



Selenium IDE

Les bases pour un micro-projet

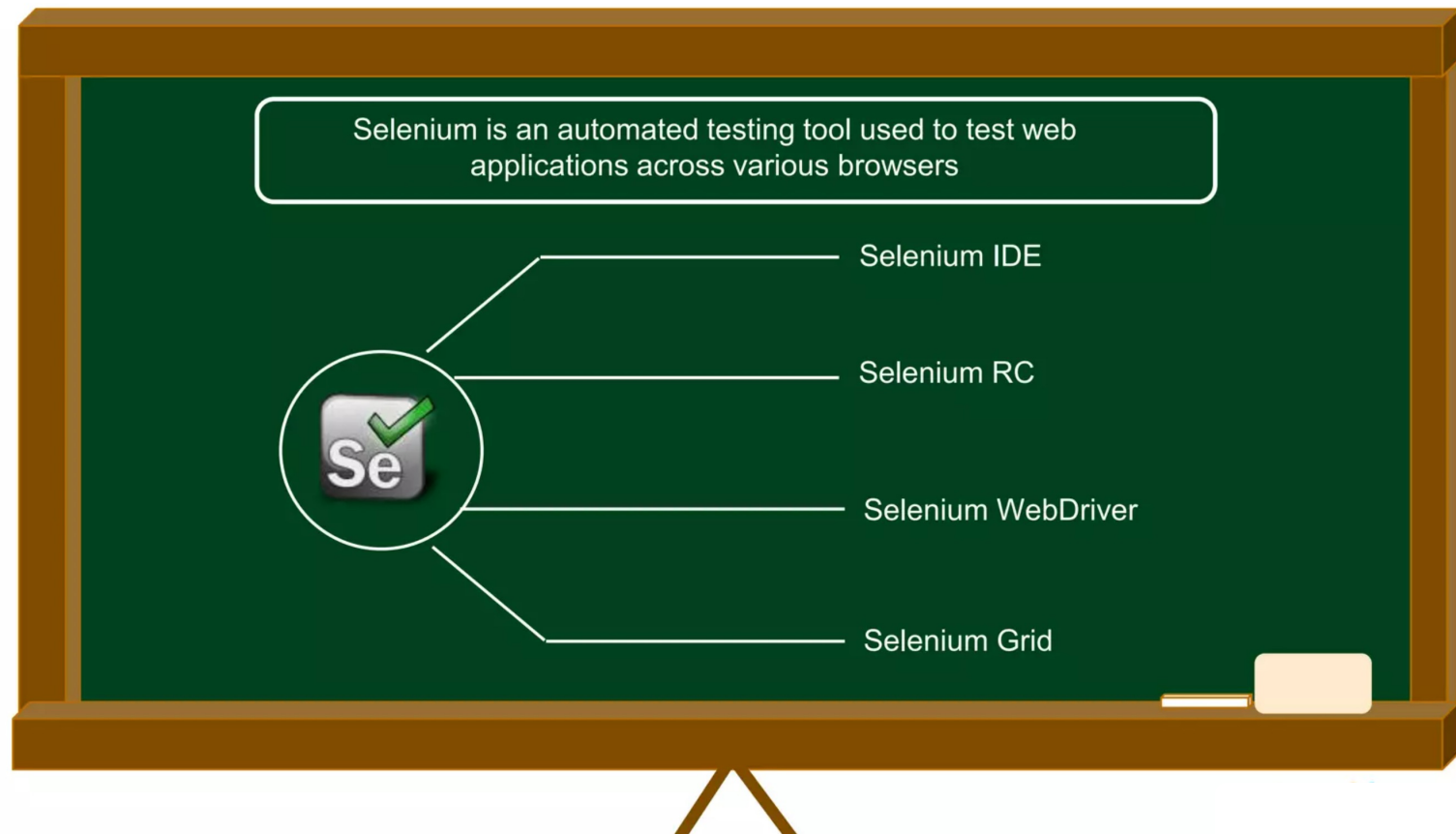


<https://www.selenium.dev/selenium-ide/>



SELENIUM

What is Selenium?



SELENIUM IDE

What is Selenium IDE?



