SeU Certified Selenium Engineer (CSE) Lehrplan

Herausgegeben auf Basis der
englischen Version 1.0 2019

Selenium United

Vielen Dank an die Autoren

- Rahul Verma, Vipul Kocher, Chandra Mouli Maddala und Jayapradeep Jiothis.

Vielen Dank an das deutschsprachige SeU Reviewkomitee

- Björn Lemke, Boris Wrubel, José Díaz, Marion Philippi, Manuel Fischer, Richard Seidl

Aus Gründen der Lesbarkeit wurde im vorliegenden Text die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf alle Geschlechter.

Revisionshistorie

Version	Datum	Bemerkungen
SeU 2018	August 2018	Erste offizielle Freigabe
SeU 1.0 2019	Jan 2019	Tech-Neutrale Version
SeU 1.0de 2019	Januar 2021	Erste deutsche Version

Inhaltsverzeichnis

Zweck dieses Dokuments

Dieser Lehrplan bildet die Grundlage von Selenium United für die Zertifizierung zum Certified Selenium Engineer (CSE). Dieses Dokument definiert die Lerninhalte, die Sie wissen müssen, um die Zertifizierungsprüfung zum Certified Selenium Engineer (CSE) zu bestehen, und ist durch SeU urheberrechtlich geschützt. Die Zertifizierungsprüfung umfasst nur Konzepte und Kenntnisse, die in diesem Dokument beschrieben sind.

Materialien der SeU

Eine Übersicht über die SeU-Materialien sowie alle relevanten Informationen über die CSE-Zertifizierung und andere Arten von SeU-Zertifizierungen finden Sie auf www.selenium-united.com - der offiziellen Website von Selenium United. Zu den Informationen auf www.selenium-united.org gehören:

- Eine vollständige Liste der anerkannten SeU-Schulungsanbieter und der verfügbaren Kurse. Für die Teilnahme an der Zertifizierungsprüfung SeU CSE wird eine Schulung empfohlen, die aber nicht erforderlich ist.
- SeU Lehrplan (dieses Dokument) zum Herunterladen.
- Ein komplettes Musterprüfungsset mit 40 SeU CSE Fragen und Antworten für Trainingszwecke.
- Wir sind bestrebt, die Dokumente so schnell wie möglich in weiteren Sprachen zur Verfügung zu haben. Die aktuell verfügbaren Sprachversionen finden Sie unter www.selenium-united.com.

Was ist Selenium?

Selenium ist eine Reihe verschiedener Werkzeuge zur Automatisierung von Browsern. Mit seiner umfassenden Bibliothek zum Testen von Webanwendungen hat es sich zum etabliertesten Werkzeug der Softwaretester entwickelt. Durch die Unterstützung aller gängigen Browser ist Selenium auch über die reine Software-Testbibliothek hinausgewachsen. Es ist das Rückgrat zahlreicher Browser-Automatisierungswerkzeuge, APIs und Frameworks. Die von Selenium WebDriver eingeführte WebDriver-Schnittestelle ist als W3C-Standard vorgeschlagen, was ihre Bedeutung im Rahmen der Testautomatisierung weiter erhöht.

Über SeU Certified Selenium Engineer (CSE)

SeU Certified Selenium Engineer (CSE) ist ein praxisbezogener Kurs für Tester, die sich mit der Automatisierung von Webtests befassen. Der Kurs behandelt Selenium als Browser-Automatisierungsbibliothek von Grund auf. Konzepte der Testautomatisierung und Testautomatisierungsdesign werden in dem Kurs auf ein Minimum reduziert, um sich mehr auf Code-Aspekte zu konzentrieren, die eine sinnvoll konzipierte Verwendung von Selenium ermöglichen. Dieser designorientierte Kurs vermittelt eine fokussierte Sicht auf Selenium und konzentriert sich mehr auf die kritischen Konzepte, die eine bessere Umsetzung in der täglichen Arbeit der Teilnehmer ermöglichen.

