



Automatisation des applications web avec Selenium: Approfondissement

Objectifs

- Mettre en place des tests automatisés pour des applications Web
- Construire un framework pour faciliter l'automatisation des tests avec Selenium





Agenda

- **Introduction à Selenium: IDE**
- **Introduction à Selenium WebDriver**
- **PageObject Patterns**
- **Selenium Grid**
- **Selenium avancé**
- **Ajax & HTML**
- **Mobile Testing**
- **Framework d'automatisation**

Chap. 1: Introduction à Selenium

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

1.1 Automatisation des tests

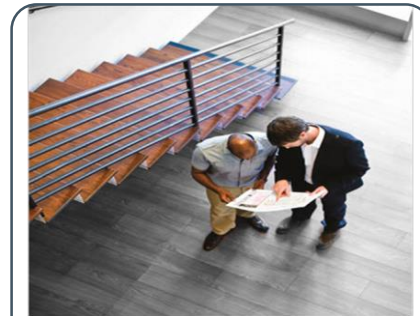
- Le Test fonctionnel permet de vérifier les caractéristiques qualité suivantes:



Précision



Interopérabilité

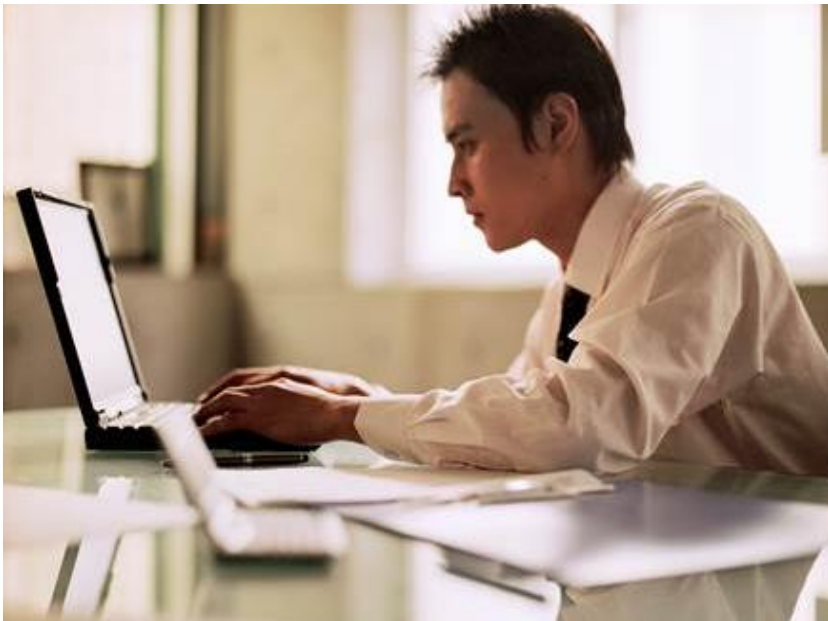


Prédictabilité



1.1 Automatisation des tests

Le Test fonctionnel peut être:



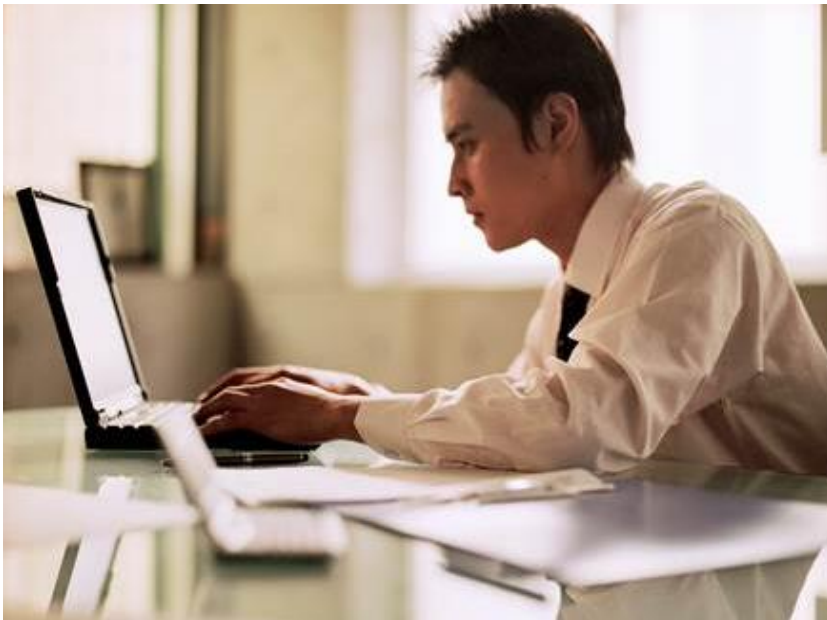
Manuel



Automatique

1.1 Automatisation des tests

Inconvénients du test manuel



Chronophage
Fastidieux/pénible
Contraignant
Coûteux
Incertain

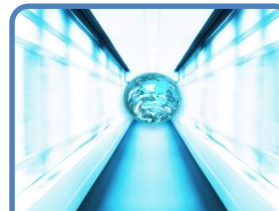
Manuel

1.1 Automatisation des tests

Avantage du test automatisé



Automatique



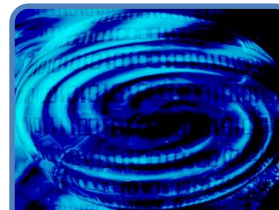
Rapide



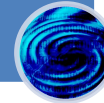
Fiable



Répétable



Compréhensible



Programmable



Réutilisable



1.1 Automatisation des tests

Les meilleurs candidats à l'automatisation

Tests de régression
des parcours-clés

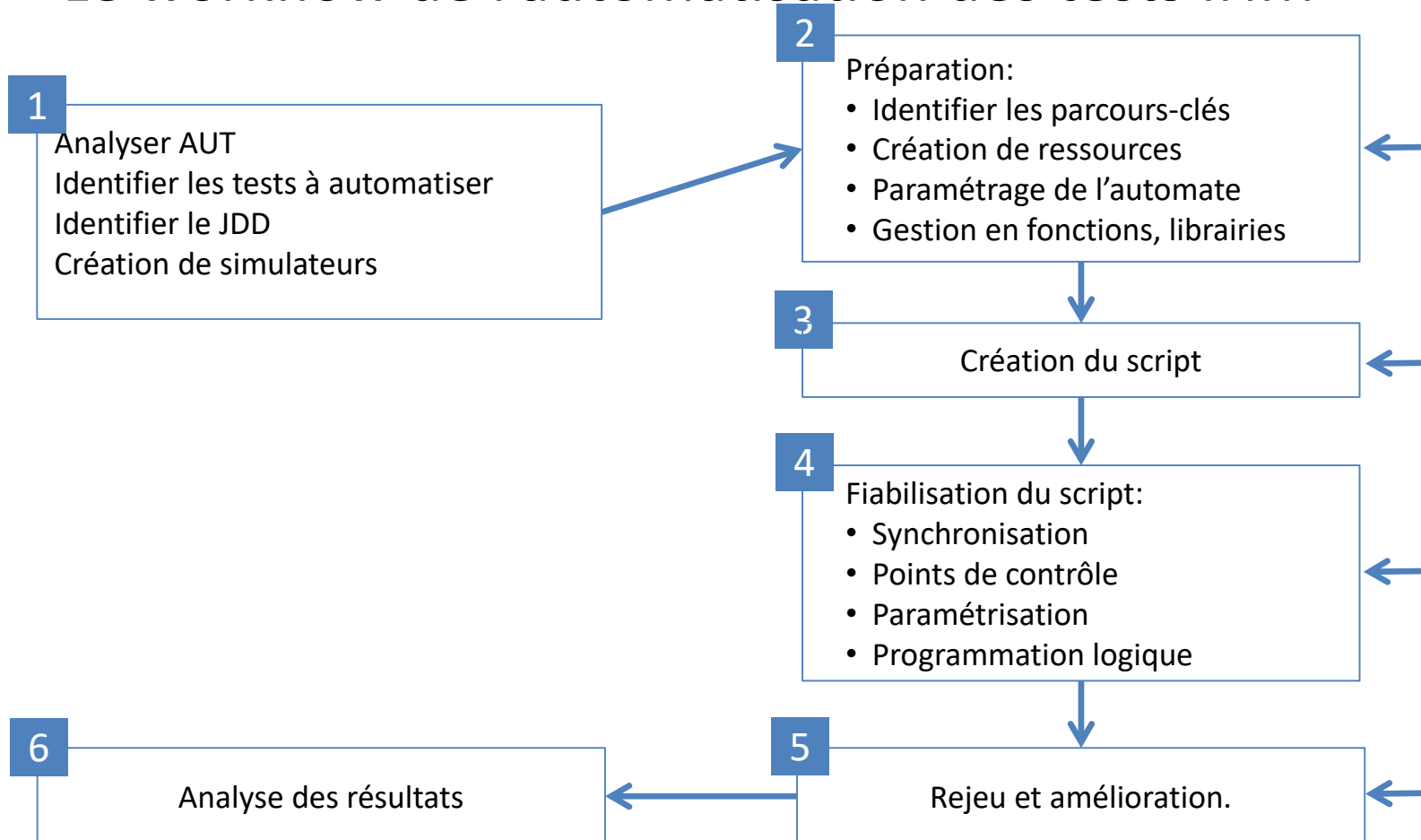
Smoke tests:
tests de confiance

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

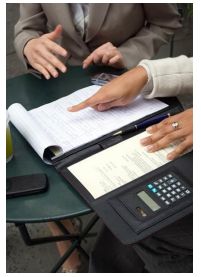
Analyse

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.

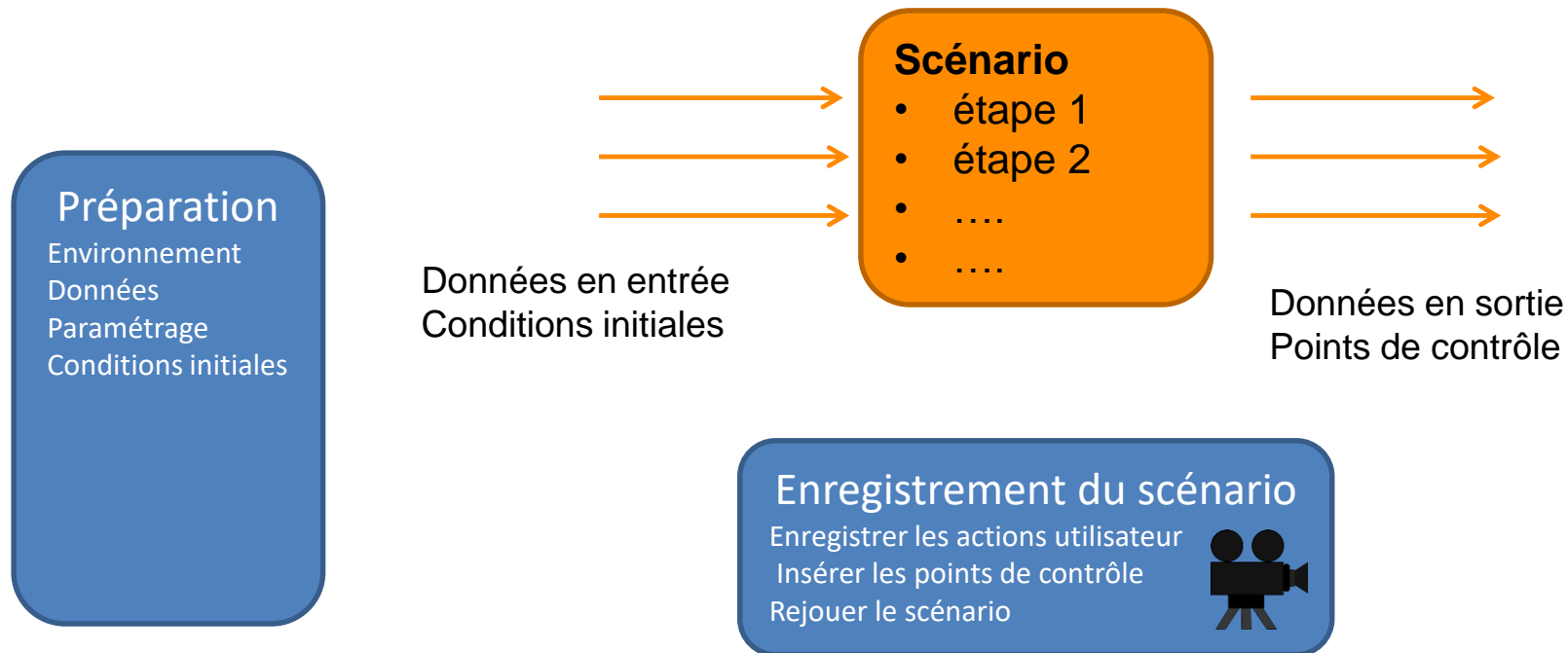
Une bonne connaissance et analyse des processus métiers permet d'optimiser l'effort de spécification des tests.