Developed by **Shinya Kastani** in 2006



Firefox add-on that helps create tests, now many other browsers plugins



Easy-to-use interface to build automated test scripts

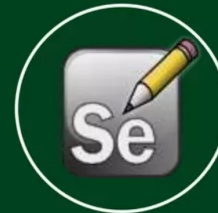


Records user interactions on the browser and exports them as a reusable script



Generally used as a prototyping tool

What is Selenium IDE?



Selenium IDE is a part of the Selenium suite and was developed to speed up the creation of automation scripts. It's a rapid prototyping tool and can be used by engineers with no programming knowledge whatsoever

SELENIUM IDE - Assets

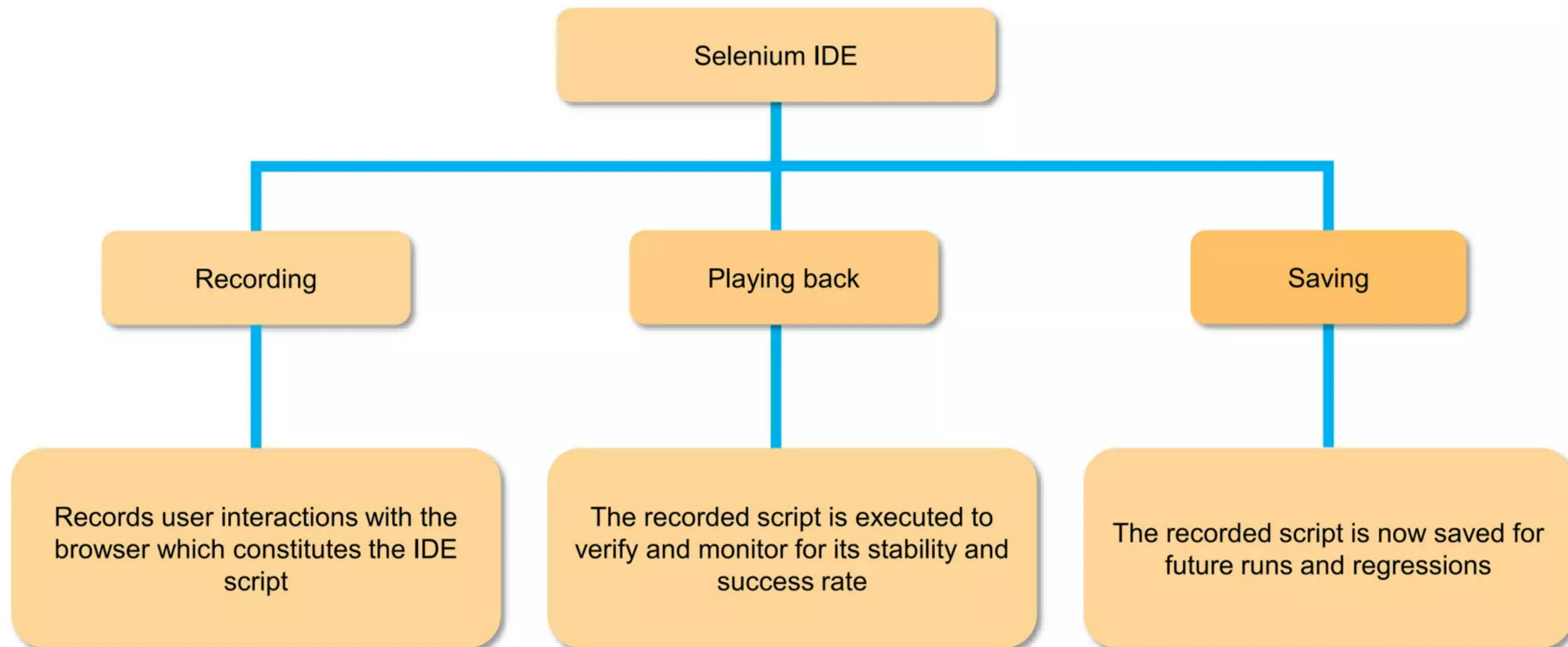


- Traditionally Selenium IDE was only a Firefox plugin. The new IDE supports both Chrome and Firefox, and many other browsers plugins
- Improved locator functionality
- Parallel execution of tests using Selenium command line runner
- Provision for control flow statements
- Automatically waits for page to load
- Supports embedded code-Runs JavaScripts
- IDE has a debugger which allows step execution, adding breakpoints
- The new version supports code exports

SELENIUM IDE - Assets

- Selenium is a software suite, based on Javascript, which allows to perform different types of testing (unit, integration, etc.) of web applications.
- Selenium allows you to manipulate the content of a page (completion of fields, click on buttons, etc.), to make assertions on Web pages.
- Selenium is able to manipulate the DOM structure whether it is known or not.

