SeU Syllabus Certified Selenium Engineer (CSE) (CSE)

Publié
Version 1.0 2020 FR

Selenium United

Droits d'auteur

Merci aux principaux collaborateurs

- Rahul Verma, Vipul Kocher, Chandra Mouli Maddala and Jayapradeep Jiothis

Merci pour votre soutien à la traduction française

- Jean Luc Cossi, Jean Julien Hessouh, Emilie Potin-Suau

Historique de révision

Version	Date	Remarques
SeU 2018	Août 2018	Première sortie officielle
SeU 1.0 2019	Janvier 2019	Version technologiquement neutre
SeU 1.0 2020 FR	Mai 2020	Version Française

Table des matières

Objet du document

Ce syllabus est le point de départ de Selenium Untied en ce qui concerne sa certification *Certified Selenium Engineer (CSE)*. Ce document définit ce que vous devez savoir pour passer l'examen de certification *Certified Selenium Engineer (CSE)*, et est la propriété de Selenium United. L'examen de certification ne portera que sur les concepts et les connaissances décrits dans ce document.

Ressources de Selenium United (SeU)

Un aperçu des ressources disponibles pour la certification SeU ainsi que toutes les informations pertinentes sont disponibles sur le site officiel de Selenium United - www.selenium-united.com. Les informations figurant sur le site www.selenium-united.org comprennent :

- Une liste complète des fournisseurs de formation SeU reconnus et des cours disponibles. Veuillez noter à ce propos que la formation est recommandée mais n'est pas obligatoire pour pouvoir se présenter à l'examen de certification SeU (CSE).
- Le syllabus SeU (ce document) à télécharger.
- Un exemple complet d'examen SeU CSE composé de 40 questions avec réponses à des fins de formation.
- Nous visons à ce que les documents soient disponibles dans d'autres langues dans les meilleurs délais. Pour les versions disponibles actuellement, veuillez consulter: www.selenium-united.com

Qu'est-ce que Selenium ?

Sélénium est une suite d'outils destinés à l'automatisation des navigateurs. Sa bibliothèque complète de tests d'applications web en fait l'outil le plus recherché par les testeurs de logiciels. Il prend en charge tous les navigateurs les plus utilisés et ne se résume plus à une simple bibliothèque de tests de logiciels. Il est la colonne vertébrale d'innombrables outils d'automatisation de navigateur, interfaces de programmation d'applications (API) et de Frameworks. L'interface WebDriver mise en place par Selenium WebDriver est amenée à devenir un standard du W3C, ce qui renforcerait encore son importance dans le domaine de l'automatisation des tests.

À propos de SeU Certified Selenium Engineer (CSE)

SeU Certified Selenium Engineer (CSE) est un cours destiné aux testeurs travaillant dans le domaine de l'automatisation des tests web (Practitioner level). Le cours aborde Selenium comme une bibliothèque d'automatisation de navigateur de bout en bout. La conception de l'automatisation des tests est maintenue au minimum afin de privilégier les concepts de codage qui permettent d'ultiliser Selenium de manière appropriée. Ce cours est axé sur la conception et fournit une expérience non diluée de Selenium. Il se concentre davantage sur les concepts critiques de Selenium, ce qui permet aux participants de mieux les mettre en œuvre dans leur travail quotidien.

Résultats Opérationnels

BO 1	Adapter l'expérience et les connaissances acquises en matière de tests et d'automatisation des tests afin de développer des tests automatisés pour les applications web utilisant.
BO 2	Utiliser Selenium afin de créer des tests automatisés pour les applications Web.
BO 3	Déboguer des tests automatisés avec Selenium pour qu'ils fonctionnent correctement.

Objectifs d'Apprentissage/Niveaux de Connaissances

Les objectifs d'Apprentissage (LOs) sont de brèves expressions qui décrivent ce que vous êtes censés savoir après avoir étudié chaque chapitre. Les LOs sont définis de la manière suivante :

- K1 : Se souvenir ;
- K2 : Comprendre ;
- K3 : Appliquer ou utiliser.