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 gratuit venue concurrencer les leaders commerciaux du marché

1.2 Framework Selenium

Plusieurs outils

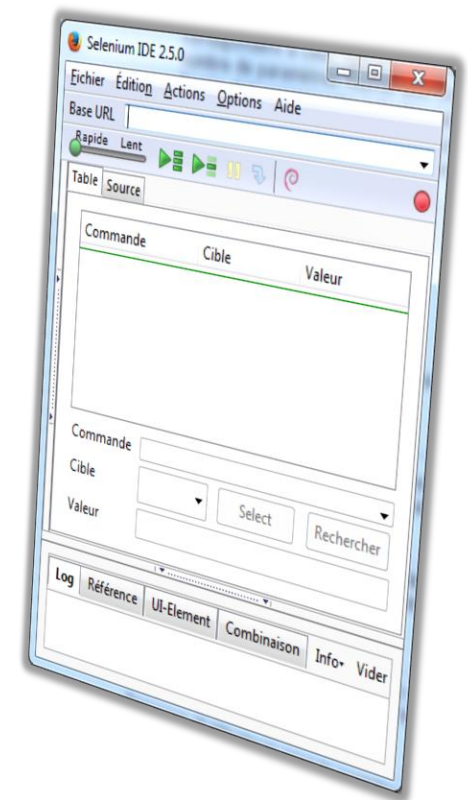
- **Selenium IDE**
- **Selenium WebDriver**
- **Selenium Grid**



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

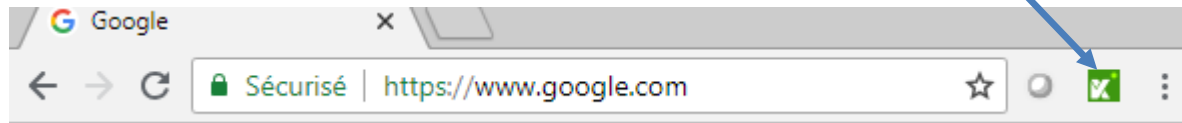
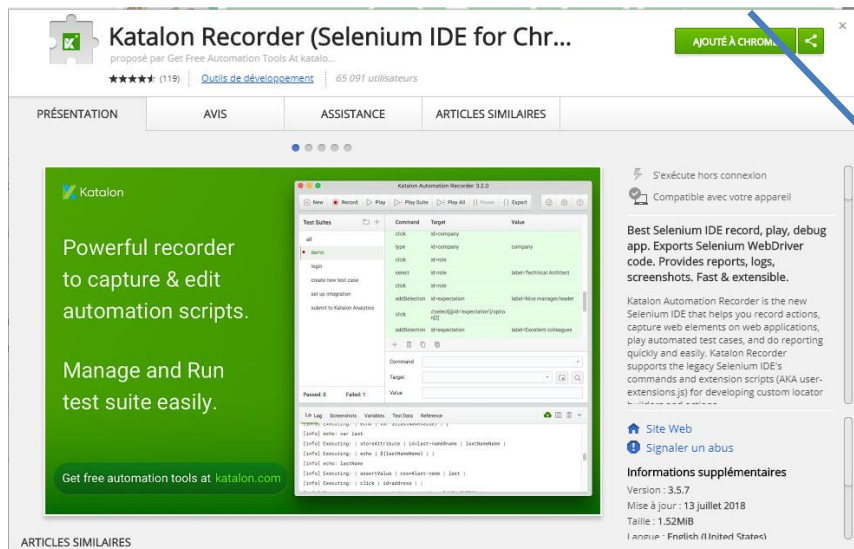
- Selenium IDE (Integrated Development Environment): outil à utiliser pour développer des cas de test Selenium.
- Plugin ou extension de navigateur
- Permet de faire de l'automatisation sans scripting
- Plusieurs solutions Selenium IDE:

The screenshot shows the 'Extensions' page of the Chrome Web Store. It lists three extensions related to Selenium IDE:

- Katalon Recorder (Selenium IDE for Chrome)**: A green extension with a 'NOUVEAU' (New) badge. It is described as a tool for recording, playing, and debugging Selenium WebDriver code. It has a 5-star rating (119 reviews) and a green 'ÉVALUER' (Review) button.
- Selenium IDE**: A blue extension with a 'NOUVEAU' (New) badge. It is described as a record and playback tool for Selenium WebDriver. It has a 5-star rating (78 reviews) and a green 'ÉVALUER' (Review) button.
- SideeX - Smart Record-Play Browser Automation**: A white extension with a parrot logo. It is described as a free and open source smart record-playback automated web application testing tool. It has a 5-star rating (24 reviews) and a blue '+ AJOUTER À CHROME' (Add to Chrome) button.

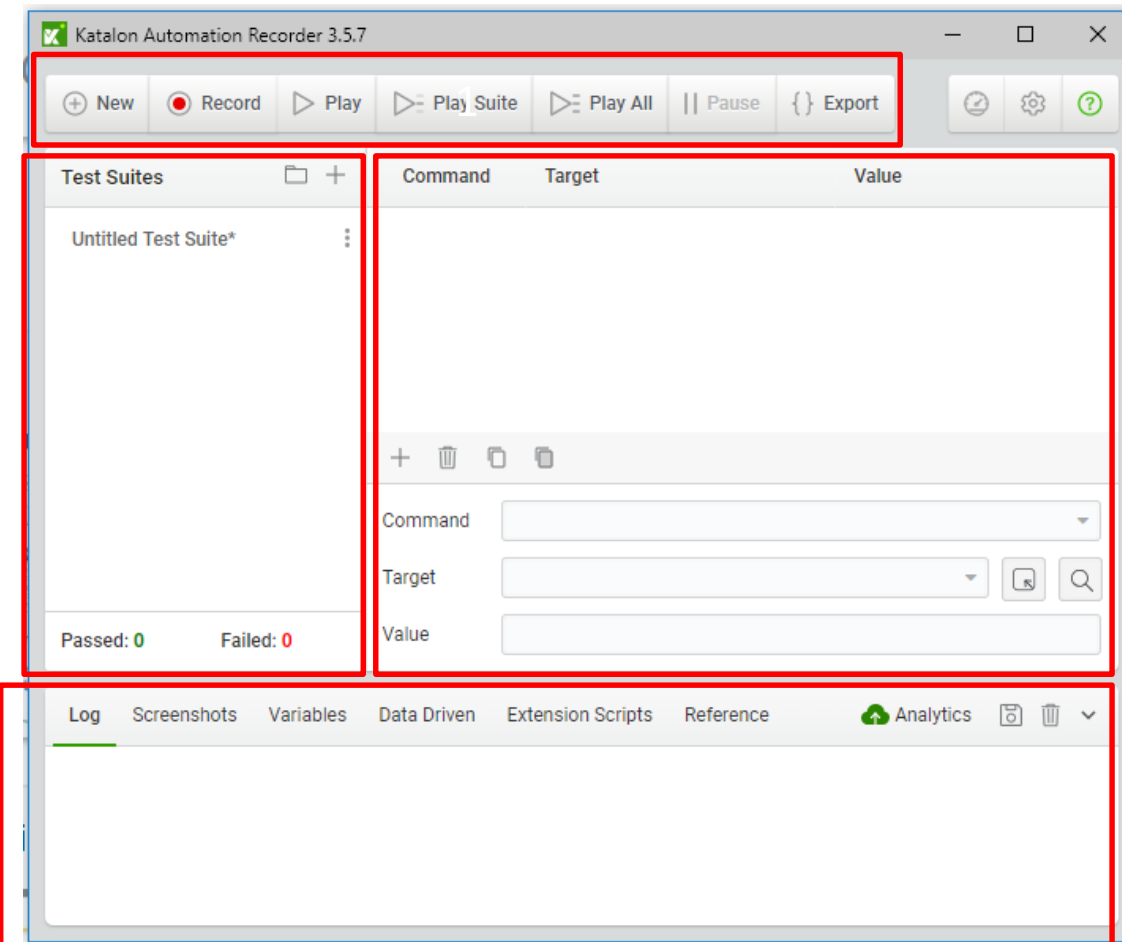
1.2 Katalon Recorder

Installation à partir de Chrome Extension



1.2 Katalon Recorder

Interface intuitif

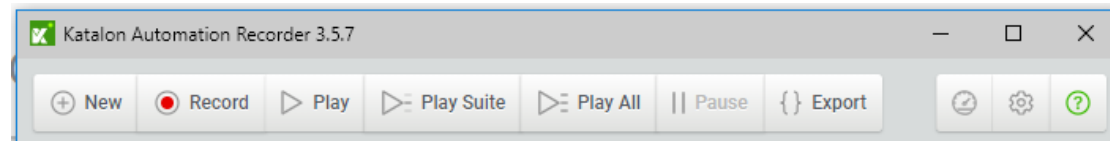


Main Toolbar
Test Case/ Suite Explorer
Test Case Details View
Log/Reference/Variable

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.

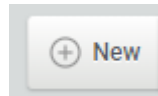
1.2 Katalon Recorder: Toolbar



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

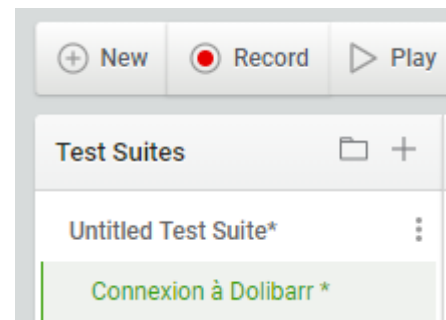
1.2 Katalon Recorder: Test

- Créer un test



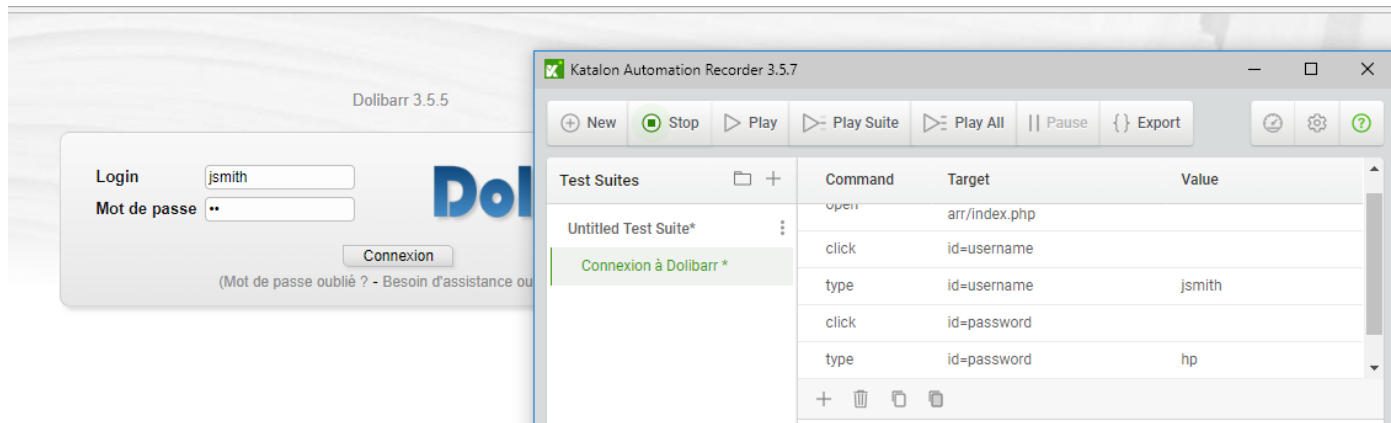
Katalon Recorder (Selenium IDE for Chrome)

Please enter the Test Case's name



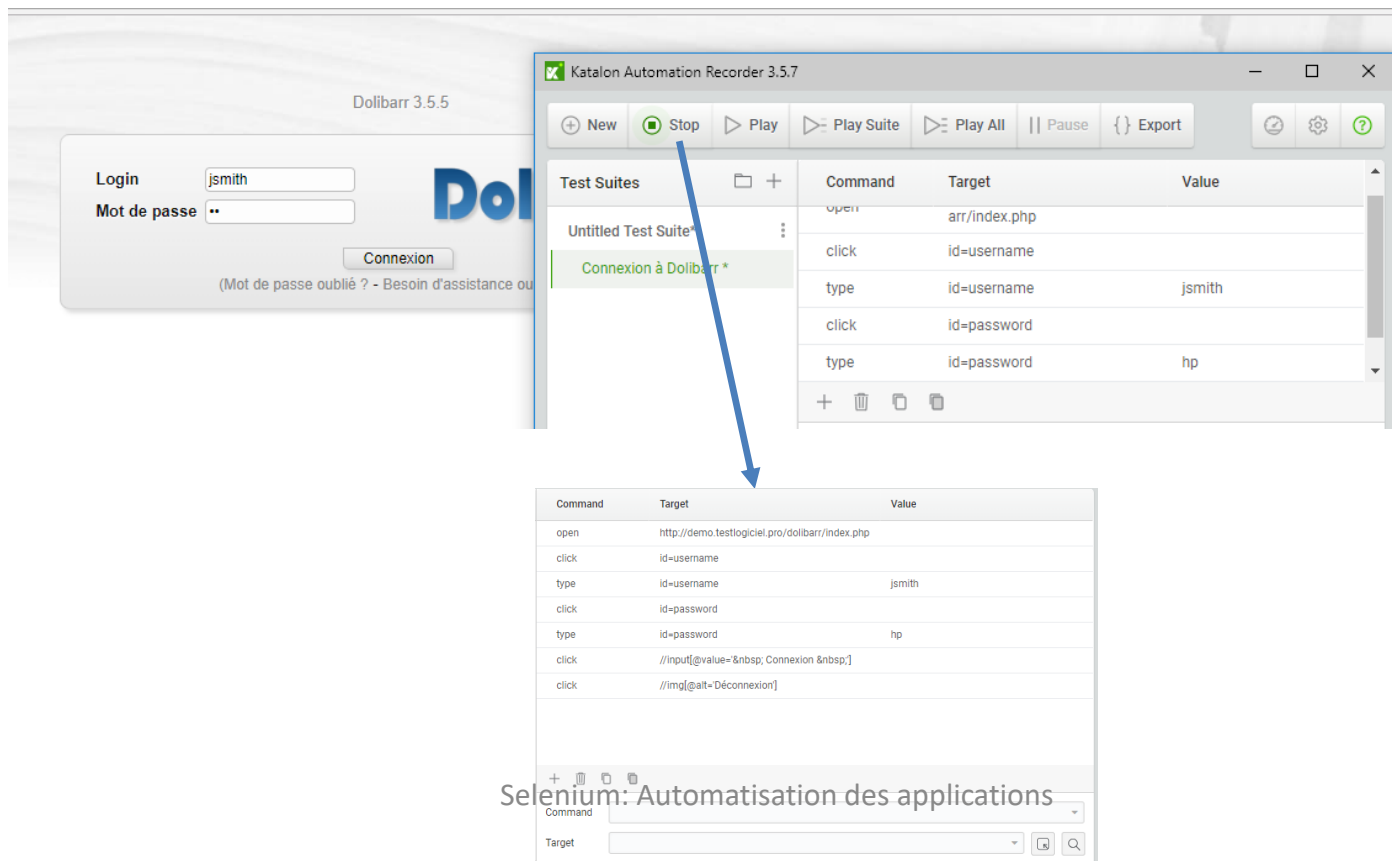
1.2 Katalon Recorder: Test

- Enregistrer un scénario



1.2 Katalon Recorder: Test

- Stopper l'enregistrement



The image shows the Katalon Recorder interface (version 3.5.7) overlaid on a Dolibarr 3.5.5 login page. The recorder window displays a test suite named 'Connexion à Dolibarr *' with a list of commands. A blue arrow points from the 'Stop' button in the recorder's toolbar to the 'Connexion à Dolibarr *' test suite entry in the list.

