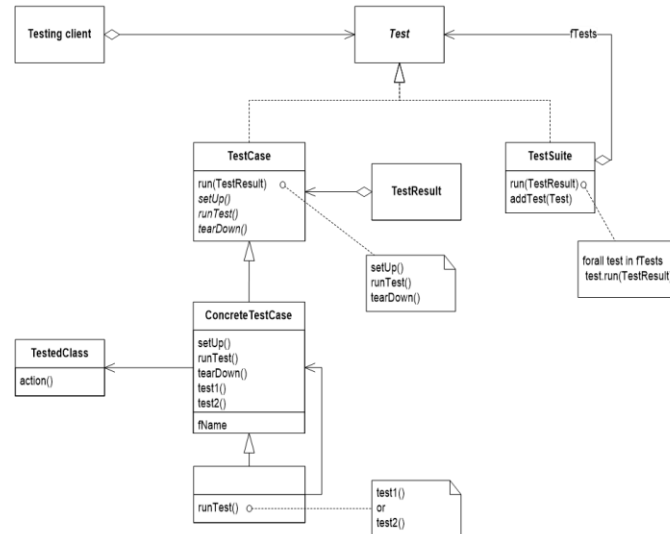
Pourquoi?

- Il facilite la conception et l'écriture des tests unitaires et d'intégration bas niveau
- Il permet le pilotage de l'exécution des tests ainsi que de l'exploitation des résultats des test
- Méthode originelle de l'approche Test Driven Development

2.1 WebDriver: JUnit

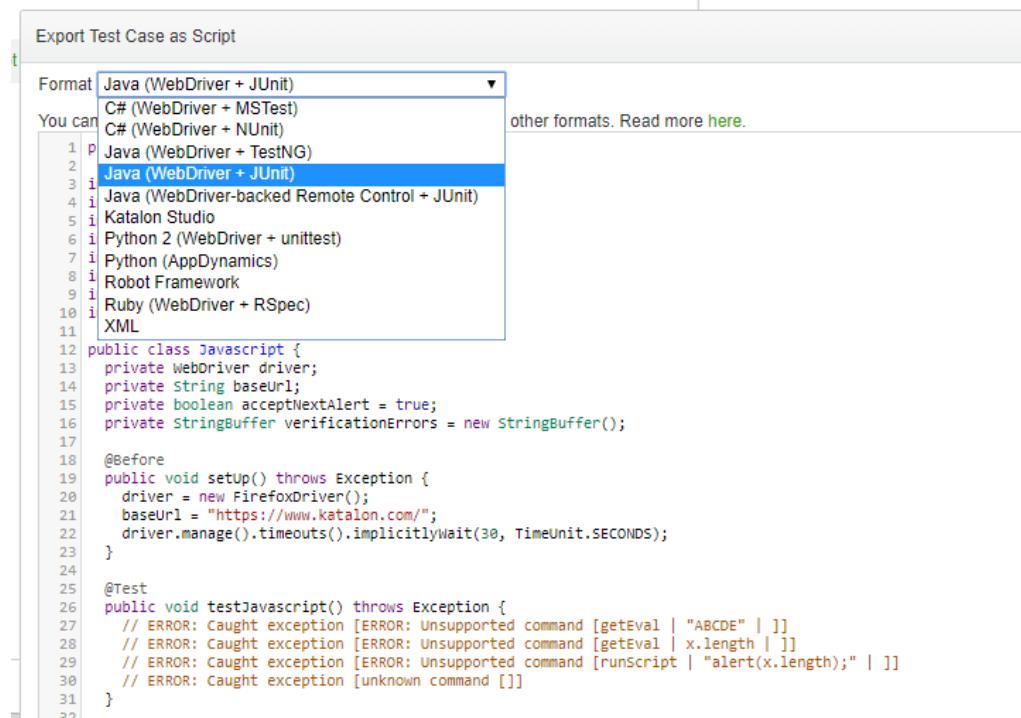
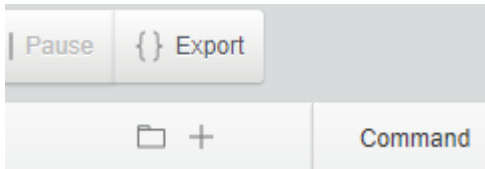
Architecture

- Un test unitaire correspond au test d'une classe
- Un cas de test vérifie le comportement d'une méthode
- Une suite de test est un ensemble de cas de test
- Le Test Runner exécute les tests et suites de test



2.1 WebDriver: JUnit

Il est possible d'exporter le script dans un langage de développement



2.1 WebDriver: JUnit

Rappel de la structure d'un test unitaire

- Import des librairies
- Classe de test suite
 - Méthode pour les CI de la suite de test
 - Méthode pour les CI pour un test
 - Méthodes pour les tests
 - Annotations pour le paramétrage des données
 - Méthode pour les CF d'un test
 - Méthode pour les CF de la suite de test

2.1 WebDriver: JUnit

Structure d'un test

- **BeforeClass**: méthode pour l'initialisation de la classe
 - **Before**: méthode exécutée avant chaque test
 - **Test**: méthode de test
 - **After**: méthode exécutée après chaque test
 - **AfterClass**: méthode exécutée à la destruction de la classe de test
-
- Un test ne retourne pas d'objet
 - Un test en échec renvoie une assertion fausse

2.1 WebDriver: librairies

Objet WebDriver

- Classe permettant de piloter le driver et les commandes Selenium
- Une classe WebDriver par browser
- ChromeDriver
- InternetExplorerDriver

```
@BeforeMethod
public void setup(){
    driver = new ChromeDriver();
}
```

2.1 WebDriver: Initialisation

Principales fonctions de configuration du Driver

- `driver.manage().window().maximize();`
- `driver.manage().window().fullscreen();`

Synchronisation

Synchronisation par défaut: Implicit Wait

On peut définir un temps de synchronisation par défaut avant d'envoyer une erreur pour une commande du driver

Navigation

- `driver.get("http://newtours.demoaut.com/");`
- `driver.navigate().to("http://newtours.demoaut.com/");`

Fermeture

- `driver.quit();`

```
driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);
```

```
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS); // Synchronisation de 5 Seconds
```

2.1 WebDriver: Initialisation

Exercice A

- Créer une classe ExerciceA
- Créer une méthode lancementChromeDriver
- Lancer chrome en maximized
- Aller à l'adresse: <http://orangehrm.selenium-formation.org/>
- Attendre 5 secondes

Utiliser la méthode: *Thread.sleep(5000);*

- Fermer le browser
- Créer une méthode lancementIEDriver et faire de même avec Internet Explorer