Le tableau suivant liste les objectifs d'apprentissage principaux pour la Certification SeU CSE :

LO1	Comprendre l'importance de la couverture du navigateur et distinguer les différentes options de test de l'interface utilisateur des applications web. (K2)
LO2	Comprendre la relation entre l'interface utilisateur Web et les langages HTML, JavaScript et CSS sous-jacents avec l'inspection du DOM (Document Object Model). (K2)
LO3	Rappel de l'histoire de Selenium et des divers outils de sa suite ainsi que leurs utilités respectives. (K1)
LO4	Comprendre l'architecture de Selenium d'un point de vue de l'interface de langage, de protocoles de communication et de pilotes. (K2)
LO5	Reconnaître l'objectif et les interfaces de programmation (API) de l'automatisation de l'interface utilisateur Web (UI) au niveau du navigateur, de la page et des éléments. (K2)

LO6	Comprendre et expliquer la structure des moteurs d'automatisation pour définir tests, fixtures et assertions. (K3)
LO7	Appliquer différentes stratégies pour identifier les éléments d'interface utilisateur. (K3)
LO8	Appliquer différentes stratégies de requête et d'interaction sur des éléments d'interface utilisateur. (K3)
LO9	Appliquer des structures ou modèles de code pour une automatisation plus avancée. (K3)
LO10	Automatiser les scénarios utilisateur de bout en bout en utilisant de bonnes pratiques de codage et des principes orientés objet, pour placer du code Selenium dans différentes classes et méthodes. (K3)
LO11	Diagnostiquer et décrire les possibilités d'amélioration au niveau de l'implémentation de l'automatisation, illustrées par des exemples de codes courts. (K3)

Prérequis du langage de programmation

La formation et le contenu de CSE peuvent être proposés dans n'importe quel langage de programmation supporté par Selenium. En fonction du langage de programmation choisi pour l'organisation du cours, la personne chargée de la formation fournira des informations sur les concepts de langage de programmation pertinents au fur et à mesure qu'ils seront utilisés durant le cours. Toutefois, ces concepts ne seront que mentionnés et non expliqués, à moins qu'une journée ne vienne s'ajouter à la durée de la formation afin de répondre à cette demande supplémentaire.

Les participants qui maitrisent bien les concepts de base de la programmation dans le langage utilisé comme support de cours pourront se concentrer davantage sur les concepts de Selenium, sans avoir à disperser leur attention pour comprendre les concepts de la programmation.

L'examen CSE comporte des extraits de code conformes au syllabus CSE. Actuellement, la formation CSE est disponible avec Java et Python.

Il est attendu des participants qu'ils aient une connaissance pratique des concepts suivants, dans le langage de programmation qu'ils ont choisi pour le CSE :

- Concept de “main”/point d’entrée d’exécution
- Compilation et exécution de code
- Les types de données primitifs, les classes correspondantes telles que supportées par le langage et les types définis par l'utilisateur au moyen de *Classes*
- *Arrays*
- Collections de base : *List*, *Map*/*Dictionary*, *Set*
- Formatage et manipulation de chaînes
- Impression vers *STDOUT* et *STDERR*
- Structures de contrôle conditionnel
- Structures de contrôle en boucle
- Traitement des exceptions et hiérarchie des exceptions
- Encapsulation : Modificateurs d'accès et méthodes d'accès
- Construction et initialisation des objets
- Accès au niveau de la classe ou de l’objet : Méthodes et variables
- Énumérations
- Création de paquets et de modules
- Héritage et polymorphisme
- Classes abstraites et méthodes abstraites
- Composition de l’objet
- Ajout de dépendances tierces (par exemple, pour Java, on peut ajouter des Jars comme configuration directe du projet ou via Maven)

Chapitre 1 — Automatisation de l'Interface Utilisateur des applications Web

LO1	Comprendre l'importance de la couverture du navigateur et distinguer les différentes options pour tester l'interface utilisateur des applications web. (K2)
LO2	Comprendre la relation entre l'interface utilisateur Web et les langages HTML, JavaScript et CSS sous-jacents avec l'inspection du DOM (Document Object Model). (K2)

Sommaire :

- Introduction
- Automatisation d'interface utilisateur avec des navigateurs réels
- Automatisation de l'interface utilisateur avec des navigateurs réels et une simulation de la taille de l'écran
- Automatisation d'interface utilisateur avec des navigateurs web sans interface graphique
- Interface utilisateur Web : point de vue utilisateur vs. point de vue navigateur

Contexte :

Les applications web doivent être accessibles à une variété d'utilisateurs qui emploient des appareils divers. Nombre d'entre eux utilisent des interfaces fluides, de sorte que les rendus de applications diffèrent selon les navigateurs. Les navigateurs eux-mêmes se comportent différemment dans certaines situations car ils ont des tailles de canevas et des technologies sous-jacentes distinctes.