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']	

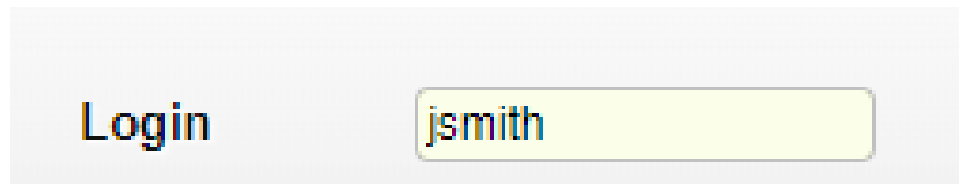
Selenium: Automatisation des applications

1.2 Katalon Recorder: Test

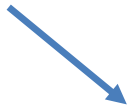
- Script Selenium

- 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



ERP/CRM

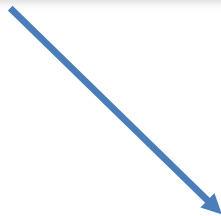
Dolibarr

Login

Mot de passe

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

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: 1Failed: 0

TargetValue

LogScreenshotsVariablesData DrivenExtension ScriptsReference

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 | |

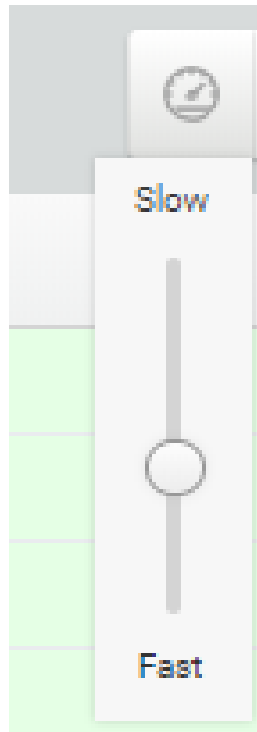
[info] Executing: | click | //img[@alt='Déconnexion'] | |

[info] Time: Sat Jul 14 2018 13:35:13 GMT+0200 (heure d'été d'Europe centrale) Timestamp: 1531568113543

[info] Test case passed

1.2 Katalon Recorder: Exécution

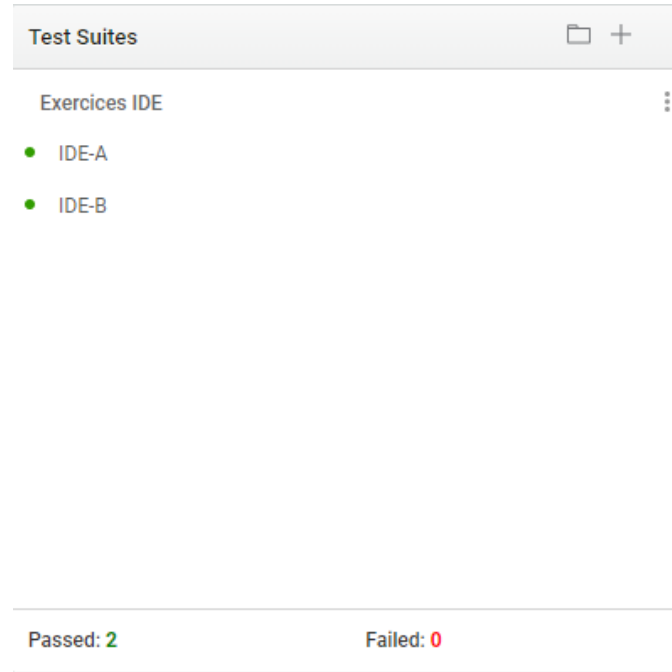
- Vitesse



1.2 Katalon Recorder: Exécution

- Résultats

Statut
d'exécution

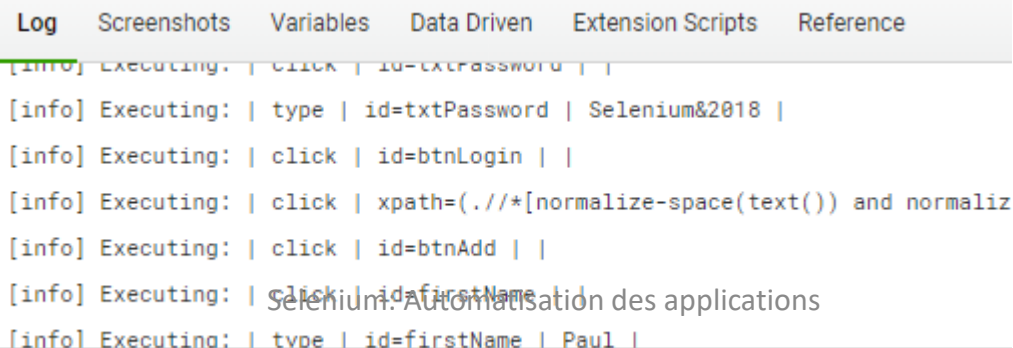


The screenshot shows the 'Test Suites' section of the Katalon Recorder interface. It lists 'Exercices IDE' with two sub-items, 'IDE-A' and 'IDE-B', both marked with green dots indicating successful execution. At the bottom, a summary bar shows 'Passed: 2' in green and 'Failed: 0' in red.

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

Passed: 2 Failed: 0

Log du
rapport
d'exécution

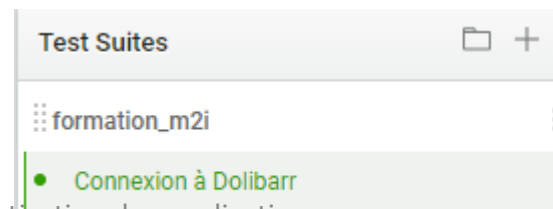
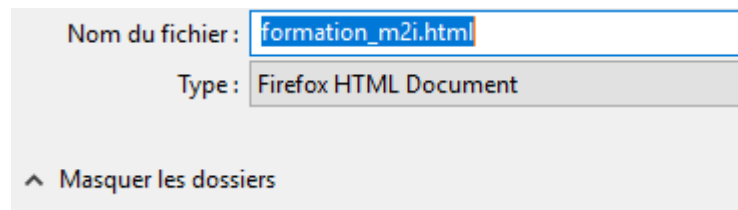
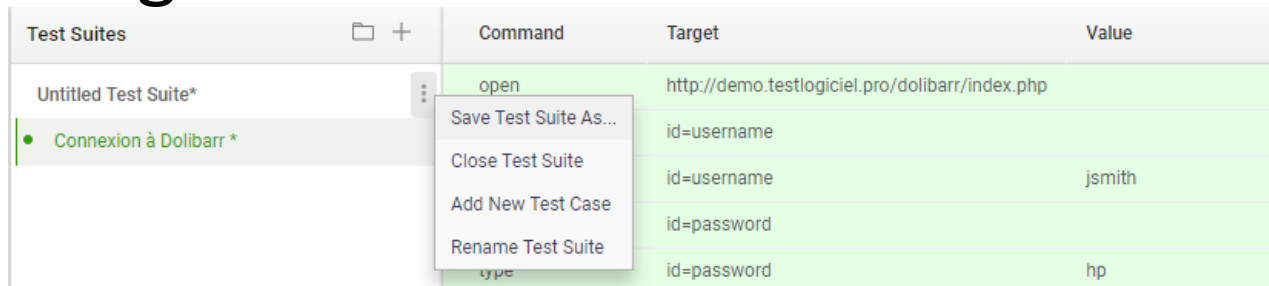


The screenshot shows the 'Log' tab of the Katalon Recorder interface. It displays a series of log entries for an execution, including commands like 'click', 'type', and 'click' with their respective parameters. The text 'Selenium: Automatisation des applications' is visible in the background.

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					

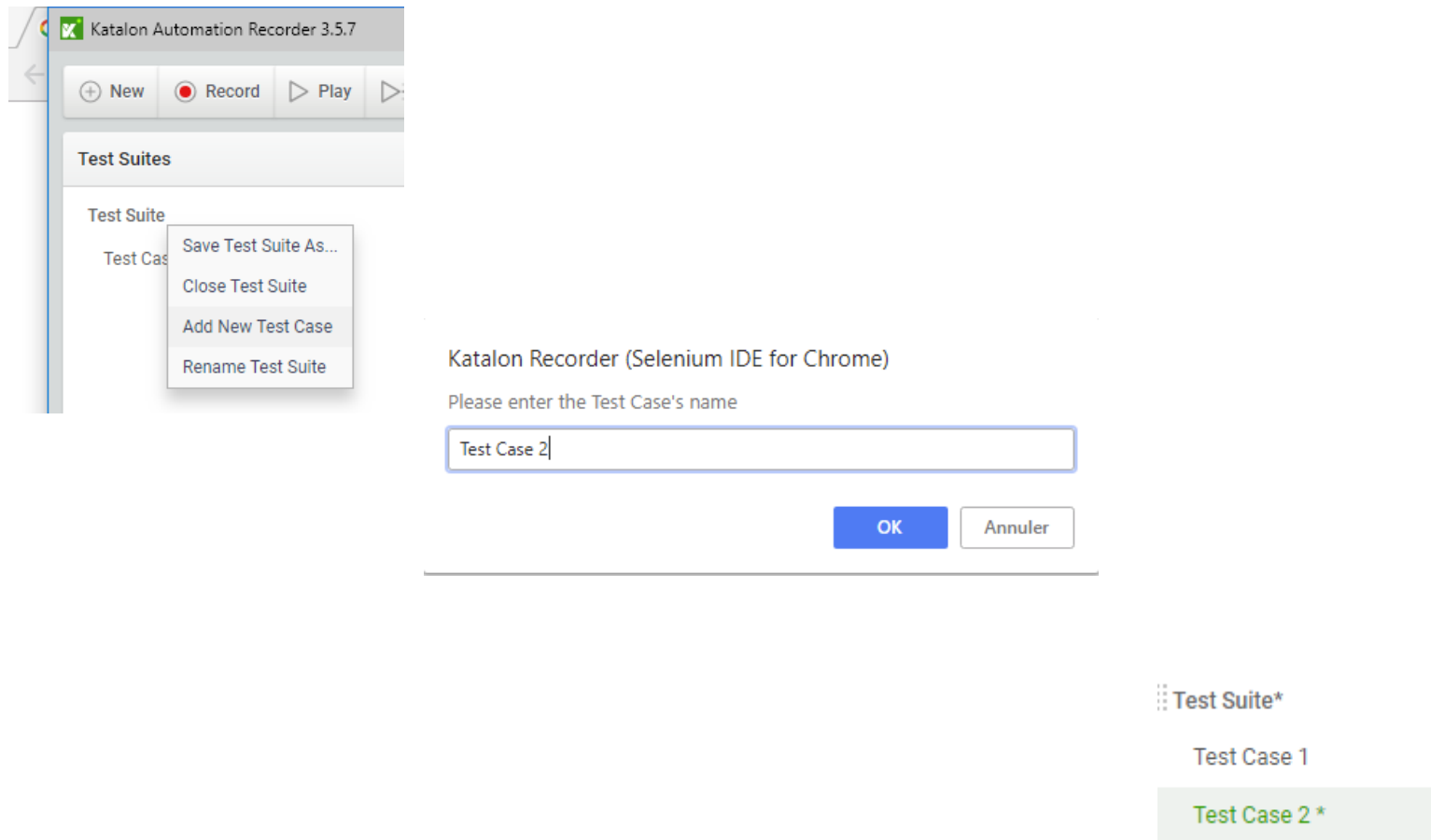
1.2 Katalon Recorder: Suite

- Sauvegarder



1.2 Katalon Recorder: Suite

Ajouter un cas de test dans la suite de test



1.2 Katalon Recorder: TP

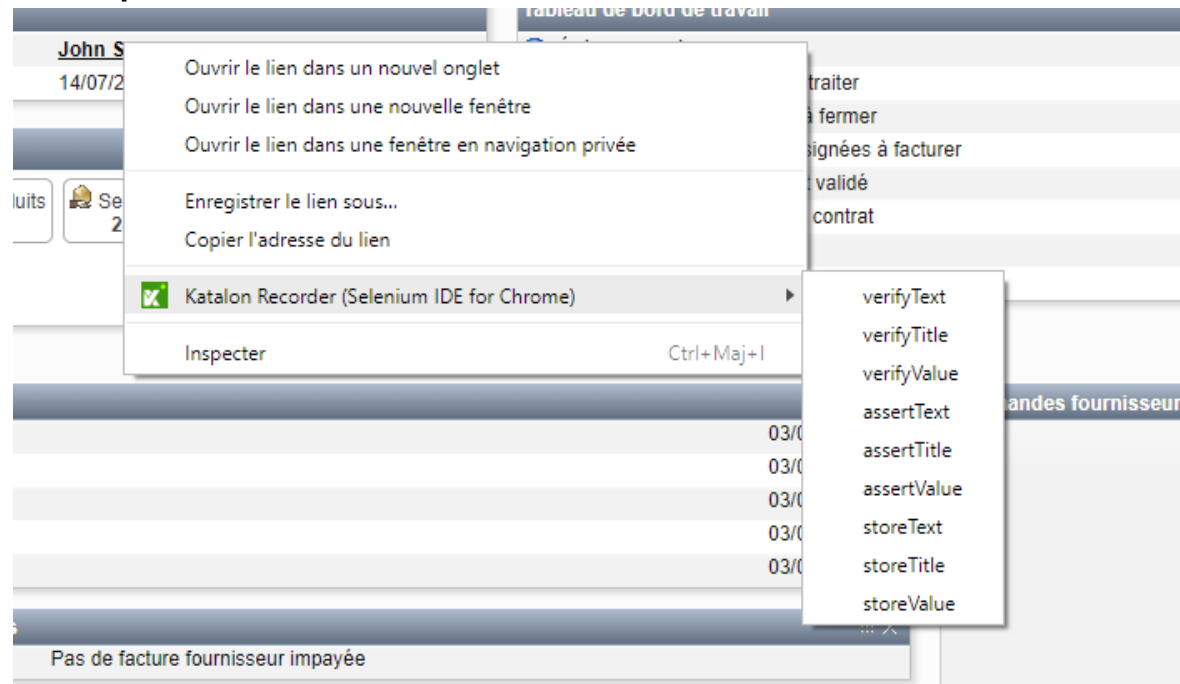
Exercice IDE-A: Enregistrer un Test case

- 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

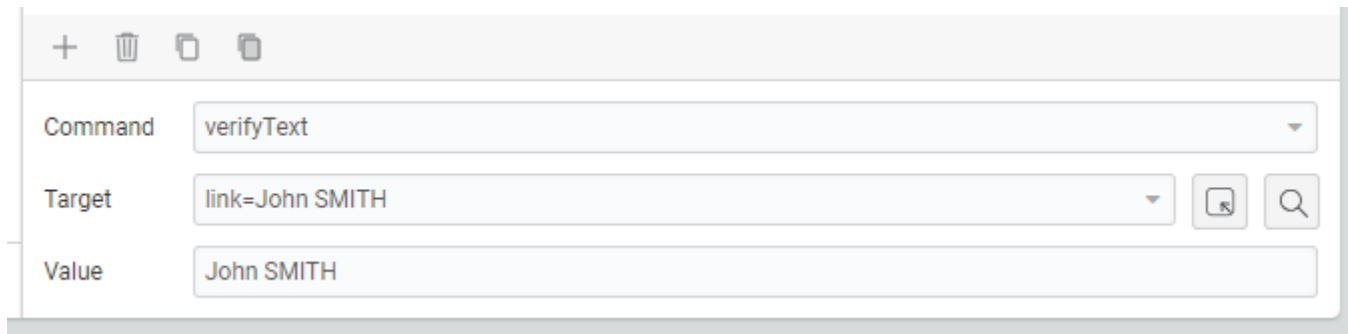


Quelles sont les vérifications possibles pour une application Web?