2.1 WebDriver: Commandes

Actions sur le browser avec un FindElementHelper

- Les commandes Selenium sont opérées sur des objets
- Il faut identifier l'élément avant
- Puis appeler la méthode correspondante



```
driver.findElementById("username").sendKeys("jsmith");  
driver.findElementByName("password").sendKeys("dolibarrhp");  
driver.findElementByClassName("button").click();
```

```
driver.get("http://demo.testlogiciel.pro/dolibarr");  
driver.findElement(By.id("username")).sendKeys("m00000");  
driver.findElement(By.id("password")).sendKeys("secret");
```

```
// Navigation à la page de connexion  
// Saisie du Login: m00000  
// Saisie du Mot de passe: secret
```

2.1 WebDriver: Commandes

Actions sur le browser avec FindElement

```
driver.get("http://demo.testlogiciel.pro/dolibarr");  
driver.findElement(By.id("username")).sendKeys("m000000");  
driver.findElement(By.id("password")).sendKeys("secret");
```

// Navigation à la page de connexion
// Saisie du Login: m000000
// Saisie du Mot de passe: secret

2.1 WebDriver: Commandes

Identification des objets

- Locator By: permet d'identifier les objets **WebElement** en se basant sur différentes propriétés d'identification:

- Par **id**: en se basant sur la valeur de l'attribut id

```
WebElement txtLogin = driver.findElement(By.id("username"));
```

- Par name: en se basant sur la valeur de l'attribut name

```
WebElement txtLogin = driver.findElement(By.name("username"));
```

- Par **classe**: en se basant sur la valeur de l'attribut class

```
WebElement txtLogin = driver.findElement(By.className("loginfield"));
```

- Par le nom du lien: en se basant sur la valeur du lien

```
WebElement linkAssistance = driver.findElement(By.linkText("Besoin d'assistance ou aide ?"));
```

```
WebElement linkAssistance = driver.findElement(By.partialLinkText("Besoin d'assistance"));
```

- Par le css: en se basant sur un sélecteur css

```
WebElement btnConnexion = driver.findElement(By.cssSelector("input.button"));
```

- Par le xpath: en se basant sur une requête xpath

```
WebElement txtLogin = driver.findElement(By.xpath("//input[@id='username']"));
```


2.1 WebDriver: Commandes

Actions sur les objets

- **sendKeys(CharSequence... texte)**: simule la saisie du texte dans un élément de type entrée de saisie

```
driver.findElement(By.id("username")).sendKeys("jsmith");
```

- **click()**: click sur l'élément

```
driver.findElement(By.cssSelector("input.button")).click();
```

- **clear()**: supprime la valeur pour un élément de type saisie

```
driver.findElement(By.id("username")).clear();
```


- **submit()**: soumettre un formulaire

```
driver.findElement(By.xpath("//form[@id='login']")).submit();
```

2.1 WebDriver: Commandes

Objet Select

```
new Select(driver.findElement(By.id("selectcountry_id"))).selectByVisibleText("France (FR)");
```

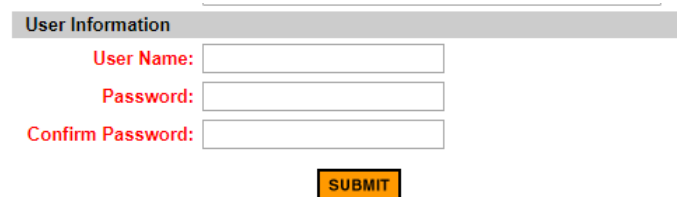


A screenshot of a web form. On the left, there is a label 'Pays' next to a text input field. To the right of this is a dropdown menu. The dropdown menu is open, showing a list of options, with 'France (FR)' selected and highlighted. A small downward arrow is visible on the right side of the dropdown menu.

2.1 WebDriver: Commandes

Exercice B1

- Créer une classe NewtoursSuite
- Créer une méthode de test registerAccount
- Lancer chrome en maximized
 - Aller à l'adresse: <http://newtours.demoaut.com>
 - Cliquer sur REGISTER
 - Sélectionner le pays au niveau du formulaire Country=France
 - Renseigner les informations du compte à créer
- Cliquer sur SIGN-OFF
- Fermer le browser
- Exécuter le test



A screenshot of a web form titled "User Information". The form contains three input fields: "User Name:", "Password:", and "Confirm Password:". Each label is in red text. Below the input fields is a yellow "SUBMIT" button.

2.1 WebDriver: Commandes

Exercice B2

- Créer une classe OrangeSuite
- Test: Ajouter un nouveau salarié

- Lancer Chrome: <http://orangehrm.selenium-formation.org>
- Saisir **Username**=admin
- Saisir **Password**=Selenium&2018
- Cliquer sur le bouton **Login**
- Cliquer sur **GIP**
- Cliquer sur le bouton **Ajouter**
- Saisir les informations suivantes:
 - **Prénom**=....
 - **Nom du milieu**=....
 - **Nom de famille**=....
- Cliquer sur **Sauver**
- Cliquer sur **Editer**
- Saisir les informations suivantes:
 - **Numéro du permis**=XXXXXX
 - **Sexe**=Masculin
 - **Etat marital**=Marié
 - **Nationalité**=Français
 - **Date de naissance**=2000-01-01
- Cliquer sur **Sauver**
- Cliquer **Welcome Admin**
- Cliquer sur **Déconnexion**

2.1 WebDriver: Assertions

Méthode	Rôle
<code>assertEquals()</code>	Vérifier l'égalité de deux valeurs de type primitif ou objet.
<code>assertFalse()</code>	Vérifier que la valeur fournie en paramètre est fausse
<code>assertNull()</code>	Vérifier que l'objet fourni en paramètre soit null
<code>assertNotNull()</code>	Vérifier que l'objet fourni en paramètre ne soit pas null
<code>assertSame()</code>	Vérifier que les deux objets fournis en paramètre font référence à la même entité Exemples identiques : <code>assertSame("Les deux objets sont identiques", obj1, obj2);</code> <code>assertTrue("Les deux objets sont identiques ", obj1 == obj2);</code>
<code>assertNotSame()</code>	Vérifier que les deux objets fournis en paramètre ne font pas référence à la même entité
<code>assertTrue()</code>	Vérifier que la valeur fournie en paramètre est vraie
<code>fail()</code>	Entraîne un échec du test. Elle renvoie une exception <code>AssertionFailedError</code>

```
assertEquals("Iphone X", driver.findElement(By.id("article")).getText());
```

2.1 WebDriver: Assertions

Vérifier le texte d'un élément

```
assertEquals("test@email.fr", driver.findElement(By.id("emailPanel")).getText());
```

Vérifier la valeur d'un input....

```
assertEquals("test@email.fr", driver.findElement(By.id("email")).getAttribute("value"));
```