Cela fait de la couverture du navigateur un aspect essentiel du test de l'interface utilisateur des applications web, qu'elle soit réalisée par un humain ou par un fragment de code. Un ingénieur en automatisation de tests avec Selenium doit comprendre comment utiliser Selenium pour automatiser différents navigateurs de même que les différentes options permettant d'imiter un navigateur. Pour des raisons de rapidité et d'efficacité, les ingénieurs doivent également considérer l'utilisation d'un navigateur sans interface (*Headless*) graphique dans l'automatisation.

Les navigateurs affichent les éléments de l'interface utilisateur qui sont représentés en *HTML* pour les charger dans un objet *DOM (Document Object Model)* et les représenter visuellement dans leur canevas, en utilisant les paramètres par défaut du navigateur et du style passés en *CSS*. JavaScript est utilisé pour la gestion d'événements, envoyant des requêtes *AJAX* au serveur et pour la manipulation du *DOM*. Un ingénieur en automatisation des tests doit être capable d'associer ce qui est vu à l'écran à sa définition en *HTML/DOM* en utilisant l'inspection *DOM*. Ces informations sont utilisées comme entrée dans le code d'automatisation de test pour identifier et interagir avec ces éléments.

Les applications web ont différents types d'interfaces utilisateurs qui prennent en charge différents types d'actions qu'un utilisateur peut réaliser. Par exemple, un utilisateur peut cliquer sur un bouton, entrer du texte dans une zone de texte, sélectionner une option dans une liste déroulante, etc. Un utilisateur peut également vérifier visuellement l'état d'un élément particulier de l'interface utilisateur, par exemple s'il est activé, le texte qu'il contient, etc. Un ingénieur en automatisation des tests doit pouvoir formuler toutes ces demandes et réaliser tous types d'actions dans le cadre d'un test automatisé.

Un ingénieur en automatisation de tests doit également connaître le style *CSS* et savoir dans quelles parties du HTML il peut être spécifié. Cela vaut également pour JavaScript, qui est le principal langage de script côté client, utilisé par les applications web modernes.

Chapitre 2 - Introduction à Selenium

LO3	Rappel de l'histoire de Selenium et des divers outils de sa suite ainsi que leurs utilités respectives. (K1)
LO4	Comprendre l'architecture de Selenium d'un point de vue interface de langage, de protocoles de communication et de pilotes. (K2)

Sommaire :

- Histoire
- Puissance et portée de Selenium
- Suite Selenium
- Architecture simplifiée de Selenium

Contexte :
Ses débuts à ThoughtWorks et les contributions ultérieures de divers membres de la communauté ont permis à Selenium de se hisser au premier rang de l'automatisation de l'interface utilisateur sur le Web.

L'API Selenium *WebDriver*, dont on dit qu'elle deviendra une norme du W3C, est utilisée par divers outils non-Selenium (indépendants) comme *Appium*, pour servir d'interface à son implémentation de base.

Il est bon pour un ingénieur en automatisation des tests de connaître l'histoire de Selenium et de se tenir informé de son évolution afin de savoir quand et pourquoi certaines fonctionnalités ont été introduites et quelles sont les perspectives d'avenir de Selenium et des outils connexes.

Selenium permet des liaisons multilingues grâce à une architecture qui répartit les responsabilités entre les liaisons client spécifiques à un langage et un composant de pilote spécifique au navigateur qui utilise le protocole JSON Wire. Comprendre cette architecture aide un ingénieur en automatisation des tests à déterminer quels composants fournissent quelles fonctionnalités.

Chapitre 3 - Automatisation d'Interface Utilisateur Web avec Selenium

LO5	Reconnaître l'objectif de l'automatisation de l'interface utilisateur Web et d'API au niveau du navigateur, de la page et des éléments. (K2)
LO6	Comprendre et expliquer la structure des moteurs d'automatisation des tests pour définir tests, fixtures et assertions (K3)
LO7	Appliquer différentes stratégies pour identifier les éléments d'interface utilisateur. (K3)
LO8	Appliquer différentes stratégies de requête et d'interaction sur des éléments d'interface utilisateur. (K3)

Sommaire :

Contexte :

Pour l'automatisation de l'interface utilisateur Web, vous avez besoin d'une API d'automatisation à 3 niveaux :

1. Niveau navigateur — Ces actions ne dépendant pas d'une page web en particulier et sont fournies par l'interface dans Selenium.