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 displays the Katalon Recorder's checkpoint configuration window. At the top, there is a toolbar with icons for adding (+), deleting (trash), copying, and pasting. Below this, the configuration is organized into three rows: '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 mouse cursor and a magnifying glass), and 'Value' with a text input field 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

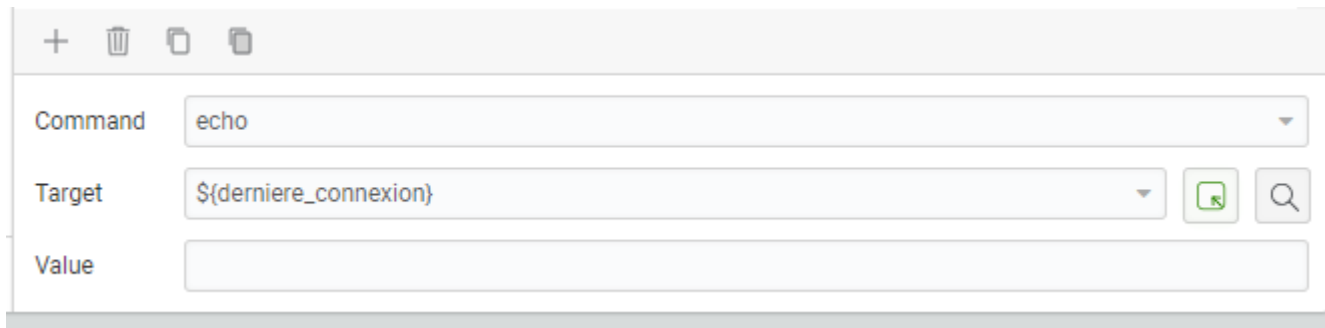
Informations	
Utilisateur	John SMITH
Connexion précédente	15/07/2018 06:02

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

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

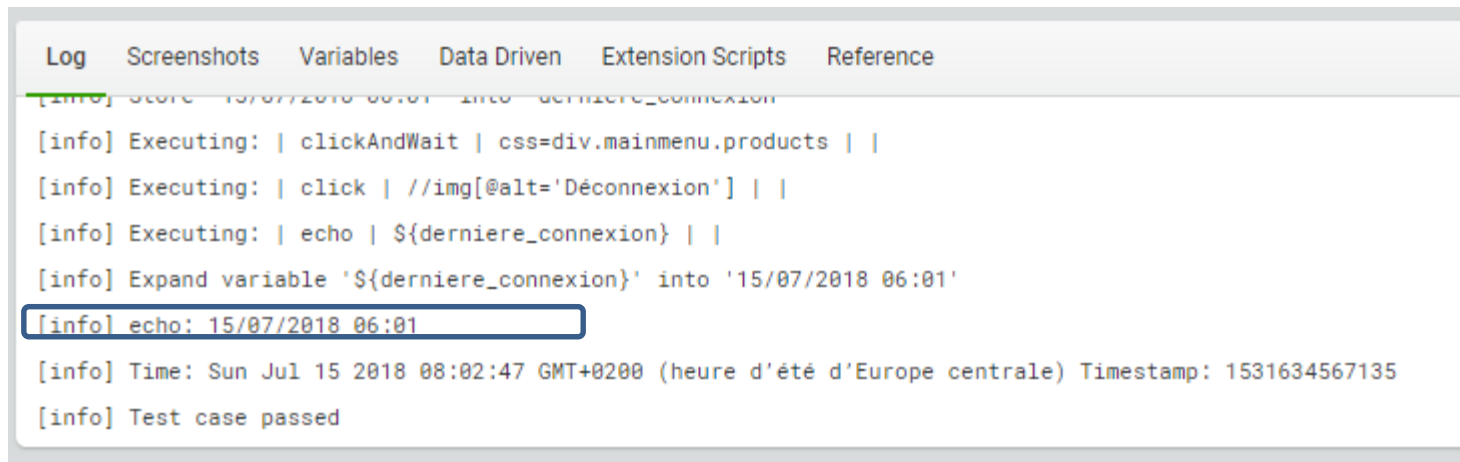
1.2 Katalon Recorder: Variable

Utiliser la variable



The screenshot shows the Katalon Recorder interface for configuring a command. It has a toolbar with icons for adding, deleting, and duplicating steps. The 'Command' dropdown is set to 'echo'. The 'Target' field contains the variable '\${derniere_connexion}' and includes a search icon. The 'Value' field is currently empty.

Command	echo
Target	<input type="text" value="\${derniere_connexion}"/>
Value	<input type="text"/>



The screenshot shows the 'Log' tab of the Katalon Recorder. It displays a series of log messages. The message '[info] echo: 15/07/2018 06:01' is highlighted with a blue box, indicating the output of the echo command. Other log messages show the execution of various steps like 'clickAndWait', 'click', and 'Expand variable'.

```
[info] Store 15/07/2018 06:01 Info 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

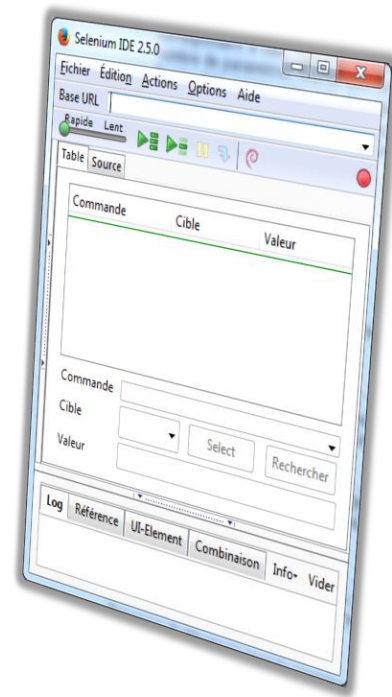
1.2 Katalon Recorder: TP

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 a "Code client" field. A blue box highlights the "Nom du tiers" input field, and a blue arrow points from it to a callout box containing the following HTML code:

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

The form fields include:

- Type du société: ☒ Créer tiers ☐ Créer un tiers + un contact/adresse fils
- Nom du tiers: [input field]
- Prénom: [input field]
- Titre civilité: [dropdown menu]
- Nom alternatif (commercial, marque, ...): [input field]
- Prospect / Client: [dropdown menu]
- Fournisseur: [dropdown menu]
- État: [dropdown menu, currently set to "Ouvert"]
- Code client: CU1904-1842

1.3 Gestion des objets

Le Target: identification des éléments

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

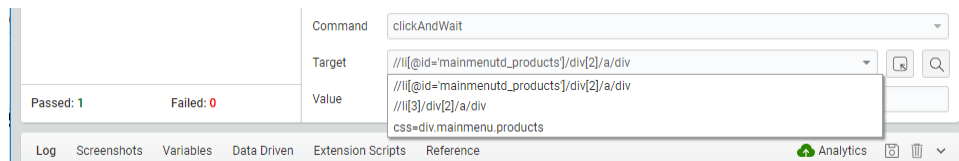


Target

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



Identification doit être fiable



clickAndWait

css=div.mainmenu.products

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']`

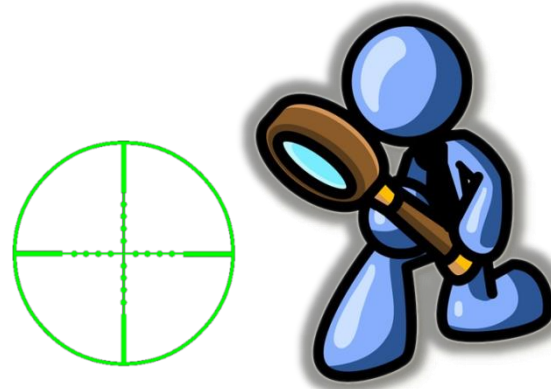
- `//table[.//th='Réf.']/tr[4]/td[2]`

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.

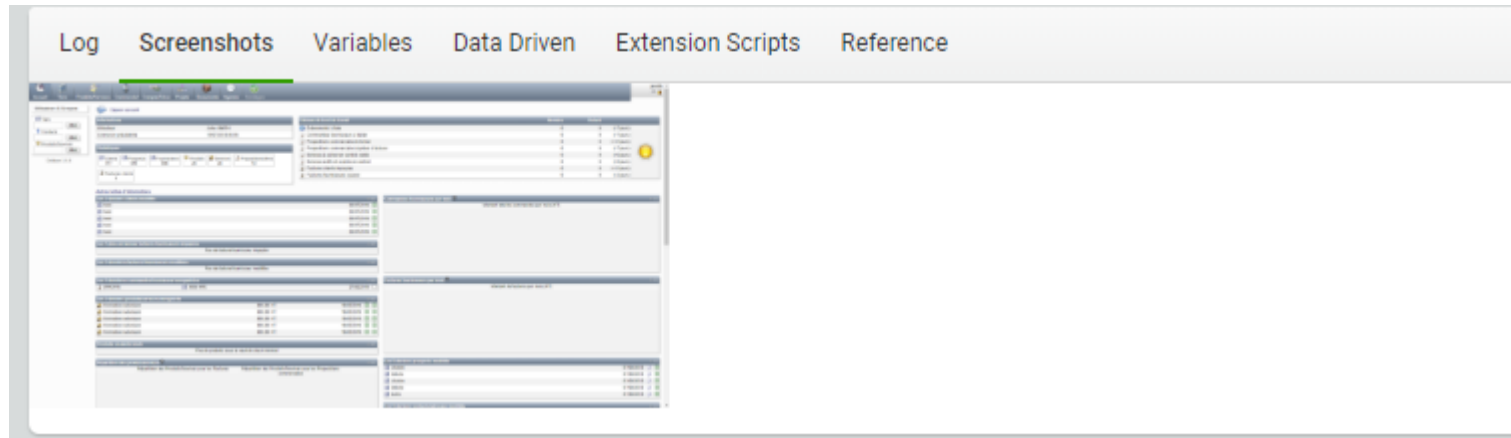
1.3 Gestion des objets

Prendre une capture d'écran

Command

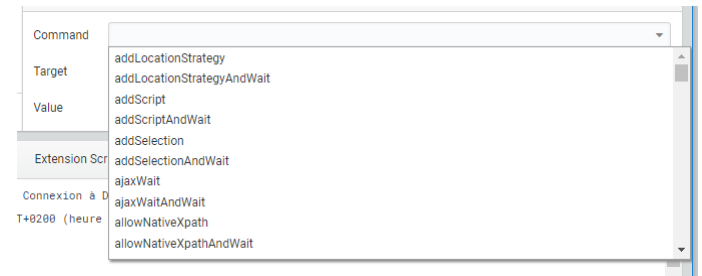
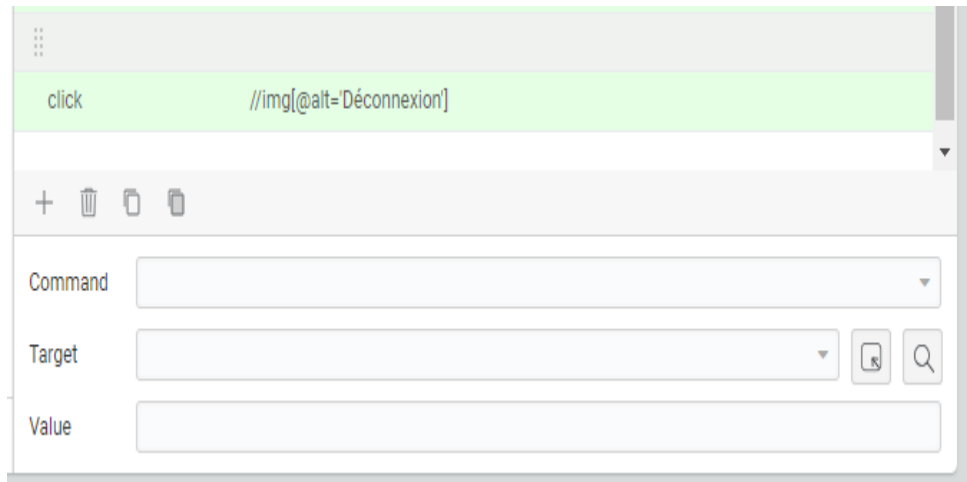
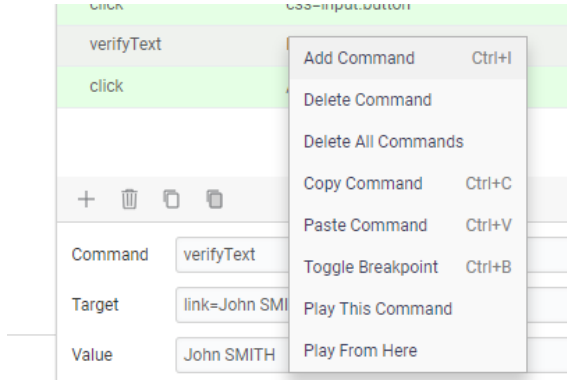
Target  

Value



1.3 Gestion des objets

Insérer une commande sans enregistrement



1.3 Gestion des objets

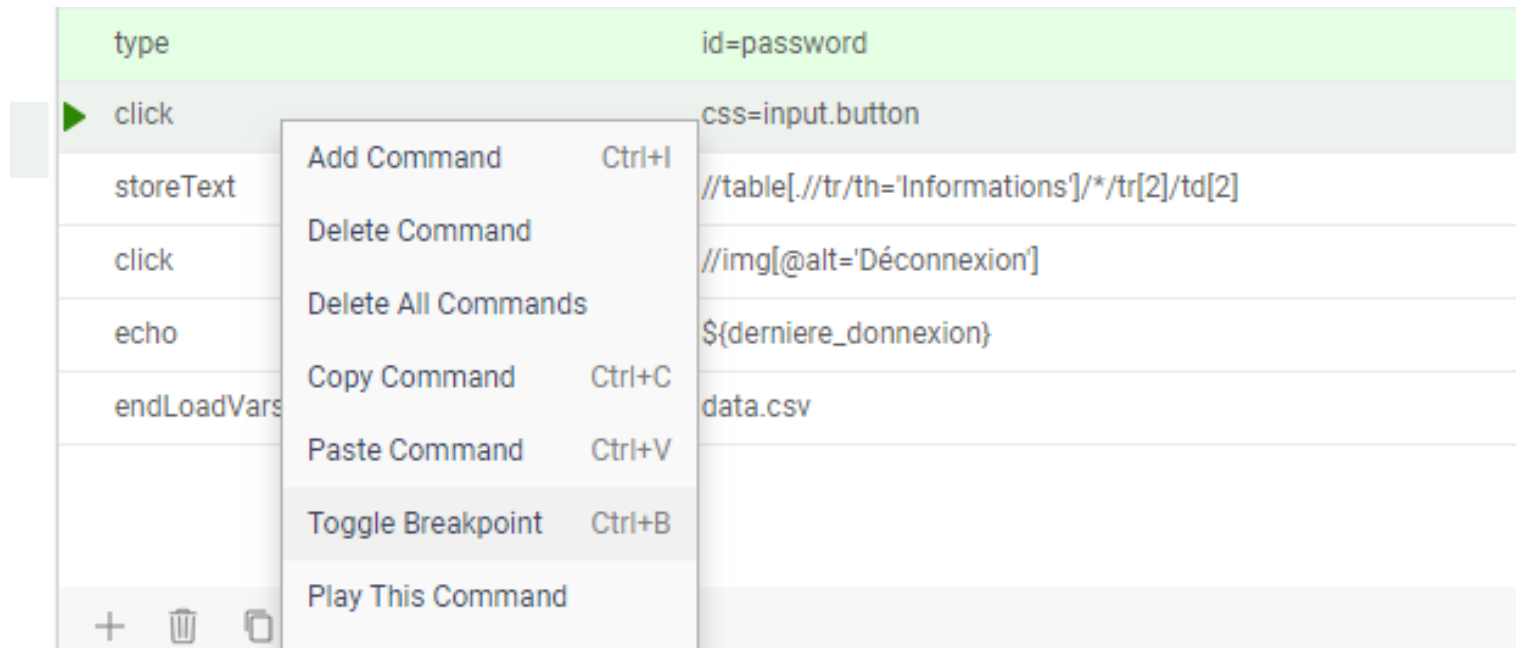
Les commandes selenese



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
-

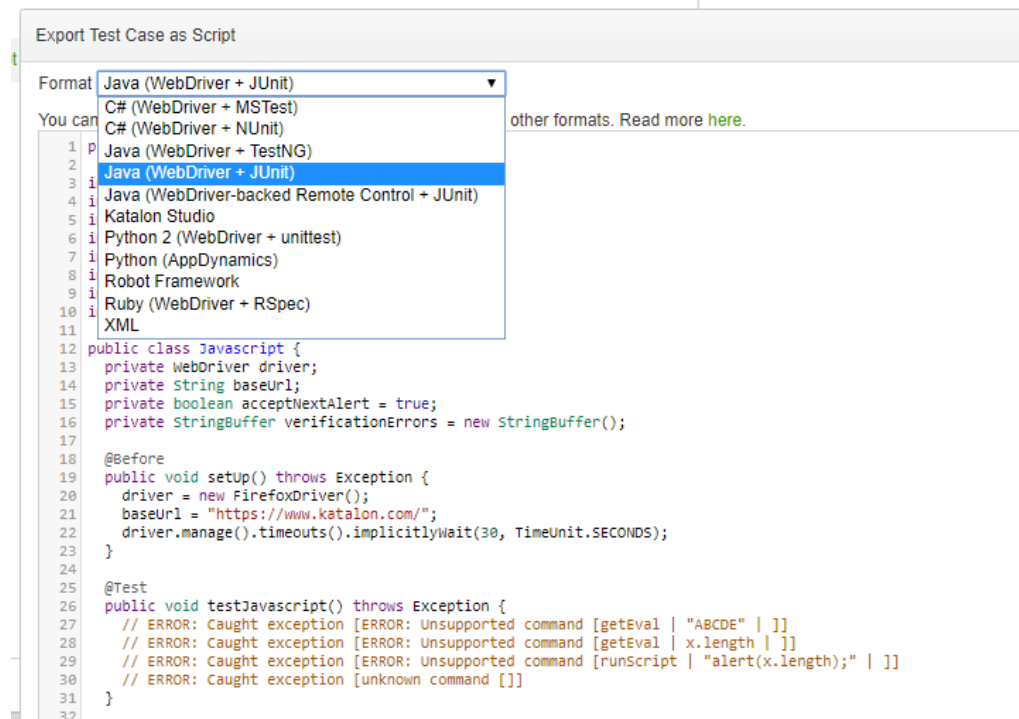
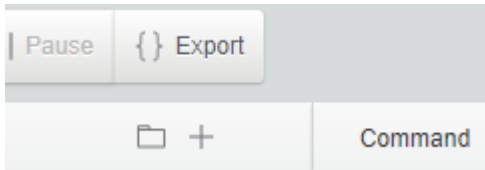
1.3 Gestion des objets

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

1.3 Gestion des objets

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



1.3 Gestion des objets

Mettre en pratique sur un scénario de test de votre application



Chap. 2: Selenium WebDriver

- Selenium WebDriver et Java
- Commandes
- Assertions

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