Vérifier qu'un élément est présent

```
assertTrue(driver.findElement(By.id("email")).isDisplayed());
```

Vérifier la sélection d'une liste

```
assertEquals("FRANCE", new Select(driver.findElement(By.id("country"))).getFirstSelectedOption().getText());
```

Vérifier qu'un texte est affiché dans le page

```
assertTrue(driver.findElement(By.tagName("body")).getText().contains("Le texte recherché"));
```

2.1 WebDriver: Assertion

Exercice D1

Reprendre la suite OrangeDemo

Ajouter un nouveau test: Scénario: Créer un poste

- Lancer l'application à l'adresse : <http://orangehrm.selenium-formation.org>
- Vérifier la présence du champ « Nom d'utilisateur »
- Saisir **Nom d'utilisateur**=admin
- Saisir **Mot de passe**=Selenium&2018
- Cliquer sur **Connexion**
 - Vérifier que vous êtes sur le Tableau de bord
- Cliquer sur **Recrutement**
- Cliquer sur **Offres d'emploi**
- Cliquer sur le bouton **Ajouter**
 - Vérifier que la valeur par défaut de Titre du poste est= « --Sélectionner --»
 - Vérifier sur Actif est coché
- Sélectionner un **Titre de poste**
- Saisir **Nom du poste vacant** : ...
- Saisir **Manager qui recrute** : John SMITH
- Nombre de postes : 3
- Décoché Publier dans Flux RSS
- Cliquer sur **Sauver**
 - Vérifier que vous êtes sur la page : Modifier les postes vacants
- Cliquer sur **Retour**
 - Vérifier que le bouton rechercher est présent
- Sélectionner Vacant=nom du poste
- Cliquer sur Rechercher
 - Vérifier que dans le tableau, la valeur de la 2è ligne 2è colonne est le nom du poste
- Prendre une capture d'écran
- Cliquer sur Déconnexion

2.2 Chromedriver

Configuration Chrome

- Classe Option: ChromeOptions
 - Elle permet de définir les options du browser
 - On rajoute chaque option en argument

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--disable-extensions"); // Désactive les extensions Chrome
options.addArguments("start-maximized");     // Lance le browser en maximize
driver = new ChromeDriver(options);           // Lancement du driver avec les options
```

- Liste des arguments: <http://www.assertselenium.com/java/list-of-chrome-driver-command-line-arguments/>
- <https://sites.google.com/a/chromium.org/chromedriver/capabilities#TOC-List-of-recognized-capabilities>

```
ChromeOptions options = new ChromeOptions();

// Add the WebDriver proxy capability.
Proxy proxy = new Proxy();
proxy.setHttpProxy("myhttpproxy:3337");
options.setCapability("proxy", proxy);

// Add a ChromeDriver-specific capability.
options.addExtensions(new File("/path/to/extension.crx"));
ChromeDriver driver = new ChromeDriver(options);
```

```
ChromeOptions options = new ChromeOptions();
options.addArguments("user-data-dir=/path/to/your/custom/profile");
```


2.2 InternetExplorerDriver

Configuration IE

- Mode protégé désactivé
- Garder le zoom et la taille des textes affichés à 100%
- <https://github.com/SeleniumHQ/selenium/wiki/InternetExplorerDriver>

```
InternetExplorerOptions ieOptions = new InternetExplorerOptions();  
ieOptions.setCapability(InternetExplorerDriver.INITIAL_BROWSER_URL, "http://demo.testlogiciel.pro");  
WebDriver driver = new InternetExplorerDriver(ieOptions);
```

2.2 OperaDriver

OperaDriver

```
OperaOptions options = new OperaOptions();  
options.setBinary("C:\\Users\\pc\\AppData\\Local\\Programs\\Opera\\56.0.3051.40\\opera.exe");  
driver = new OperaDriver(options);
```

OperaDriver est similaire à ChromeDriver → **Les options sont similaires**

2.2 Headless

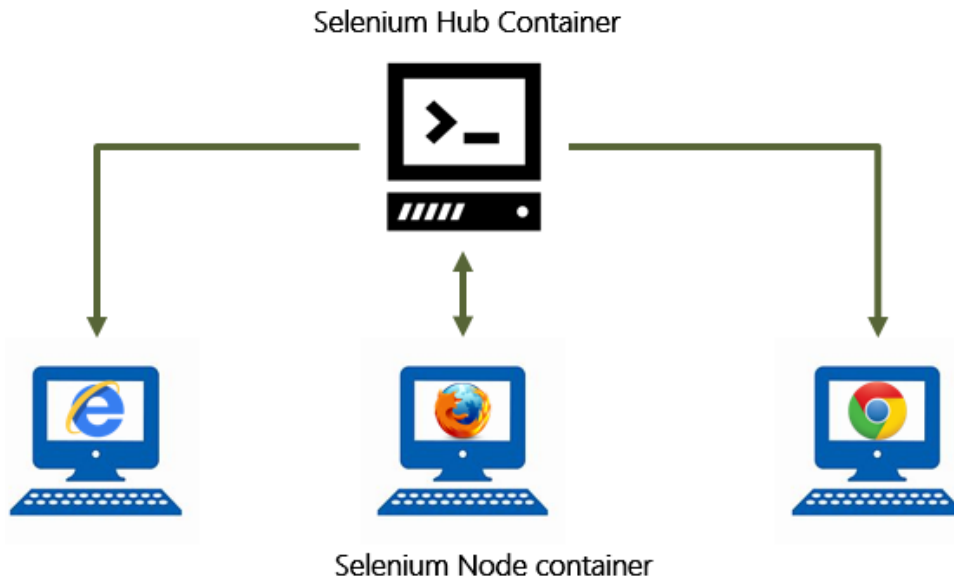
Chrome, Firefox

```
ChromeOptions chromeOptions=new ChromeOptions();  
chromeOptions.addArguments("headless");  
return new ChromeDriver(chromeOptions);
```

2.2 Selenium Grid

Serveur d'exécution des scripts Selenium

- Permet de tester la portabilité des applications:
 - Sur différentes configurations OS
 - Sur différents navigateurs
- Utile pour lancer les tests Selenium à partir d'une plateforme d'intégration continue
- Exécution en parallèle



- Un Hub Selenium référence plusieurs nœuds
- Chaque nœud correspond à une configuration cliente (OS-Browser) => Capability

2.2 Selenium Grid

Installation du Hub

- Récupération du jar: <http://selenium-release.storage.googleapis.com/index.html>
- Exécution de la commande: `java -jar selenium-server-standalone-<version>.jar -role hub`