SELENIUM IDE - Principles



SELENIUM IDE - Components

The screenshot shows the Selenium IDE interface with the following components highlighted by dashed blue boxes and callouts:

- Menu bar:** Located at the top of the application window.
- Tool bar:** Located below the menu bar, containing icons for file operations and recording.
- Start/Stop recording bar:** A bar with play and stop icons, used to control the recording process.
- Address bar:** Shows the current URL being tested, `https://www.google.fr`.
- Test script editor box:** A table containing the test script commands.
- Test case panel:** A list on the left side showing test cases, with `testtestnamev1` selected.
- Commands and manually written data:** A form at the bottom for creating new commands, with fields for Command, Target, Value, and Description.
- Test log and reference:** A log window at the bottom showing the execution history and any errors.

Command	Target	Value
1 ✓ open	/	
2 ✓ set window size	868x699	
3 ✓ click	name=q	
4 ✓ type	name=q	stephane maag
5 ✓ send keys	name=q	\$(KEY_ENTER)

Runs: 2 Failures: 1

```
Running 'testtestnamev0' 14:04:26
1. open on / OK 14:04:26
2. setWindowSize on 868x699 OK 14:04:26
3. Trying to find css=#W0wltc > .QS5gu... Failed: 14:04:26
   Implicit Wait timed out after 30000ms
'testtestnamev0' ended with 1 error(s) 14:04:56
Running 'testtestnamev1' 14:04:56
```

Menu bar

Tool bar

Start/Stop recording bar

Address bar

Test script editor box

Test case panel

Commands and manually written data

Test log and reference

TP – micro projet

■ Objectifs:

- Effectuer des test unitaires en utilisant Selenium IDE
- Obtenir un script de test « signé » par un UIO (signant l'arrivée correct dans un état défini).
 - Ce script sera choisi et établi par vos soins en justifiant la signature UIO.

■ Votre travail sera mentionné en rouge dans la suite.

■ Lisez TOUT le TP avant de commencer!

➔ Toutes les démarches réalisées devront être explicitées dans un document séparé.

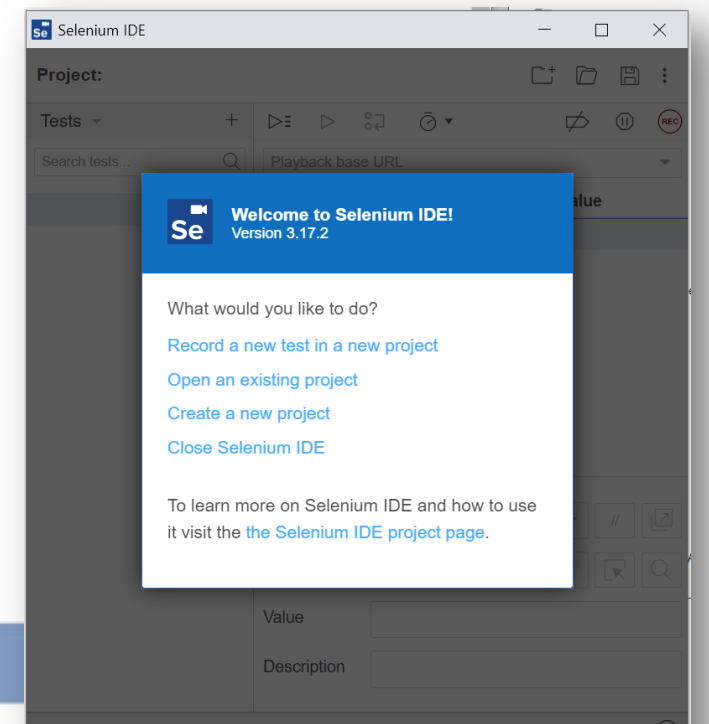
■ Installer Selenium IDE:

<https://www.selenium.dev/selenium-ide/>

■ Lancer Selenium IDE:

documentation:

<https://www.selenium.dev/selenium-ide/docs/en/introduction/getting-started>



TP – micro projet

1. **Click sur ‘Record a new test in a new project’**
2. **Donner un nom de projet**
3. **Entrer l’adresse ‘à tester’: <http://www.google.fr> → Start Recording**
4. **Une fenêtre navigateur s’ouvre, entrer votre prénom+nom dans la fenêtre google et cliquer sur la recherche.**
5. **Stopper l’enregistrement Selenium (bouton Stop).**
6. **Donner un nom à votre premier test.**
7. **Rejouer le test à différentes vitesses**
 1. Tentez de comprendre les blocages éventuels lors de la relecture du test

Toujours enregistrer son travail !!