```
pom.xml
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.

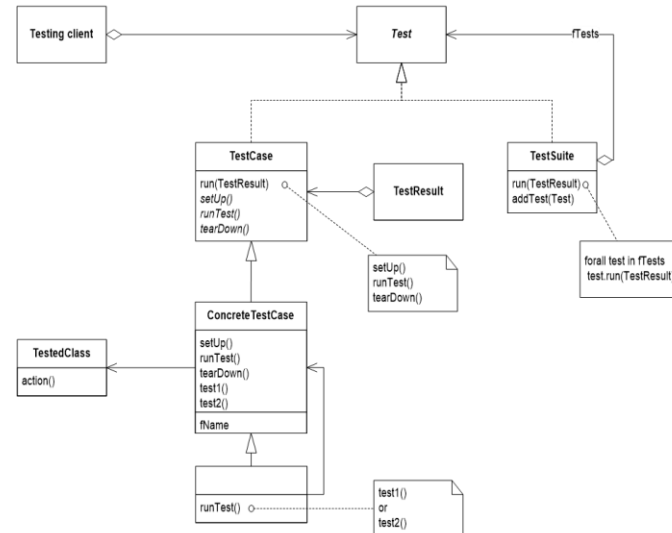
Pourquoi?

- 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

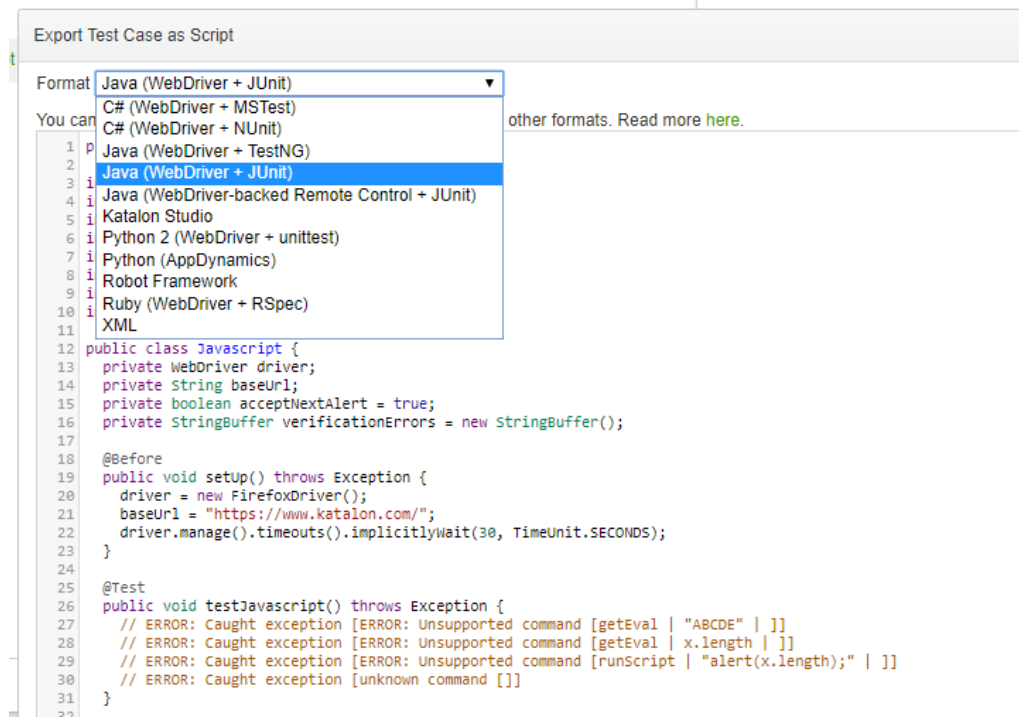
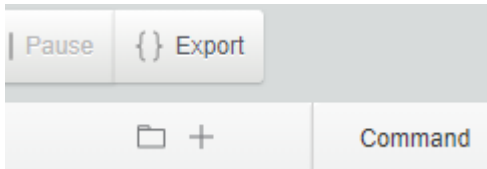
Architecture

- 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://demo.testlogiciel.pro/orangehrm>
- Attendre 5 secondes

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

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

2.2 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
```

2.2 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

2.2 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']"));
```

2.2 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();
```

2.2 WebDriver: Commandes

Objet Select

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

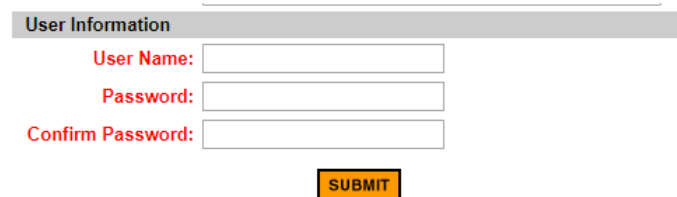


The image shows a web form with a label 'Pays' and a dropdown menu. The dropdown menu is open, showing 'France (FR)' as the selected option. The dropdown menu has a blue border and a small downward arrow on the right side.

2.2 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". It contains three input fields labeled "User Name:", "Password:", and "Confirm Password:". Below the fields is a yellow "SUBMIT" button.

User Information	
User Name:	<input type="text"/>
Password:	<input type="password"/>
Confirm Password:	<input type="password"/>
<input type="button" value="SUBMIT"/>	

2.2 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 **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**

2.3 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());
```

2.3 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"));
```

2.3 WebDriver: Assertions

Exercice C1

- Reprendre la classe OrangeSuite
- Créer une méthode valeursAverifier1
- Lancer chrome en maximized
 - Aller à l'adresse: <http://orangehrm.selenium-formation.org>
 - Se connecter avec les identifiants:
 - User name: admin
 - Password: Selenium&2018
 - Vérifier que le titre de la page est: « OrangeHRM »
 - Vérifier que le lien Admin est affiché
 - Vérifier que le texte Dashboard est affiché
 - Cliquer sur Admin
 - Vérifier que le texte Utilisateur du système est affiché
 - Vérifier que le bouton Ajouter est présent
 - Vérifier que le champ Nom d'utilisateur est affiché et modifiable
 - Cliquer sur Déconnexion
- Modifier le script: rajouter après la connexion que le champ de login « Nom d'utilisateur » n'est pas présent
 - Que se passe t'il?

2.3 WebDriver: Assertions

Exercice C2

- Reprendre la classe OrangeSuite
- Reprendre le scénario B2
 - 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**

2.3 WebDriver: Assertions

Vérifier qu'un élément n'est pas présent

```
public boolean verifyElementPresent(WebDriver driver, By locator){  
  
    try{  
        driver.findElement(locator);           //On recherche l'élément  
        return true;                           //On retourne vrai si l'élément est trouvé  
    }catch(Exception NoSuchElementException){  //On capture l'exception renvoyée si l'élément n'est pas trouvé  
        return false;  
    }  
}
```

2.3 WebDriver: Table

La gestion des tables est importante car très présente en html

Informations	
Utilisateur	John SMITH
Connexion précédente	19/07/2018 19:27

```
WebElement tableInfos =driver.findElement(By.xpath("//table[.//th='Informations']"));
```

```
String nomUser = tableInfos.findElements(By.tagName("tr")).get(1).findElements(By.tagName("td")).get(1).getText();
```

Retourne un tableau
contenant toutes les lignes
(avec le tag <tr>) de l'objet
Tableau

Récupération
de la ligne 2
Attention,
l'index
commence à
0

Retourne un tableau
contenant toutes les colonnes
de la ligne (avec le tag <td>)

2.3 WebDriver: TakeScreenshot

Prendre une capture d'écran

```
File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);  
try {  
    FileUtils.copyFile(src, new File("C:\\tmp\\error.png"));  
}catch (IOException e){}
```


2.3 WebDriver: Assertion

Exercice D1

Reprener 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

2.3 WebDriver: Assertion

Exercice D2

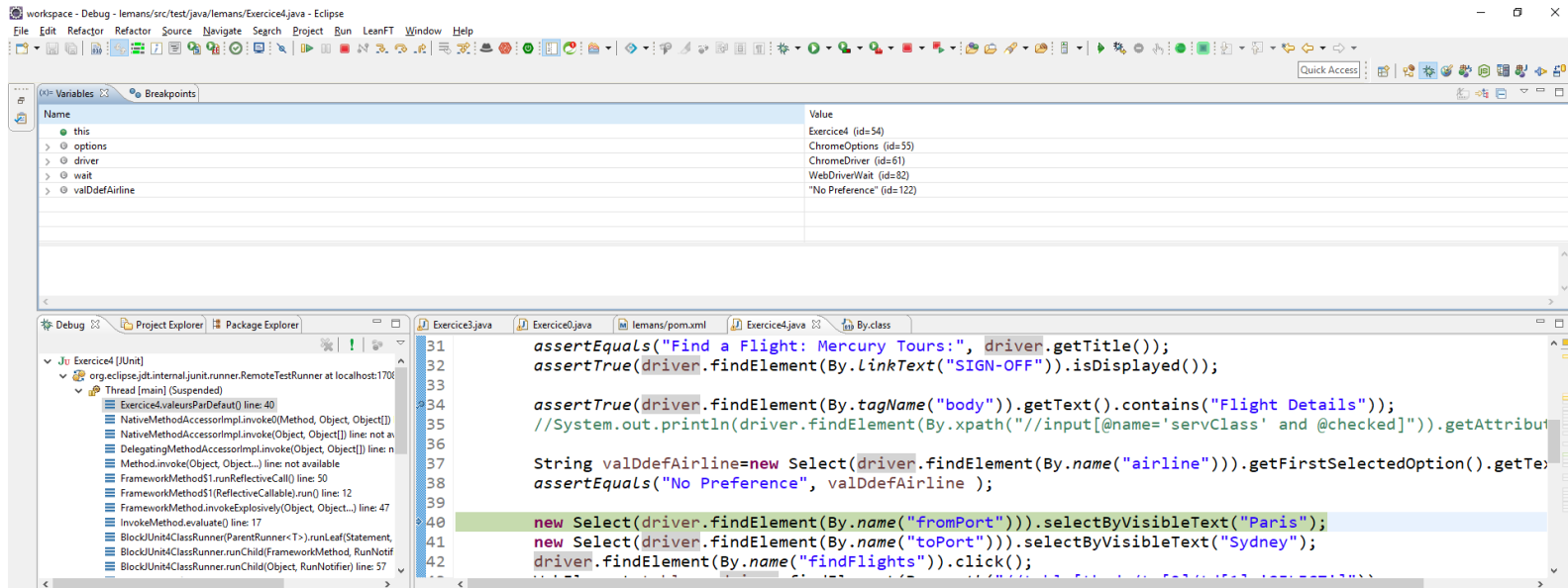
- Créer une nouvelle classe DolibarrSuite

Scénario: Créer une note de frais

- Lancer Chrome: <http://dolibarr.selenium-formation.org/>
- Saisir Login=jsmith
- Saisir Mot de passe=Selenium&2018
- 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

2.3 WebDriver: Débogage

Fonctionnalités de débogage



2.3 WebDriver: Débogage

Exercice D3: Utiliser le débogage

- Créer un scénario suivant, exécuter le en mode Debug avec du pas à pas
 - Lancer Chrome: <http://newtours.demoaut.com/>
 - Cliquer sur le lien SIGN-ON
 - Saisir Username=tonton
 - Saisir Password=tonton
 - Cliquer sur SUBMIT
 - Vérifier que le lien PROFILE est présent
 - Vérifier que le lien REGISTER n'est plus affiché
 - Vérifier qu'on est sur la page FLIGHT FINDER
 - Cliquer sur PROFILE
 - Vérifier que First Name=Tonton
 - Vérifier que Country=France
 - Cliquer sur le lien SIGN-OFF

2.3 WebDriver: Ordre

JUnit: tri des méthodes

```
import org.junit.FixMethodOrder;
import org.junit.Test;
import org.junit.runners.MethodSorters;