```
λ java -jar selenium-server-standalone-3.14.0.jar -role hub
17:06:21.257 INFO [GridLauncherV3.launch] - Selenium build info: version: '3.14.0', revision: 'aaccce0'
17:06:21.268 INFO [GridLauncherV3$2.launch] - Launching Selenium Grid hub on port 4444
2018-10-14 17:06:21.952:INFO::main: Logging initialized @1275ms to org.seleniumhq.jetty9.util.log.StdErrLog
17:06:22.672 INFO [Hub.start] - Selenium Grid hub is up and running
17:06:22.672 INFO [Hub.start] - Nodes should register to http://192.168.146.1:4444/grid/register/
17:06:22.673 INFO [Hub.start] - Clients should connect to http://192.168.146.1:4444/wd/hub
```

localhost:4444



Selenium Grid Hub v.3.14.0

Whoops! The URL specified routes to this help page.

For more information about Selenium Grid Hub please see the [docs](#) and/or visit the [wiki](#). Or perhaps you are looking for the Selenium Grid Hub [console](#).

Happy Testing!

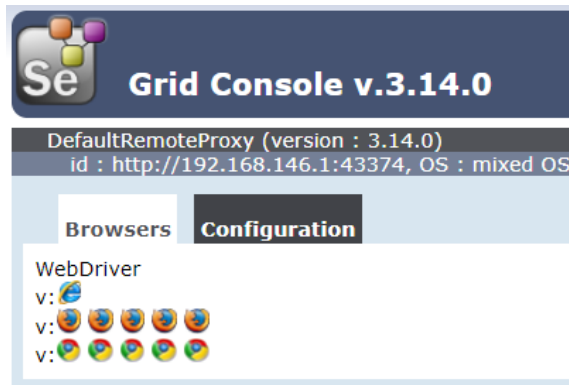
2.2 Selenium Grid

Installation d'un noeud

- Configurer le node avec les drivers
- Exécution de la commande:

```
java -jar selenium-server-standalone-<version>.jar -role node -hub http://localhost:4444/grid/register
```

```
λ java -jar selenium-server-standalone-3.14.0.jar -role node -hub http://localhost:4444/grid/register
17:17:27.821 INFO [GridLauncherV3.launch] - Selenium build info: version: '3.14.0', revision: 'aaccce0'
17:17:27.837 INFO [GridLauncherV3$3.launch] - Launching a Selenium Grid node on port 43374
2018-10-14 17:17:28.633:INFO::main: Logging initialized @1146ms to org.seleniumhq.jetty9.util.log.StdErrLog
17:17:28.868 INFO [SeleniumServer.boot] - Selenium Server is up and running on port 43374
17:17:28.868 INFO [GridLauncherV3$3.launch] - Selenium Grid node is up and ready to register to the hub
17:17:29.146 INFO [SelfRegisteringRemote$1.run] - Starting auto registration thread. Will try to register every 5000 ms.
17:17:29.146 INFO [SelfRegisteringRemote.registerToHub] - Registering the node to the hub: http://localhost:4444/grid/register
17:17:29.911 INFO [SelfRegisteringRemote.registerToHub] - The node is registered to the hub and ready to use
```



On peut configurer les noeuds selon les éléments en le précisant lors de l'enregistrement

```
java -jar selenium-server-standalone-<version>.jar -role node -hub http://localhost:4444/grid/register -browser browserName=firefox,version=3.6,maxInstances=5,platform=LINUX
```

[view config](#)

2.2 Selenium Grid

RemoteDriver

- Lancement des scripts sur un serveur distant
 - Sélection de la capacité

```
DesiredCapabilities capability = DesiredCapabilities.firefox();  
  
capability.setBrowserName("firefox" );  
capability.setPlatform("LINUX");  
capability.setVersion("3.6");
```

- Création du driver en Remote

```
WebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), capability);
```

```
String URL = "http://t3z66:4444/wd/hub";  
  
DesiredCapabilities caps = new DesiredCapabilities();  
caps.setCapability("os", "WINDOWS");  
caps.setCapability(CapabilityType.BROWSER_NAME, "internet explorer");  
caps.setCapability("browser", "internetexplorer");  
  
driver = new RemoteWebDriver(new URL(URL), caps);
```

2.2 Selenium Grid

RemoteDriver

Hub en ligne



Quick Launch	iPhone		iPad	
Android				
iOS	<input type="checkbox"/> iPhone 6S Plus	9	<input type="checkbox"/> iPad Air 2	9
	<input type="checkbox"/> iPhone 6S	9	<input type="checkbox"/> iPad Air	9
	<input type="checkbox"/> iPhone 6 Plus	8	<input type="checkbox"/> iPad 4	9
Windows Phone	<input type="checkbox"/> iPhone 6	8	<input type="checkbox"/> iPad Mini 3	9
	<input type="checkbox"/> iPhone 5S	7	<input type="checkbox"/> iPad Mini 2	7
Windows	iPhone 5	6	iPad Pro	9.3
10	iPhone 4S (WW)	6	iPad Air 2	9.3
8.1	iPhone 4S	5.1	iPad Air	8.1
8	iPhone 4	4	iPad Mini 4	9.1
7	iPhone 3GS	3	iPad Mini 2	8.1
XP			iPad Mini	7
Mac				

2.2 Selenium Grid

Piloter l'exécution sur les navigateurs par les variables d'environnement

Une classe pour gérer le Driver selon les variables