2. Niveau page — Ces actions dépendent de la page actuelle chargée dans le navigateur et sont également fournies par l'objet *WebDriver*.
3. Niveau élément de l'interface utilisateur — Ces actions sont liées à un élément particulier de l'interface utilisateur. Elles sont pour la plupart fournies par l'objet *WebElement*.

Un ingénieur en automatisation des tests être en mesure de déterminer les actions que Selenium prend en charge via son API, à chacun de ces niveaux. Il n'est pas attendu de l'ingénieur qu'il se souvienne de toutes les API ; il doit cependant être capable de se référer à l'API de Selenium et de définir le but des différents appels API, et de les utiliser alors qu'il rédige un code d'automatisation de test.

Dans le processus, l'ingénieur est censé savoir comment utiliser un moteur d'automatisation des tests pour exploiter différentes fonctionnalités standard telles que l'écriture des tests, leurs fixtures, les assertions, à utiliser avec le code Selenium afin d'écrire un test automatisé.

Pour l'identification des éléments, Selenium prend en charge plusieurs stratégies de localisation via son objet By :

- Par Nom
- Par ID
- Par Nom de classe
- Par Nom d'étiquette (Tag)
- Par Lien du Texte et par Lien Partiel du Texte
- Par Sélecteur CSS
- Par XPath

Tout ingénieur en automatisation des tests doit connaître le but de chacune de ces stratégies de localisation. Elle/il doit être capable de choisir la bonne stratégie de localisation pour une situation donnée. Les sélecteurs CSS et XPath sont extrêmement puissants et offrent de multiples moyens de définir une stratégie de localisation. L'ingénieur en automatisation de tests doit comprendre en profondeur les sélecteurs CSS et XPath, ainsi que les implications des différentes stratégies en termes de flexibilité et de performances.

Les interactions avec les éléments de l'interface utilisateur nécessitent également des requêtes avant et après les éléments pour vérifier si une action peut être effectuée et si elle a été effectuée avec succès, respectivement.

Chapitre 4 - Au-delà des simples constructions de code Selenium

LO9	Appliquer diverses constructions d'automatisation approfondies avec Selenium, qui jettent les bases d'une automatisation plus avancée. (K3)

Sommaire :

- Meilleure gestion des attentes
- Gestion des listes déroulantes
- Faire correspondre plusieurs éléments
- Gestion des éléments imbriqués
- Import/chargement de fichier
- Exécution de JavaScript
- Gestion des fenêtres/onglets
- Gestion des alertes
- Gestion des cadres
- Prise de capture d'écran
- Chaîne d'action
- Actions du clavier
- Gestion des cookies
- Navigation sans interface graphique

Contexte :

En raison de la latence du réseau et autres opérations côté client, l'interface utilisateur web peut ne pas être dans un état permettant d'effectuer une action souhaitée. Par exemple, on peut vouloir cliquer sur un bouton, mais ce bouton n'a pas encore été affiché, ou le bouton n'est pas encore cliquable. En pratique, un test automatisé contient un code permettant d'attendre un état souhaité avant d'effectuer une action. Cette attente est également nécessaire — le temps du chargement d'une page — avant qu'un élément ne puisse être identifié. Plutôt que d'utiliser des attentes statiques codées en dur, les tests devraient recourir à un mécanisme d'attente basé sur des attentes dynamiques, fourni par Selenium.

Selenium fournit un *Select Object* (objet/classe de sélection) pour activer l'*API* de niveau supérieur pour les listes déroulantes.

Dans certaines situations, il faudrait identifier de multiples éléments selon une stratégie de localisation et agir sur eux. Dans d'autres situations, les éléments ne se trouvent pas à la racine du DOM, mais plutôt comme des éléments enfants dans un élément. Selenium prend en charge l'automatisation de ces scénarios.

L'import ou le chargement de fichier est une fonctionnalité courante dans les applications web et l'automatisation des tests devrait pouvoir simuler cette fonctionnalité.