@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class TestMethodOrder {

    @Test
    public void testA() {
        System.out.println("first");
    }
    @Test
    public void testB() {
        System.out.println("second");
    }
    @Test
    public void testC() {
        System.out.println("third");
    }
}
```

2.3 WebDriver: Ordre

JUnit: tri des classes avec Suite

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
    TestFeatureLogin.class,
    TestFeatureLogout.class,
    TestFeatureNavigate.class,
    TestFeatureUpdate.class
})

public class FeatureTestSuite {
    // the class remains empty,
    // used only as a holder for the above annotations
}
```

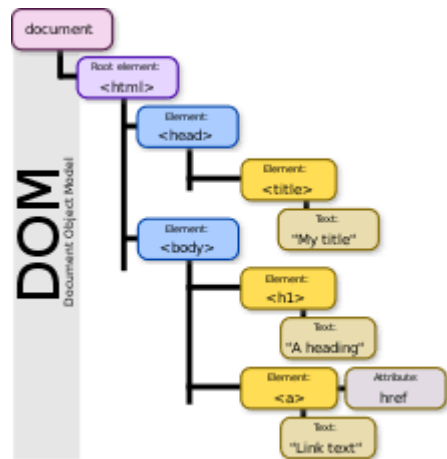
2. WebDriver: TP

Exercice E: TP

Application	
Dolibarr	Connexion avec commercial
Dolibarr	Créer un produit
Dolibarr	Créer un client avec plusieurs contacts
Dolibarr	Créer une proposition commerciale
Dolibarr	Déconnexion

Chap. 3: Gestion des objets

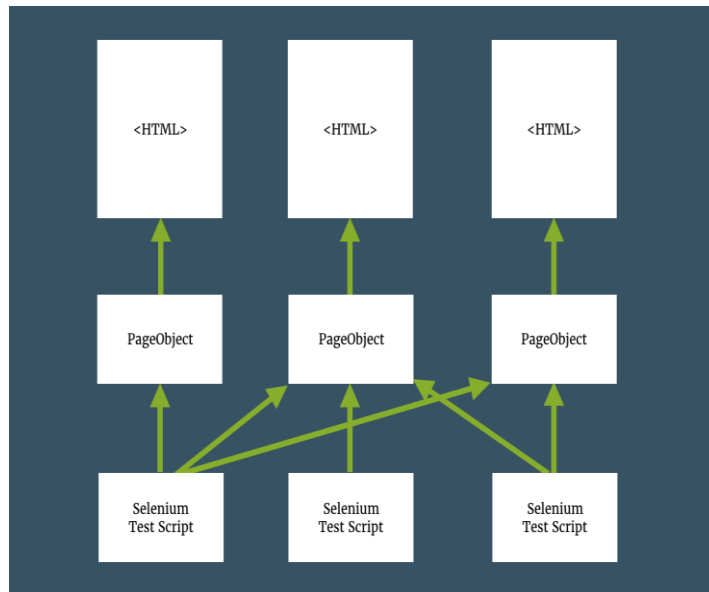
- PageObjects



3. WebDriver: Gestion des objets

Page Objects

- Pattern permettant de gérer les objets de l'application Web
- Meilleure lisibilité des scripts de test
- Maintainabilité améliorée des scripts de test



this API is about
the application

`selectAlbumWithTitle()
getArtist()
updateRating(5)`

Page Objects

Album
Page

Album List
Page

this API is
about HTML

`findElementsWithClass('album')
findElementsWithClass('title-field')
getText()
click()
findElementsWithClass('ratings-field')
setText(5)`

3. WebDriver: Gestion des objets

Page Objects

- Approche
 - Création d'une classe pour une Page Web: elle implémente les objets de la page ainsi que les opérations possibles sur ces objets

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class LoginPage {

    protected WebDriver driver; // Selenium WebDriver

    @FindBy(id="username") // Locator pour identifier le champ login
    private WebElement txtLogin;

    @FindBy(id="password") // Locator pour identifier le champ password
    private WebElement txtPassword;

    @FindBy(className="button") // Locator pour identifier le bouton connexion
    private WebElement btnConnexion;

    public LoginPage(WebDriver driver) {
        this.driver = driver; // Initialisation de la page avec le WebDriver
    }

    // Opération de saisie dans le champ Login
    public void taperLogin(String texte) {
        txtLogin.clear();
        txtLogin.sendKeys(texte);
    }

    // Opération de saisie dans le champ Password
    public void taperPassword(String texte) {
        txtPassword.clear();
        txtPassword.sendKeys(texte);
    }

    // Opération de clic sur le bouton Connexion
    public void valider(){
        btnConnexion.click();
    }
}
```

```
public class ConnexionPage {
    private WebDriver driver;
    private WebElement txtUsername;
    private WebElement txtPassword;
    private WebElement btnConnexion;

    public ConnexionPage(WebDriver driver){
        driver = this.driver;
        txtUsername=driver.findElement(By.id("username"));
        txtPassword=driver.findElement(By.id("password"));
        btnConnexion=driver.findElement(By.xpath("//input[contains(@value, 'Connexion')]"));
    }
}
```

3. WebDriver: Gestion des objets

Page Objects

- Approche
 - Utilisation

```
driver.get("http://demo.testlogiciel.pro/dolibarr");  
LoginPage loginPage = PageFactory.initElements(driver, LoginPage.class);  
loginPage.getTxtUsername().sendKeys(...charSequences: "jsmith");  
loginPage.getTxtPassword().sendKeys(...charSequences: "dolibarrhp");
```

3. WebDriver: Gestion des objets

LoadableComponent

- Approche
 - Permet de gérer le chargement d'une page en se basant sur des éléments au lieu de *document.readyState*
 - Evite les problèmes de synchronisation
 - Technique adaptée pour les nouveau framework AngularJS, Ajax

```
public class ConnexionPage extends LoadableComponent<ConnexionPage> {  
    private WebDriver driver;  
    private WebElement txtUsername;  
    private WebElement txtPassword;  
  
    protected void isLoading() throws Error {  
        System.out.println("Je suis chargé quand");  
        wait.until(ExpectedConditions.elementToBeClickable(btnConnexion));  
    }  
  
    public ConnexionPage get() {  
        try {  
            isLoading();  
            return this;  
        } catch (Error e) {  
            load();  
        }  
        isLoading();  
        return this;  
    }  
}
```

```
ConnexionPage loginPage = new ConnexionPage(driver).get();
```

Scénario: Automatisation des applications

3. WebDriver: Gestion des objets

Exercice G: Page Objet

- Copier la classe l'exercice D1
- Mettre en place PageObjects pour automatiser ce scénario

Chap. 4: Selenium WebDriver

- Les drivers
- Selenium Grid
- Assertions

4.1 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");
```

4.1 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);
```


4.1 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**

4.1 Headless

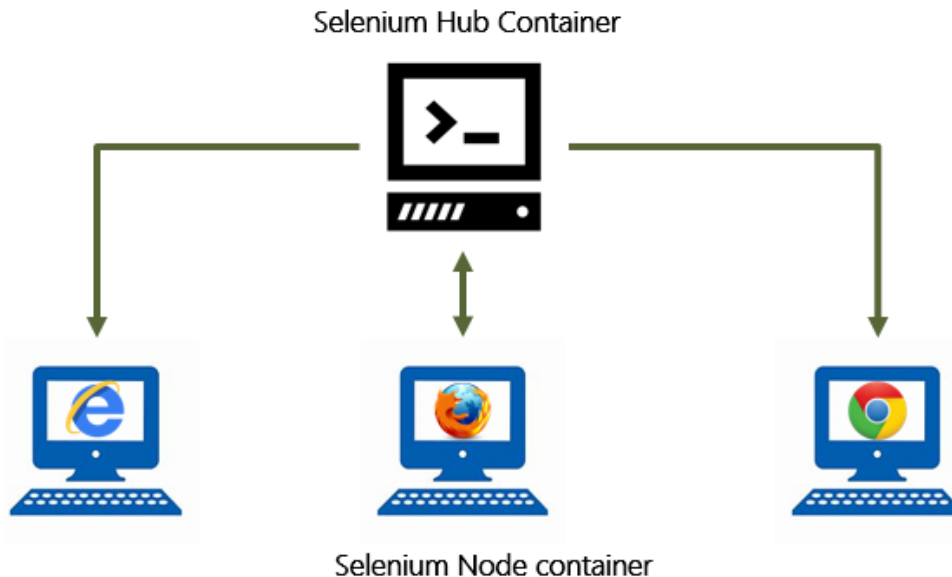
Chrome, Firefox

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

4.2 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

4.2 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!

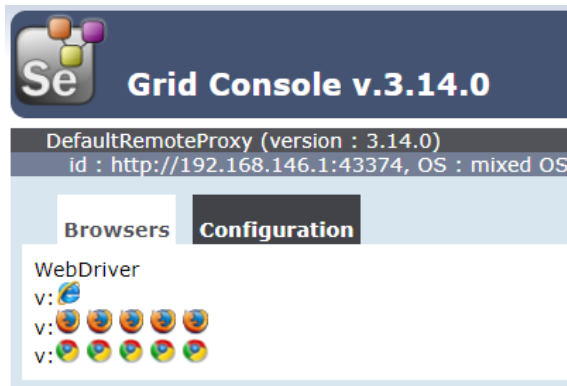
4.2 Selenium Grid

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](#)

4.2 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);
```


4.2 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
```


4. Selenium Grid

TP: Mettre une architecture en place



Chap. 5: Fonctions Utiles

- Pilotage des données
- Gestion des actions clavier
- Assertions

5.1 DataDriven

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.

5.1 DataDriven

JUnitParams

```
<dependency>
  <groupId>pl.pragmatists</groupId>
  <artifactId>JUnitParams</artifactId>
  <version>1.1.0</version>
</dependency>
```

```
@RunWith(JUnitParamsRunner.class)
public class SafeAdditionUtilTest {

    private SafeAdditionUtil serviceUnderTest
        = new SafeAdditionUtil();

    @Test
    @Parameters({
        "1, 2, 3",
        "-10, 30, 20",
        "15, -5, 10",
        "-5, -10, -15" })
    public void whenWithAnnotationProvidedParams_thenSafeAdd(
        int a, int b, int expectedValue) {

        assertEquals(expectedValue, serviceUnderTest.safeAdd(a, b));
    }
}
```

5.1 DataDriven

JunitParams: CSV

```
@Test
@FileParameters("src/test/resources/JunitParamsTestParameters.csv")
public void whenWithCsvFile_thenSafeAdd(
    int a, int b, int expectedValue) {

    assertEquals(expectedValue, serviceUnderTest.safeAdd(a, b));
}
```

5.1 DataDriven

```
FileInputStream excelFile = new FileInputStream(file);
Workbook workbook = new XSSFWorkbook(excelFile);
Sheet datatypeSheet = workbook.getSheetAt(0);
Iterator<Row> iterator = datatypeSheet.iterator();

int row = 0;
int column = 0;

while (iterator.hasNext()) {
    Row currentRow = iterator.next();
    Iterator<Cell> cellIterator = currentRow.iterator();
    data.add(column, new ArrayList<Object>());
    while (cellIterator.hasNext()) {
        Cell currentCell = cellIterator.next();
        //getCellTypeEnum shown as deprecated for version 3.15
        //getCellTypeEnum ill be renamed to getCellType starting from version 4.0

        if (currentCell.getCellTypeEnum() == CellType.STRING) {
            System.out.print(currentCell.getStringCellValue() + " | ");
            data.get(column).add(row, currentCell.getStringCellValue());
        } else if (currentCell.getCellTypeEnum() == CellType.NUMERIC) {
            double cellDoubleValue = currentCell.getNumericCellValue();
            // if my cell is an integer
            if ((cellDoubleValue % 1) == 0) {
                int cellIntValue = (int) cellDoubleValue;
                System.out.print(cellIntValue + " | ");
                data.get(column).add(row, cellIntValue);
            } else {
                System.out.print(cellDoubleValue + " | ");
                data.get(column).add(row, cellDoubleValue);
            }
        } else if (currentCell.getCellTypeEnum() == CellType.BLANK) {
            data.get(column).add(row, null);
        }
        row++;
    }
    row = 0;
    System.out.println();
    System.out.print("-----");
    column++;
    System.out.println();
}
```

5.1 DataDriven

Exercice H1: Paramétrage avec un tableau

- Scénario: Création de tiers

- Lancer l'application à l'adresse: <http://dolibarr.selenium-formation.org>
- Vérifier que le titre est Login contient « Login »
- Saisir Login=<username>
- Saisir Mot de passe=<password>
- Cliquer sur Connexion
- Cliquer sur **Tiers**
- Cliquer sur Nouveau tiers
- Saisir nom du tiers=<nomdutiers>
- Sélectionner Prospect/Client=Client
- Sélectionner Fournisseur=non
- Valider
- Se déconnecter
- Paramétrer cette classe avec la collection suivante:

username	password	nomDuTiers
jsmith	dolibarrhp	TOTO1
lsmith	olibarrhp	TOTO2

5.1 DataDriven

Exercice H2: Paramétrage avec un fichier CSV

- Créer le scénario suivant à partir d'une nouvelle classe:
 - 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

5.2 Action

Action Driver

- Classe ActionBuilder permet de faire les opérations de saisie de clavier et souris
 - click, contextClick, doubleClick, clickAndHold, dragAndDrop
 - keyUp, keyDown
 - move,
 - pause,
 - sendKeys
- Utilisation de Robot pour des actions hors du navigateur: browser

```
Actions builder = new Actions(driver);  
WebElement btnConnexion =driver.findElement(By.cssSelector("input.button"));  
builder.click(btnConnexion).perform();
```

```
builder.sendKeys(Keys.TAB).build().perform();  
builder.sendKeys("john").build().perform();  
.....
```

5.2 Action

Robot

- Utilisation de Robot pour des actions hors du navigateur: browser

```
public void saisirClavier(String texte) throws AWTException{
    StringSelection selection = new StringSelection(texte);
    Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
    clipboard.setContents(selection, selection);
    Robot robot = new Robot();
    robot.keyPress(KeyEvent.VK_CONTROL);
    robot.keyPress(KeyEvent.VK_V);
    robot.keyRelease(KeyEvent.VK_V);
    robot.keyRelease(KeyEvent.VK_CONTROL);
}

public void faireTab() throws AWTException{
    Robot robot = new Robot();
    robot.keyPress(KeyEvent.VK_TAB);
    robot.keyRelease(KeyEvent.VK_TAB);
}
```