```
>mvn test -Dlocal=false -Dbrowser=chrome
```

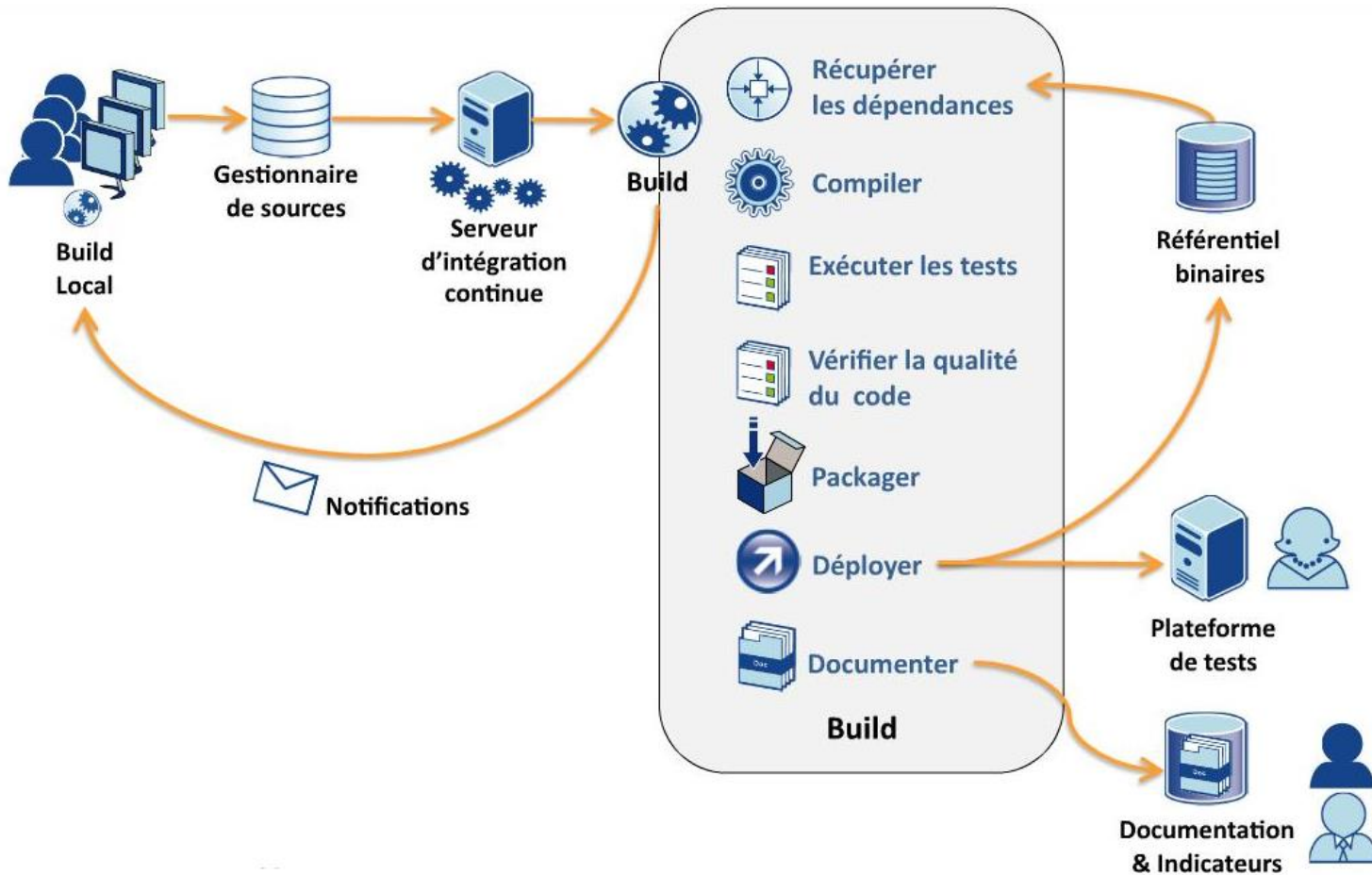
2.2 Selenium Grid

TP: Mettre une architecture en place



2.3 Jenkins

Serveur d'intégration continue et ordonnanceur



2.3 Jenkins

Job Jenkins: exécuter une suite de traitement

Git

Repositories

Build periodically

Schedule

Project name

lcgcmake-aarch64

Description

LCG nightly builds (aarch64)

Repository URL

https://gitlab.cern.ch/jwsmith/lcgcmake.git

Credentials

- none -

Add

H(1-10) 2 ***

Would last have run at Monday, January 25, 2016 2:05:48 AM UTC; would nex Tuesday, January 26, 2016 2:05:48 AM UTC.

String Parameter

Name

LCG_INSTALL_PREFIX

Default Value

/home/jwsmith/jenkins/releases

Description

Location to look for already installed packages

Execute shell

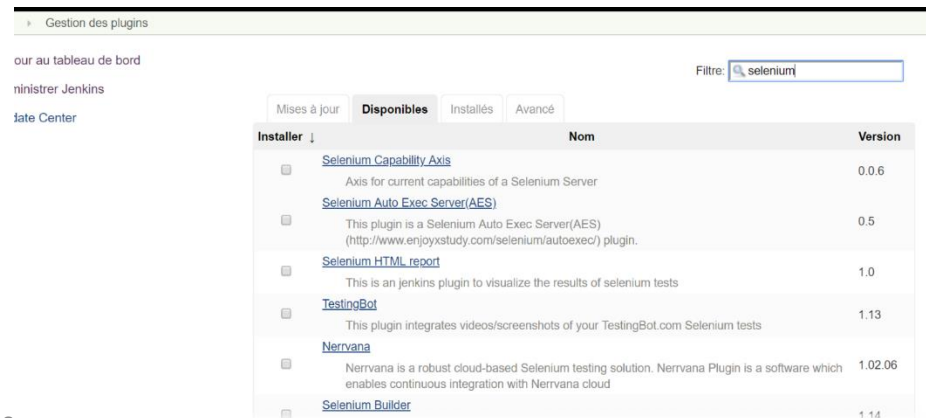
```
Command #!/bin/bash -x
mkdir build ; cd build
cmake -DCMAKE_INSTALL_PREFIX=../install ../lcgcmake
make -j12 -k
error_value=$?
####lcgcmake/cdash/isDone.sh 1
if [ $error_value -eq 0 ]
then
    echo "Successfully build"
    lcgcmake/cdash/isDone.sh 1
    exit 0
else
    echo "THE BUILD HAS FAILED *****"
    lcgcmake/cdash/isDone.sh 2
    exit 1
fi
cd ..
rm -rf build
rm -rf install
```

2.3 Jenkins

Plugin Jenkins: solution très connectée

Gestion des plugins

- Installation des plugins permettant de piloter les différentes étapes de l'intégration
 - Plugin de compilation
 - Plugin d'exécution et de reporting de test
 - Plugin de déploiement
 - Plugin de gestion de package
 - Plugin de notification
 - Plugin d'intégration avec des outils
 -



2.3 Maven

→ Simplifier et uniformiser le processus de build

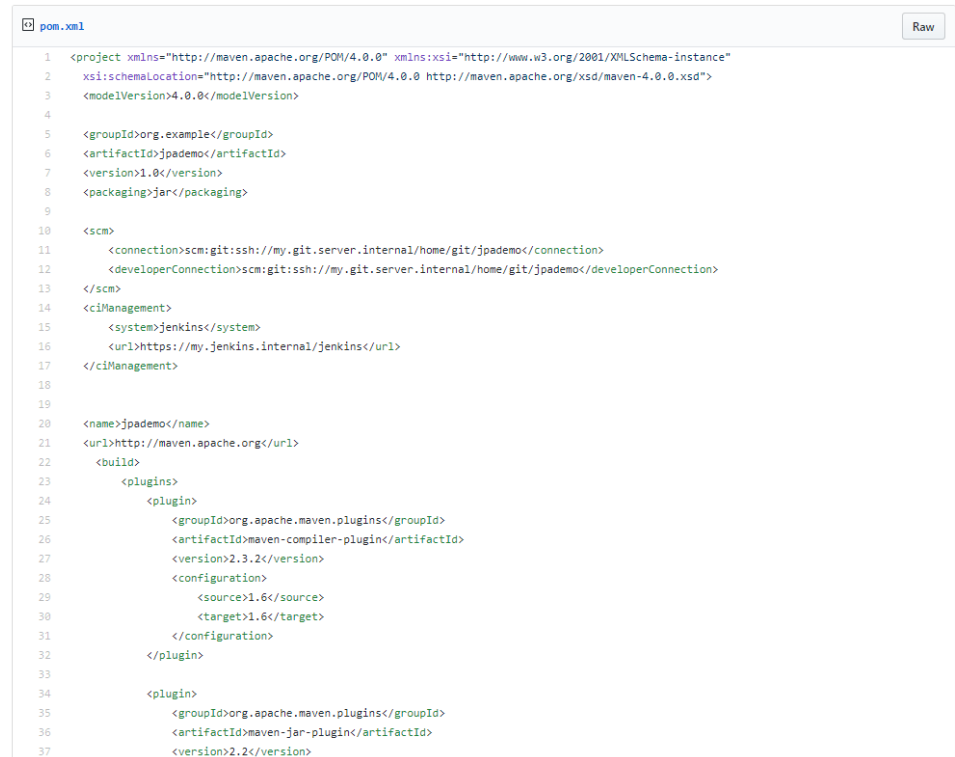
Va permettre de lancer l'exécution des tests selenium et générer le reporting

Un projet Maven

- Fichiers du code sources
- Fichiers de configuration
- Licences
- Fichiers de ressources
- Dépendances

POM: Project Object Model

- Fichier Maven
- Normalise et décrit le projet

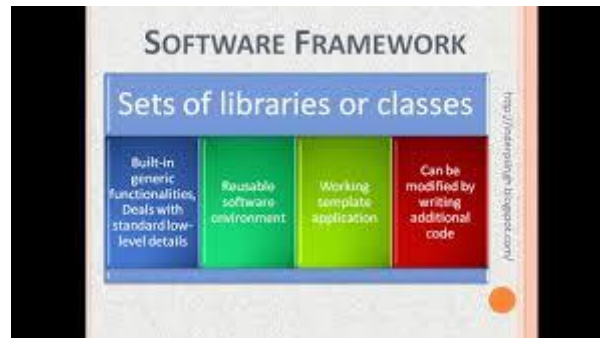
A screenshot of a code editor showing a Maven Project Object Model (POM) file named 'pom.xml'. The file is displayed with line numbers from 1 to 37. The XML content defines a project with the following details: groupId 'org.example', artifactId 'jpademo', version '1.0', and packaging 'jar'. It includes a source control management (scm) section with a connection to a local git repository and a developer connection. The ciManagement section specifies a Jenkins system with a URL. The build section contains two plugins: 'maven-compiler-plugin' with version '2.3.2' and configuration for source and target compatibility (1.6), and 'maven-jar-plugin' with version '2.2'.

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>org.example</groupId>
6   <artifactId>jpademo</artifactId>
7   <version>1.0</version>
8   <packaging>jar</packaging>
9
10  <scm>
11    <connection>scm:git:ssh://my.git.server.internal/home/git/jpademo</connection>
12    <developerConnection>scm:git:ssh://my.git.server.internal/home/git/jpademo</developerConnection>
13  </scm>
14  <ciManagement>
15    <system>jenkins</system>
16    <url>https://my.jenkins.internal/jenkins</url>
17  </ciManagement>
18
19  <name>jpademo</name>
20  <url>http://maven.apache.org</url>
21  <build>
22    <plugins>
23      <plugin>
24        <groupId>org.apache.maven.plugins</groupId>
25        <artifactId>maven-compiler-plugin</artifactId>
26        <version>2.3.2</version>
27        <configuration>
28          <source>1.6</source>
29          <target>1.6</target>
30        </configuration>
31      </plugin>
32    </plugins>
33
34    <plugin>
35      <groupId>org.apache.maven.plugins</groupId>
36      <artifactId>maven-jar-plugin</artifactId>
37      <version>2.2</version>
```