TP – micro projet

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1274x1440	
3	✓ click	name=q	
4	✓ type	name=q	stephane
5	✓ send keys	name=q	\${KEY_DOWN}
6	✓ send keys	name=q	\${KEY_ENTER}

COMMANDS: <https://www.selenium.dev/selenium-ide/docs/en/api/commands>

■ les cas de test sont composés de commandes

- open: permet d'ouvrir un document donné en argument
- type: indique que l'on ajoute du texte dans un élément HTML
- send keys: permet de simuler une touche. Click correspond à l'action click sur un élément

■ Trois types de commandes:

- actions: telles que click, select. Ces actions ont souvent un suffixe *"AndWait"* pour ajouter une attente sur l'action (ex: attendre qu'une page se charge),
- accesseurs: actions sur l'état de l'application et sur des variables (stockage),
- assertion (et vérifications).

TP – micro projet

	Command	Target	Value
1	✓ <i>open</i>	/	
2	✓ <i>set window size</i>	1274x1440	
3	✓ <i>click</i>	name=q	
4	✓ <i>type</i>	name=q	stephane
5	✓ <i>send keys</i>	name=q	\${KEY_DOWN}
6	✓ <i>send keys</i>	name=q	\${KEY_ENTER}

- Dans le test précédent, il n'y a pas les assertions.
 - Possible de les ajouter manuellement ('à la main')
 - Possible de les ajouter dynamiquement pendant l'enregistrement (clic droit et utiliser les commandes proposées).
 - **Recréer un nouveau Test en y insérant une assertion de votre choix et expliquer le résultat.** <https://www.selenium.dev/selenium-ide/docs/en/api/commands#assert>

NB: l'assert `element present` est sans doute le plus simple à tester au début.

- Il se peut que l'assertion ne soit pas automatiquement intégrer dans le test case. Auquel cas, à vous de le mettre manuellement.
 - Target et Value peuvent enrichir le test case (<https://www.seleniumhq.org/selenium-ide/docs/en/introduction/faq/#how-can-i-use-regex-in-text-verifications>).

TP – micro projet

TEST CASE COMPLEXE – dans ce cas de test, voici ce que vous devrez faire:

- Choisir un site web (de **votre choix**) dans lequel vous soumettrez une requête (champ 'submit') quelconque (e.g., <https://fr.yahoo.com/>)
- Dans le cas de test, faites la recherche de 2 mots de votre choix
<https://www.seleniumhq.org/selenium-ide/docs/en/introduction/faq/#how-can-i-use-regex-in-text-verifications>
- Ajoutez une assertion pour vérifier si un 3^{ème} mot est présent.
- Ce cas de test doit retourner FAIL. Le cas de test stoppe son exécution.
- On peut toutefois utiliser d'autres commandes, à la place des assert, pour vérifier des propriétés et continuer le test, en utilisant les commandes de la famille *verify* (<https://www.selenium.dev/selenium-ide/docs/en/api/commands#verify>).
- Modifiez votre cas de test précédent en utilisant `verifyText` à la place de `assertText`.

- Faites un nouveau cas de test permettant de vous logger sur un site web de votre choix.

TP – micro projet

TEST CASE AVEC UIO – cas de test se terminant par une signature UIO:

Vous avez maintenant tous les outils vous permettant de construire un cas de test ‘signé’ par un UIO.

- Choisir la page/site web de votre choix. Ce choix doit être réfléchi.
 - Identifier un comportement unique dans cette page/site qui pourrait faire l’objet d’une signature UIO.
 - Créer un cas de test (celui de votre choix) se terminant par ce UIO
- ➔ **Toute la démarche devra être explicitée dans un document séparé.**

Micro TP évalué à rendre au PLUS TARD le vendredi le 17/11 à 18h00

stephane.maag@telecom-sudparis.eu

References

SimpliLearn – Selenium IDE
TP de S.Salva