```
@Test
public void testRegister2() throws InterruptedException, AWTException{
    WebDriver driver = new ChromeDriver();
    driver.get("http://newtours.demoaut.com");
    driver.findElement(By.linkText("REGISTER")).click();
    driver.findElement(By.name("firstName")).click();
    saisirClavier("test");
    faireTab();
    saisirClavier("test");
    faireTab();
    Thread.sleep(5000);
}
```

5.2 Action

Utiliser des fonctionnalités du langage Java

- For, While
- If
- Fonctions sur les String
- Fonctions sur les dates
- Fonctions arithmétiques
- Gestion des tableaux
- Portées des variables et propriétés d'une classe

5.2 Action

Exercice H4: Clavier souris

- Scénario: Connexion par actions clavier
 - Lancer chrome
 - Aller à l'adresse de <http://demo.testlogiciel.pro/dolibarr>
 - Faire Tabulation
 - Saisir Login
 - Faire Tabulation
 - Saisir Mot de passe
 - Faire Tabulation
 - Taper Entrer

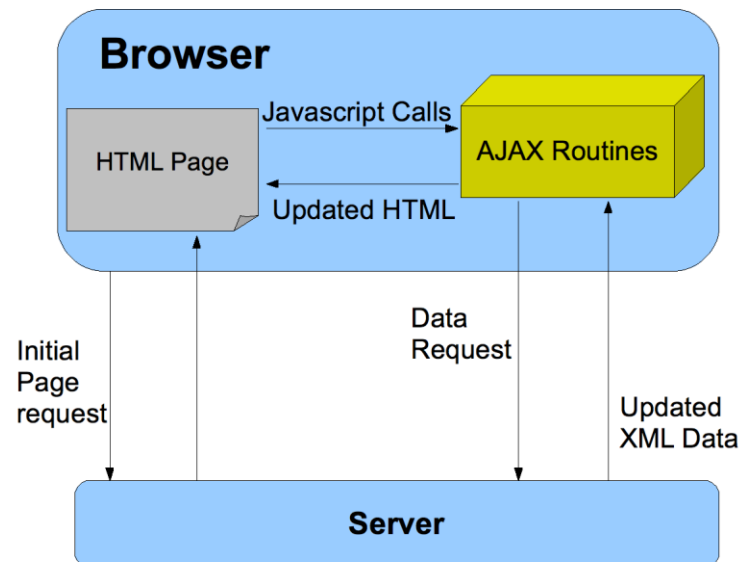
Chap. 6: Objets asynchrones

- Ajax
- HTML5
- Objets complexes

6.1 Ajax

Introduction

- 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



6.1 Ajax

Problématiques pour Selenium

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



6.1 Ajax

Synchronisation WaitFor.... (IDE)

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

Synchronisation Explicit package org.openqa.selenium.support.ui.

WebDriverWait & **ExpectedCondition** permettent de synchroniser les actions de selenium avec les réponses de l'application et donc gérer les réponses asynchrones

```
WebDriverWait wait = new WebDriverWait(driver, 10);
```

↓
TimeOut maximum

```
wait.until(  
    ExpectedConditions.visibilityOf(driver.findElement(By.linkText("John SMITH")))  
);
```

Condition de fin de synchronisation

6.1 Ajax

Attendre le traitement JQuery

```
public void waitForAjax(WebDriver driver) {  
    new WebDriverWait(driver, 180).until(new ExpectedCondition<Boolean>(){  
        public Boolean apply(WebDriver driver) {  
            JavascriptExecutor js = (JavascriptExecutor) driver;  
            return (Boolean) js.executeScript("return jQuery.active == 0");  
        }  
    });  
}
```

6.1 Ajax

JavaScript Executor est une interface permettant d'exécuter du Javascript à partir du WebDriver Selenium

- executeScript
- executeAsyncScript

```
JavascriptExecutor js = (JavascriptExecutor) driver;  
js.executeScript(Script, Arguments);
```

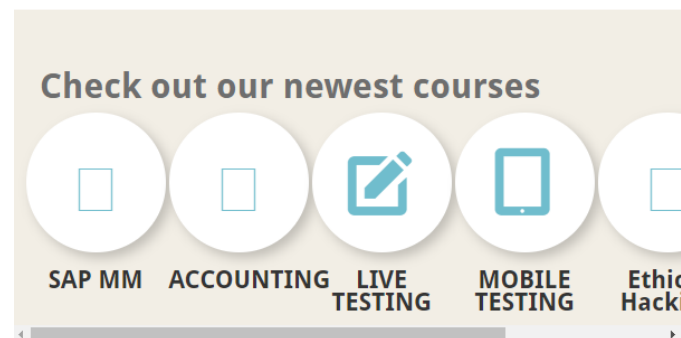
```
driver.get("https://demo-dolibarr.with.novafirstcloud.com");  
JavascriptExecutor js = (JavascriptExecutor) driver;  
js.executeScript(s: "document.getElementById('username').value='bsimpson';", ...objects: "");  
js.executeScript(s: "document.getElementById('username').style.backgroundColor='red';", ...objects: "");  
js.executeScript(s: "document.getElementById('password').value='homer';", ...objects: "");  
js.executeScript(s: "document.getElementById('password').style.backgroundColor='red';", ...objects: "");
```



6.1 Ajax

```
@Test
public void testScrolling() throws InterruptedException{
    ChromeDriver driver = new ChromeDriver();
    driver.navigate().to("http://demo.guru99.com/test/guru99home/scrolling.html");
    WebElement element1 = driver.findElement(By.className("icon-code-fork"));
    WebElement element2 = driver.findElement(By.className("icon-suitcase"));
    ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element1);
    Thread.sleep(3000);
    ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element2);
    Thread.sleep(3000);

    driver.close();
}
```



6.2 HTML5

Introduction



- Nouvelle spécification HTML
- Remplace HTML 4.01 (1999)
- Des nouvelles balises
- Plus d'interactions par les API (Drag & Drop, Stockage des données, Cache)

```
▼<body>
  ▶<header class="header">...</header>
  ▼<div class="main">
    ▶<aside class="aside">...</aside>
    ▶<article class="article">...</article>
    ▶<article class="article">...</article>
    <br style="clear:both;">
  </div>
  ▶<footer class="footer">...</footer>
</body>
```

6.2 HTML5

WebStorage



- Stockage sécurisé et amélioré des données au niveau du navigateur
 - Local: LocalStorage
 - Session: SessionStorage

```
WebStorage webStorage = (WebStorage) driver;  
LocalStorage localStorage = webStorage.getLocalStorage();  
localStorage.setItem("username", "bob");  
for(String key : localStorage.keySet())  
{  
    System.out.println(key + ": " + localStorage.getItem(key));  
}  
localStorage.clear();
```

```
driver.get("https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_webstorage_local_clickcount");  
JavascriptExecutor js = (JavascriptExecutor) driver;  
WebStorage webStorage = (WebStorage) driver;  
LocalStorage localStorage = webStorage.getLocalStorage();  
  
driver.switchTo().frame("iframeResult");  
driver.findElement(By.xpath("//button[@type='button']")).click();  
System.out.println("Click count: " + localStorage.getItem("clickcount"));  
driver.findElement(By.xpath("//button[@type='button']")).click();  
driver.findElement(By.xpath("//button[@type='button']")).click();  
System.out.println("Click count: " + localStorage.getItem("clickcount"));  
System.out.println("Click count: " + js.executeScript("return window.localStorage.getItem('clickcount')"));
```

6.2 HTML5

Iframe

- `switchTo().frame`



```
driver.get("https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_webstorage_local_clickcount");
JavascriptExecutor js = (JavascriptExecutor) driver;
WebStorage webStorage = (WebStorage) driver;
LocalStorage localStorage = webStorage.getLocalStorage();

driver.switchTo().frame(s: "iframeResult");
driver.findElement(By.xpath("//button[@type='button']")).click();
System.out.println("Click count: " + localStorage.getItem(s: "clickcount"));
driver.findElement(By.xpath("//button[@type='button']")).click();
driver.findElement(By.xpath("//button[@type='button']")).click();
System.out.println("Click count: " + localStorage.getItem(s: "clickcount"));
System.out.println("Click count: " + js.executeScript(s: "return window.localStorage.getItem('clickcount');"));
```

6.3 Objets complexes

Exercice I1: Gestion d'objets

- **Scénario: Ajouter un ordinateur**
 - Lancer l'application à l'adresse: <http://demo.glpi-project.org/index.php>
 - Se connecter avec le profil admin en français
 - Aller dans Parc > Ordinateurs
 - Ajouter un Ordinateur avec les informations:
 - Nom
 - Responsable technique: Young Elliot
 - Utilisateur: Fox Scott
 - Type: Laptop
 - Aller sur la page de recherche
 - Rechercher l'ordinateur
 - Vérifier que les données correspondent aux valeurs saisies précédemment
 - Se déconnecter

6.3 Objets complexes

Exercice I2: TP



Chap. 7: Mobile Testing



7. Mobile testing

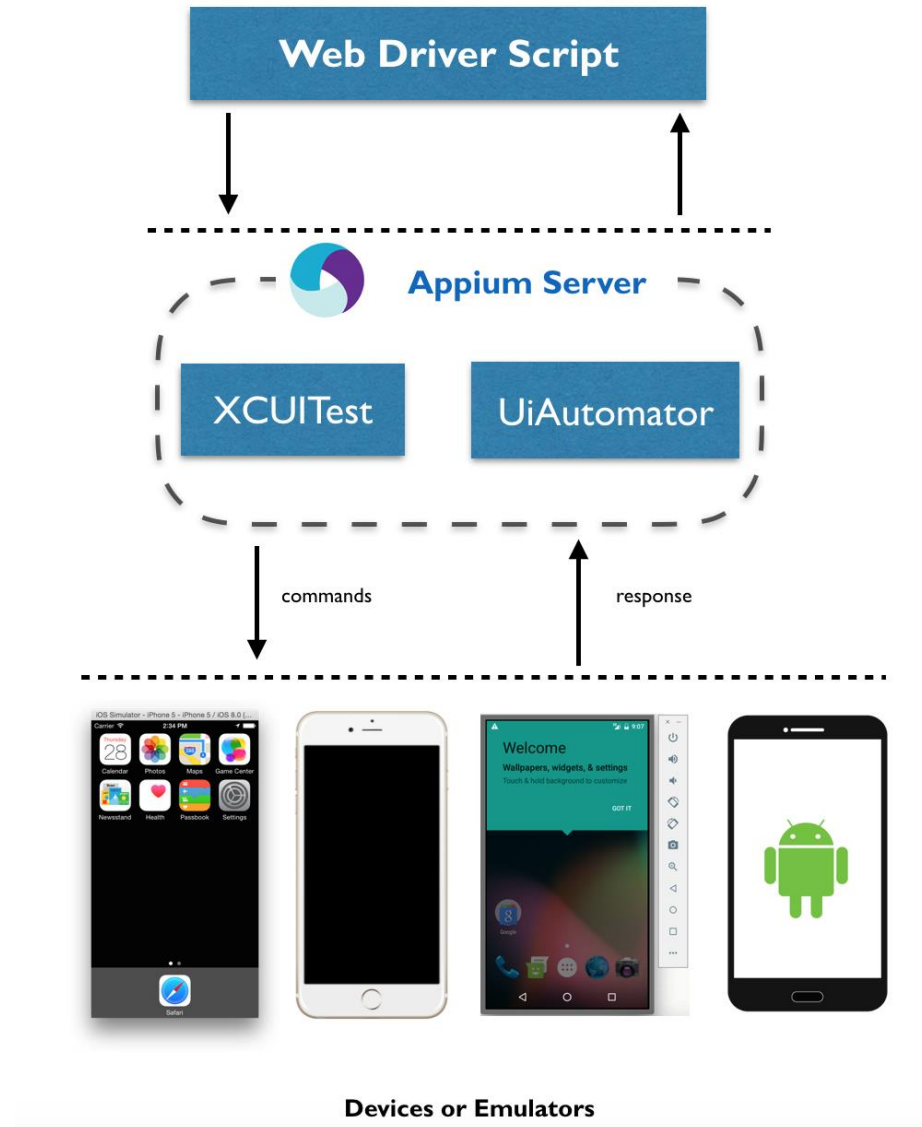
Introduction

- Framework de test pour les applications mobile créé en 2012
- Objectif: faciliter les tests mobiles en évitant la compilation du code de l'applcatif
- Construit autour de la technologie Selenium



7. Appium

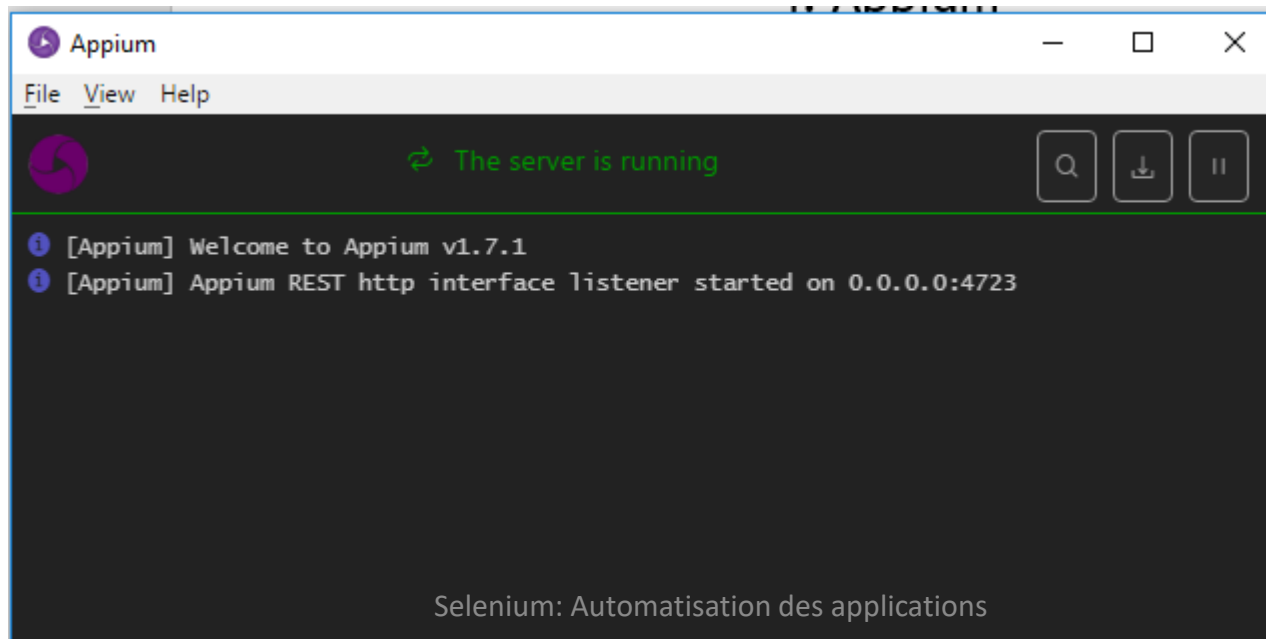
Architecture



7. Appium

Installation: Android

- Installation Android SDK
- Installation Appium Server (hub)
- Lancer Appium Server



7. Appium

Android: Capabilities

```
/**
 * Android Browser Local Test.
 */
public class AndoridBrowserLocalTest
{
    public static AndroidDriver<> mobiledriver;
    @BeforeTest
    public void beforeTest( ) throws MalformedURLException {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability(MobileCapabilityType.APPIUM_VERSION, "1.7.2");
        capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "4.4");
        capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "Android");
        capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "Appium");
        capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "Android Emulator");
        capabilities.setCapability(MobileCapabilityType.BROWSER_NAME, "Browser");
        capabilities.setCapability("newCommandTimeout", 2000);
        mobiledriver = new AndroidDriver<>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
    }
}
```

7. Appium

Android: Exemple de script