2.3 TP

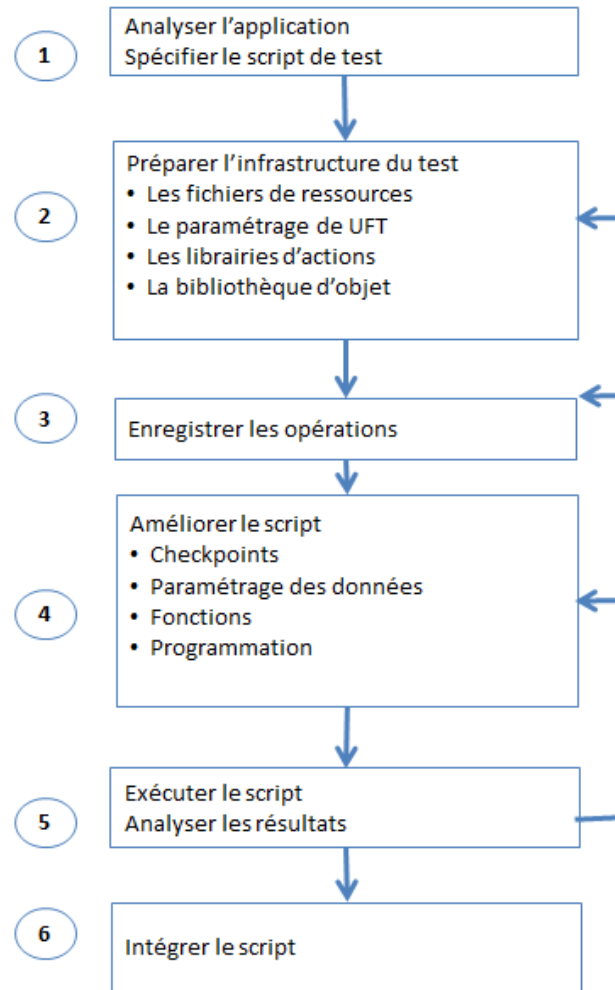
- Installation de Jenkins/Maven/Git
- Création d'un job pour lancer des scripts Selenium

Chap. 3: Framework



3. Framework

Workflow standard d'automatisation des tests:



3. Framework: Maintenir

Rappel de l'enjeu de la maintenance dans l'automatisation

Evolutions des applications

➔ Maintenance des scripts de test

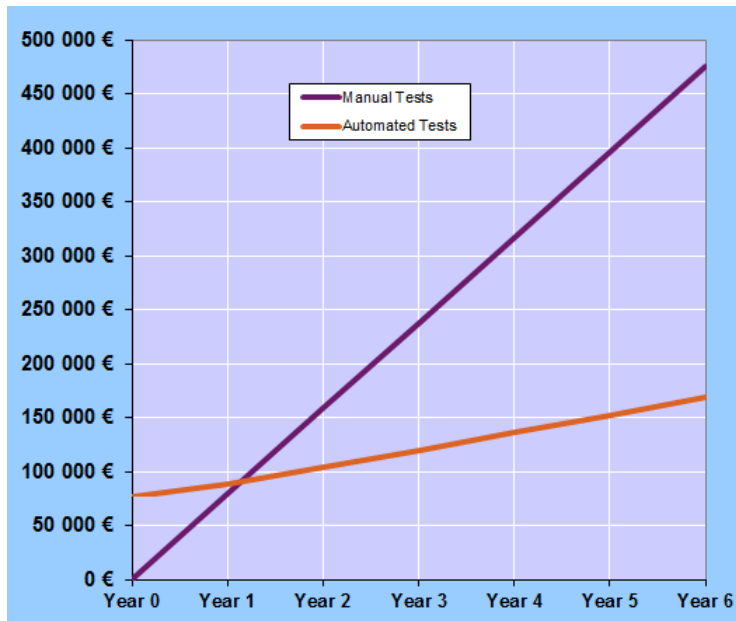
Processus de maintenance

- Évaluer les impacts potentiels suite à une évolution de l'application
- Identifier les scripts impactés
- Prendre une décision (corriger, repasser en manuel) pour les scripts impactés
- Modifier les scripts impactés
- Valider les scripts

3. Framework: Maintenir

Rappel de l'enjeu de la maintenance dans l'automatisation

- Le coût de la maintenance doit être pris en compte pour vérifier la rentabilité du script.
- Un référentiel difficile à maintenir n'est pas rentable par rapport à une exécution manuelle.
- Dans la démarche, il est important de définir une méthodologie impliquant un coût de maintenance le plus bas possible.



✈ Coût manuel du test :

$$\text{Coût (j/h)} = n \cdot \text{Exe M}$$

✈ Coût du test automatisé :

$$\text{Coût (j/h)} = n \cdot (\text{Exe A} + A + \text{MAI}) + \text{DEV}$$

✈ Seuil de rentabilité :

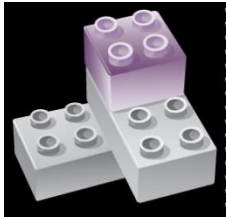
$$n = \text{DEV} / (\text{Exe M} - \text{Exe A} - A - \text{MAI})$$

3. Framework: Maintenir

Qu'est ce qu'un framework?

Un Framework (architecture) d'automatisation est un ensemble de choix, de concepts et de méthodes dont le but est d'organiser la création, l'utilisation et la réutilisation des scripts d'automatisation, et en faciliter la maintenance, dans un contexte donné.

Pourquoi construire un framework?

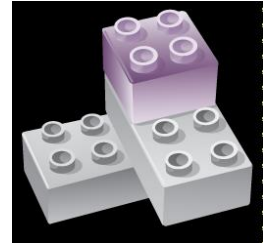


- Répondre aux problématiques soulevées (maintenance, robustesse)
- Permettre de dépasser certaines limites de l'outil d'automatisation
- Utiliser les fonctions de paramétrage pour répondre le plus efficacement à votre besoin
- Fournir une bibliothèque « ready-to-use » aux scripteurs
- Faciliter l'exécution des scénariis automatisés

3. Framework: Maintenir

Bénéfices

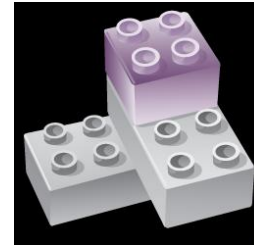
- Réduction de l'effort d'automatisation à la conception
- Réduction de l'effort d'automatisation à l'exécution
- Réduction de l'effort d'automatisation à la maintenance
- Faciliter le transfert de connaissance des scripts



3. Framework: Maintenir

Les éléments d'un framework

- Objets partagés
- Fonctions
- Actions réutilisables
- Données centralisées
- Scénario de recouvrement
- Paramétrages communs



3. Framework: Maintenir

Comment faciliter la maintenance?

2 principes permettent de faciliter cette maintenance

- **Réutilisation**
 - Mutualiser les objets communs facilite la maintenance.
 - La réutilisation: un atout majeur pour diminuer la maintenance.
- **Exclure les scénarios complexes à maintenir**
 - Un scénario complexe à automatiser peut être plus simple à exécuter manuellement



3. Framework: Architecture

Les différents types d'architecture

Une architecture d'automatisation est un référentiel de techniques, de concepts et de méthodes permettant d'organiser le processus d'automatisation des test dans l'optique de faciliter la maintenance notamment par la réutilisation d'objets (scripts, fonctions, données, ...) dans un contexte client spécifique.

3 types d'architectures se dégagent:

- **Data-driven:** les tests sont pilotés uniquement par les données externalisées
- **Framework-based:** Les tests sont divisés et organisés selon des composants
- **Executive specification/Mots-clés:** les tests sont dirigés par les spécifications fonctionnelles