Selenium permet d'exécuter du JavaScript brut dans le contexte d'un navigateur. Il s'agit d'une fonctionnalité très pratique, grâce à laquelle un ingénieur d'automatisation de test peut injecter tout type de JavaScript personnalisé dans le navigateur et l'exécuter. On peut également passer des objets à un tel JavaScript à partir du code Selenium ainsi que recevoir un objet du JavaScript exécuté.

Les tests automatisés doivent également permettre de gérer les fenêtres/onglets, les alertes et les *IFrames*.

En cas d'échec, un test automatisé peut prendre une capture d'écran du contenu du navigateur afin que ces informations aident au dépannage ou recherche de causes. Selenium permet de faire de telles captures d'écran dans différents formats.

Certaines actions sont regroupées ensemble pour représenter une seule action. Par exemple, on peut survoler une barre de navigation, qui à son tour révèle un sous-menu et on peut cliquer sur un élément du sous-menu. Ces actions peuvent être simulées dans Selenium en utilisant une chaîne d'actions. Il est également utilisé pour simuler des actions complexes du clavier.

En général, une application web gère son état et les préférences de l'utilisateur à l'aide de cookies. Selenium donne accès aux cookies et des actions connexes peuvent être implémentées dans un test automatisé.

Selenium prend en charge la navigation sans interface graphique si un navigateur donné le permet. Chrome et Firefox disposent de cette option. *HtmlUnitDriver* est un navigateur développé spécialement en tant que navigateur sans interface graphique. Il s'agit d'une fonctionnalité très pratique qui peut être utilisée par les

ingénieurs en automatisation pour faire fonctionner de nombreux navigateurs en parallèle, avec une utilisation réduite des ressources par navigateur.

Chapitre 5 — Mettre en œuvre un Framework de base

LO10	Automatisez les scénarios utilisateur de bout en bout en utilisant de bonnes pratiques de codage et des principes orientés objet, pour placer du code Selenium dans différentes classes et méthodes. (K3)
LO11	Diagnostiquer et décrire des possibilités d'amélioration de mise en œuvre d'automatisation, illustrées par des exemples de codes courts. (K3)

Sommaire :

- Exercice long : Automatiser des scénarios de bout en bout.
- Refactorisation du code pour créer et utiliser une classe *WebAutomator* qui encapsule la construction et les interactions avec *WebDriver*
- Prochaines étapes : (Revue de haut niveau et Démonstration)
    - Implémenter *Event Listener*
    - Patron de conception *Page Object*
    - Utilisation des *Page Factories* et *Pages* comme des *Loadable Components* (disponible dans les interfaces de certains langages de programmation)
    - *Selenium Grid*

Contexte :

Sans la mise en œuvre de bonnes pratiques de programmation et de patrons de conception, un code automatisé est rapidement surchargé de nombreuses instructions dupliquées et de code difficile à maintenir. Des incohérences de codage se glissent également dans le code lorsque différents ingénieurs codent avec leur propre style spécifique.

Il est utile de placer toutes les structures couramment utilisées dans un cadre central, qui sert à écrire des tests. On peut utiliser les principes orientés objet pour créer très facilement un cadre de base et continuer à améliorer ses fonctionnalités au fur et à mesure de l'avancement du projet.

Bien que ce ne soit pas le sujet direct de ce cours, il est tout de même important de comprendre qu'il existe de nombreuses mesures qu'un ingénieur en automatisation de tests peut prendre pour mettre en place un cadre robuste. Il est donc nécessaire de connaître les concepts de *Page Object Model (POM)*, *Event*

Listener, *Selenium Grid*, etc. qui peuvent améliorer encore les caractéristiques et la robustesse d'une infrastructure logicielle de base. Certaines interfaces de programmation fournissent également des facilités supplémentaires comme *Page Factory* et *Loadable Component* pour faciliter le développement de *Page Object Model*. Ces concepts sont couverts à un niveau plus élevé dans la formation CSE afin que les participants puissent poursuivre leur apprentissage au-delà de ce programme. L'examen CSE ne couvre pas ces concepts.

Références

- Site Web SeleniumHQ: https://www.seleniumhq.org/
- Tout (Ce document) a été conçu et créé en utilisant des expériences directes recueillies dans l'industrie (le secteur) par les PME impliquées dans la création de Selenium United.