```
DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
desiredCapabilities.setCapability(MobileCapabilityType.BROWSER_NAME, value: "safari");
desiredCapabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, value: "Iphone");
desiredCapabilities.setCapability(MobileCapabilityType.DEVICE_NAME, value: "Iphone 6");
URL url = new URL( spec: "http://127.0.0.1:4723/wd/hub");
driver = new AppiumDriver(url, desiredCapabilities);
driver.get("https://demo-dolibarr.with.novafirstcloud.com");
driver.findElement(By.id("username")).sendKeys( ...charSequences: "admin");
driver.findElement(By.id("password")).sendKeys( ...charSequences: "Kanary1511");
driver.findElement(By.cssSelector("input.button")).click();
Assert.assertTrue(driver.findElement(By.tagName("body")).getText().contains("Espace accueil"));
driver.findElement(By.className("fa-sign-out")).click();
```

7. Appium

IOS: Installation

- MacOS
- XCode
- XCUITest library



7. Appium

IOS:Capabilities

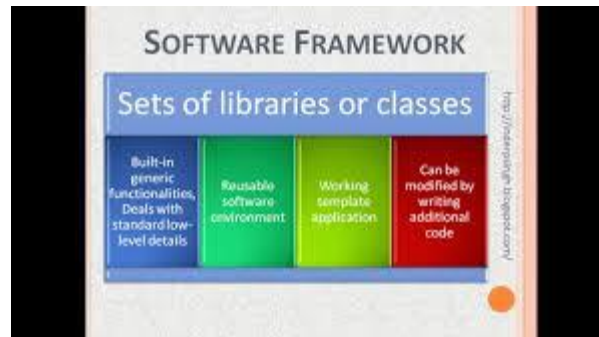
```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setCapability("platformName", "iOS");
capabilities.setCapability("deviceName", "iPhone 6");
capabilities.setCapability("platformVersion", "8.4");
capabilities.setCapability("app", "https://s3.amazonaws.com/appium/TestApp8.4.app.zip");
capabilities.setCapability("browserName", "");
capabilities.setCapability("deviceOrientation", "portrait");
capabilities.setCapability("appiumVersion", "1.5.3");

WebDriver driver = new IOSDriver(new URL(URL), capabilities);

/**
 * Test Actions here...
 */

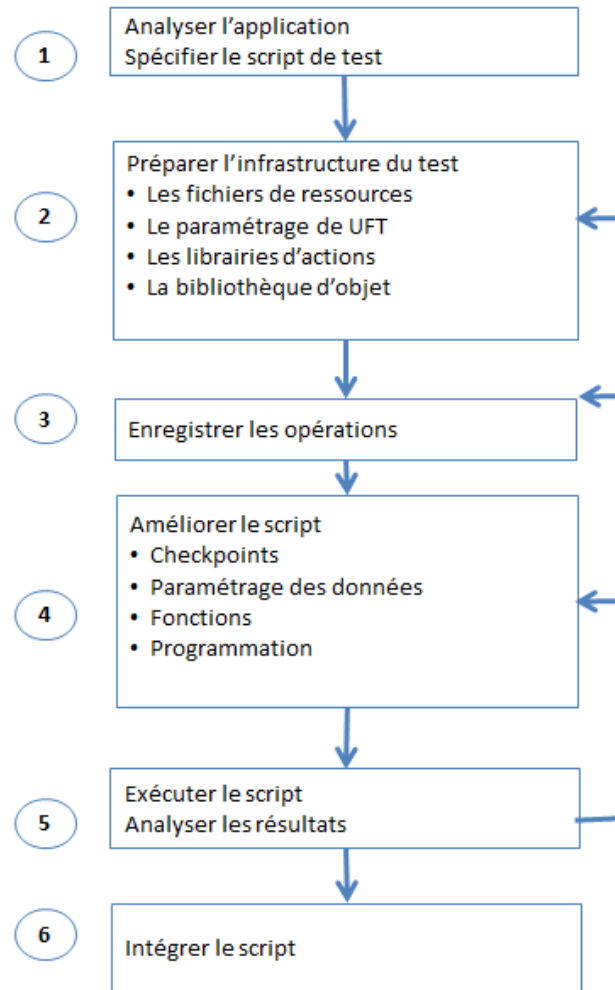
driver.quit();
```


Chap. 8: Framework



8. Framework

Workflow standard d'automatisation des tests:



8. Framework: Maintenir

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

Evolutions des applications

➔ Maintenance des scripts de test

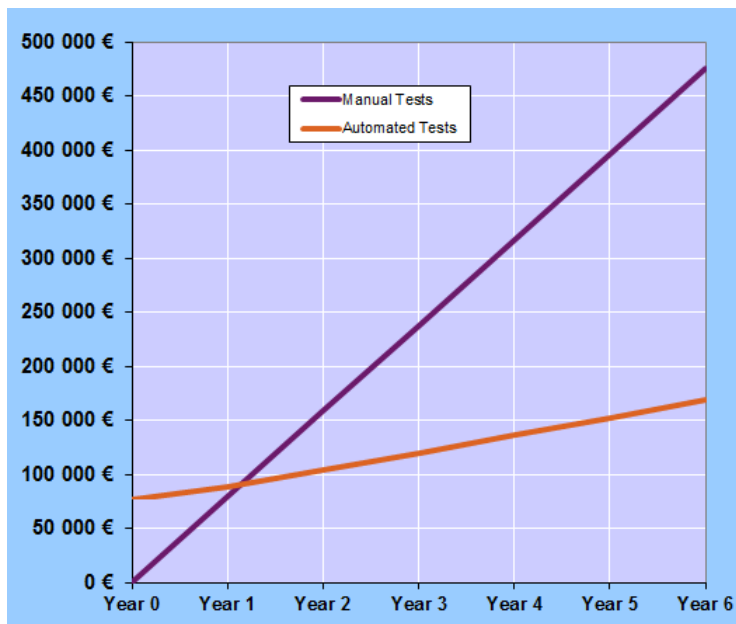
Processus de maintenance

- 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

8. 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 :

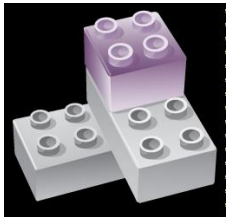
$$n = \text{DEV} / (\text{Exe M} - \text{Exe A} - A - \text{MAI})$$

8. 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?

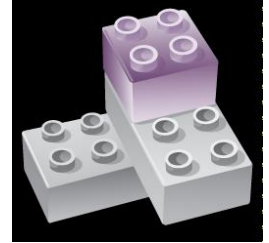


- Répondre aux problématiques soulevées (maintenance, robustesse)
- Permettre de dépasser certaines limites de l'outil d'automatisation
- Utiliser les fonctions de paramétrage pour répondre le plus efficacement à votre besoin
- Fournir une bibliothèque « ready-to-use » aux scripteurs
- Faciliter l'exécution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

8. Framework: Maintenir

Bénéfices

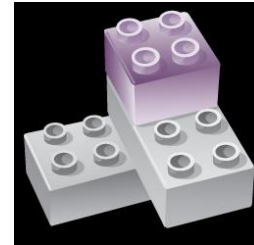
- Réduction de l'effort d'automatisation à la conception
- Réduction de l'effort d'automatisation à l'exécution
- Réduction de l'effort d'automatisation à la maintenance
- Faciliter le transfert de connaissance des scripts



8. Framework: Maintenir

Les éléments d'un framework

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



8. 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



8. 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

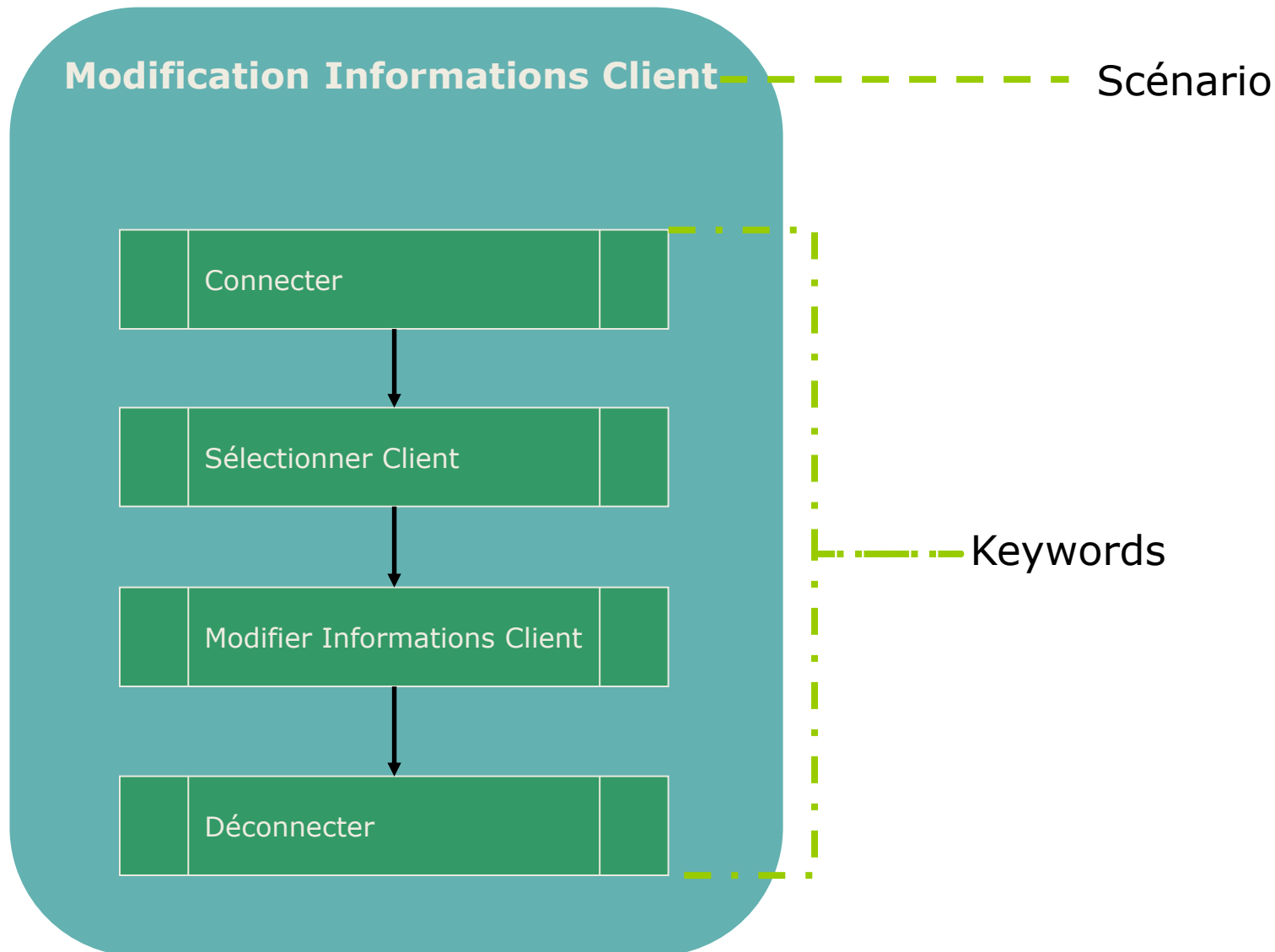
Il existe aussi des architectures hybrides

8. Framework: Architecture

Les composants d'une architecture d'automatisation

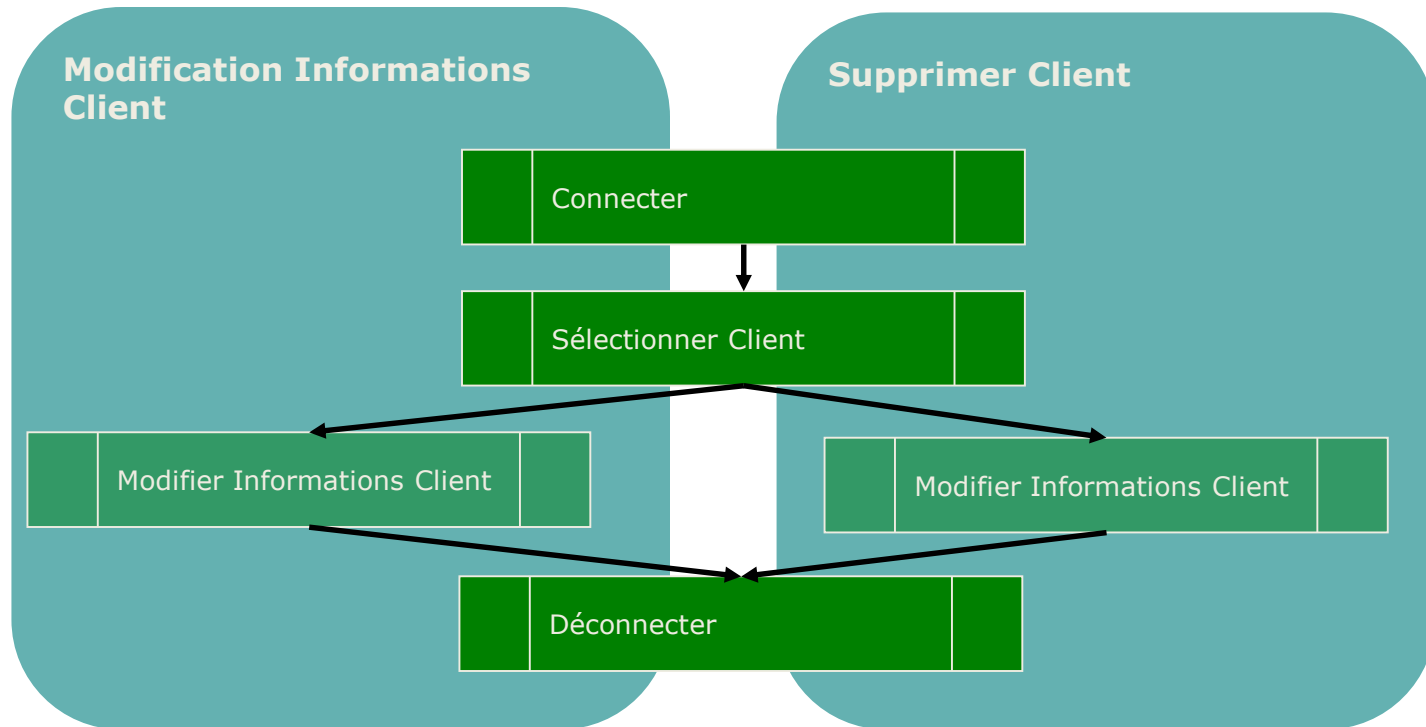
- L'externalisation des données de test/objets
 - Les données nécessaires à l'exécution du script dans un jeu dont la forme (compatibles avec l'outil) peut être un fichier Excel, XML, etc
 - Les données peuvent prendre en charge des résultats attendus à chaque point de vérification/contrôle
 - Les données 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

8. Framework: Keyword driven



8. Framework: Keyword driven

- Réduction du temps de l'automatisation



8. 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 librairies 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