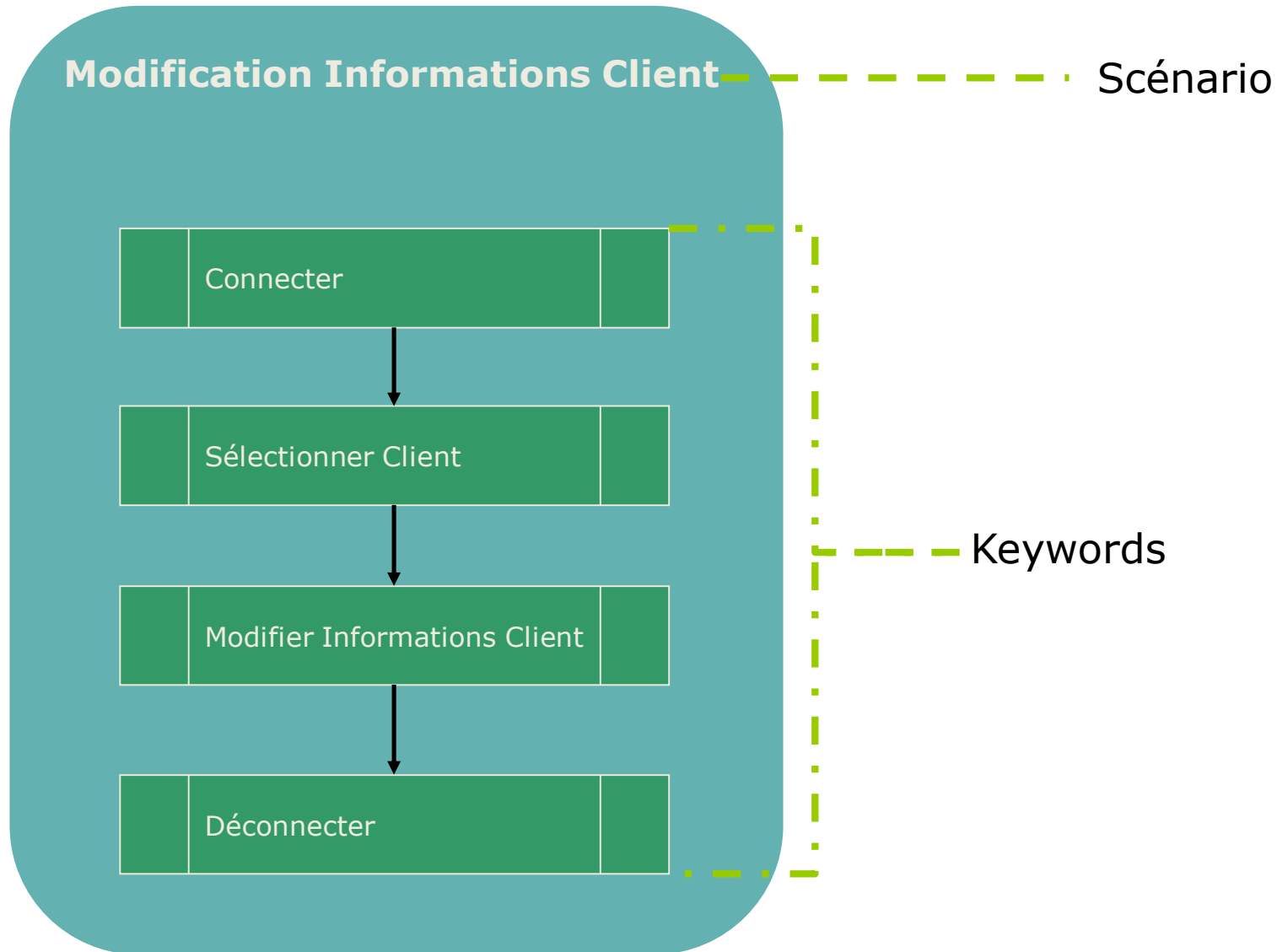
Il existe aussi des architectures hybrides

3. Framework: Architecture

Les composants d'une architecture d'automatisation

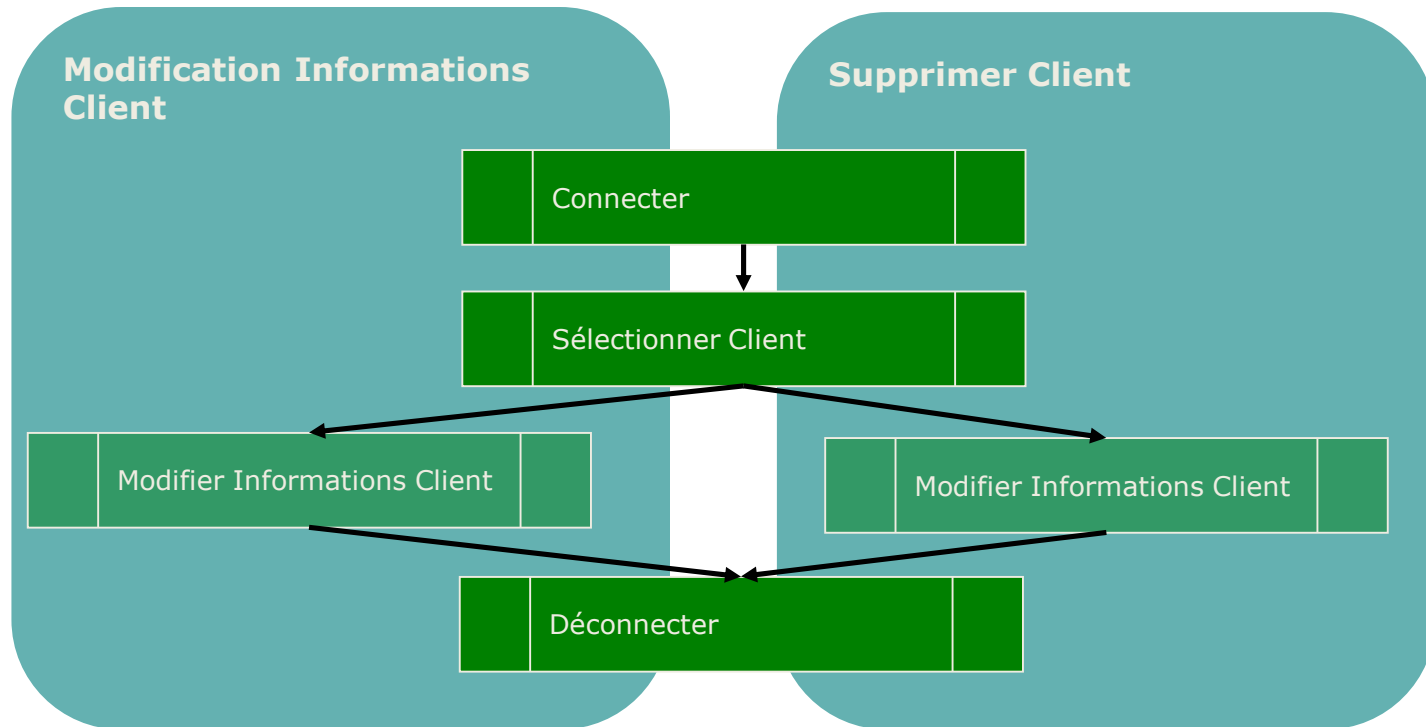
- L'externalisation des données de test/objets
 - Les données nécessaires à l'exécution du script dans un jeu dont la forme (compatibles avec l'outil) peut être un fichier Excel,XML,etc
 - Les données peuvent prendre en charge des résultats attendus à chaque point de vérification/contrôle
 - Les données nécessaires au contrôle des résultats des tests
- **Le développement de scripts génériques**
 - Script réutilisable
- **La mise en place de librairies de scripts, fonctions**
 - _Faciliter le référencement des scripts et donc leur réutilisation
 - A faciliter la maintenance et le développement des scripts, en adoptant une approche de développement modulaire

3. Framework: Keyword driven



3. Framework: Keyword driven

- Réduction du temps de l'automatisation



3. Framework: Keyword driven

- a. Définition d'un Framework
- b. Externalisation des objets
- c. Développement de composants génériques
- d. Externalisation des données et la gestion de données intelligentes
- e. La mise en place de bibliothèques de fonctions
- f. Paramétrage par défaut
- g. Mise en place de solution de contournement
- h. Mise en place de solution de robustesse
- i. Intégration du Framework
- j. Pilotage de l'exécution des tests