Business Outcomes (BOs)

BO 1	Anpassung der vorhandenen Erfahrungen und Kenntnisse im Bereich Testing & Testautomatisierung zur Entwicklung automatisierter Tests für Webanwendungen mit Selenium.
BO 2	Verwendung von Selenium, um Automatisierungstests für Webanwendungen zu erstellen.
BO 3	Debuggen Selenium-basierter automatisierter Tests um die korrekte Funktionalität sicherzustellen

Lernziele

Lernziele (Learning Objectives - LOs) sind kurze Aussagen, die beschreiben, was Sie im Kern über diese Inhalte wissen müssen. Die LOs werden in Wissensstufen klassifiziert:

- K1: Erinnern
- K2: Verstehen
- K3: Anwenden

Die folgende Tabelle listet die LOs für die SeU CSE-Zertifizierung auf:

LO1	Sie verstehen die Bedeutung der Browserabdeckung und können zwischen verschiedensten Optionen zum Testen der Benutzeroberfläche von Webanwendungen unterscheiden. (K2)
LO2	Sie verstehen die Abhängigkeit der Web-Benutzeroberfläche zu dem zugrunde liegenden HTML-, JavaScript- und CSS-Code mittels DOM Inspection. (K2)
LO3	Sie kennen die Entstehungsgeschichte von Selenium und können die verschiedenen Werkzeuge der Suite und deren Zweck erklären. (K1)
LO4	Sie verstehen die Selenium-Architektur in Bezug auf Sprachimplementierung (Languagebindings), Kommunikationsprotokolle und Treiber. (K2)
LO5	Sie verstehen die Zweckmäßigkeit und die API der Web UI Automation auf Browser-, Seiten- und Elementbasis. (K2)

LO6	Sie können den Aufbau und die Notwendigkeit des Aufbaus eines Testautomationsframeworks verstehen und erklären. (K3)
LO7	Sie können verschiedene Identifizierungsstrategien für UI-Elemente anwenden. (K3)
LO8	Sie können verschiedene Anfrage- und Interaktionsmethoden für UI-Elemente anwenden. (K3)
LO9	Sie können verschiedene Selenium-Automatisierungskonstrukte, die die Grundlage für eine weitergehende Automatisierung bilden, anwenden. (K3)
LO10	Sie können End-to-End-Benutzerszenarien automatisieren, indem Sie gute Programmiermethoden und objektorientierte Prinzipien verwenden, um Selenium-Code in verschiedenen Klassen und Methoden aufzuteilen. (K3)
LO11	Sie können Probleme beheben, sowie Verbesserungspotentiale erkennen und diese in kurzen Code-Beispielen beschreiben. (K3)

Voraussetzungen für Programmiersprachenkenntnisse

Der SeU CSE Workshop/Inhalt kann in jeder Programmiersprache durchgeführt werden, für die Selenium-Implementierungen verfügbar sind. Der Trainer wird für die gewählte Programmiersprache des Workshops auf die relevanten Konzepte und Eigenheiten im Kurs eingehen. Diese Konzepte werden jedoch nur erwähnt. Eine tiefere Erklärung findet nur statt, wenn ein zusätzlicher Workshop-Tag der Ausbildung hinzugefügt wird.

Teilnehmer, die bereits gut mit den grundlegenden Programmierkonzepten der verwendeten Programmiersprache vertraut sind, können sich viel besser auf die vorgestellten Selenium-Konzepte konzentrieren.

Die SeU CSE-Prüfung beinhaltet Codeausschnitte, die für den SeU CSE-Lehrplan relevant sind. Derzeit ist SeU CSE in Java und Python verfügbar.

Von den Teilnehmern wird erwartet, dass sie über Grundkenntnisse in folgenden Konzepten für die Programmiersprache verfügen, die sie für SeU CSE gewählt haben:

- Konzept des Einstiegspunktes "main"
- Kompilieren und Ausführen von Programmcode
- Primitive Datentypen, entsprechende von der Sprache unterstützte Klassen und benutzerdefinierte Typen über Klassen
- Arrays
- Generische Collections: List, Map & Set
- Zeichenkettenformatierung und -manipulation (Strings)
- Ausgabe nach STDOUT und STDERR
- Kontrollstrukturen (if, select, ...)
- Schleifen
- Exeptionhandling
- Verkapselung (Encapsulation): Zugriffsmodifikatoren und Zugriffsverfahren
- Objektkonstruktion und -initialisierung
- Klassen- versus Zugriff auf Objektebene: Methoden und Variablen
- Enumeration
- Erstellen von Paketen und Modulen
- Vererbung und Polymorphismus
- Abstrakte Klassen und abstrakte Methoden
- Objektzusammensetzung
- Abhängigkeiten zu 3rd-Party-Komponenten (z.B. für Java: Jars als direkte Projektkonfiguration oder via Maven)

Kapitel 1 - Automatisierung der **Weboberfläche**

LO1	Sie verstehen die Bedeutung der Browserabdeckung und können zwischen verschiedensten Optionen zum Testen der Benutzeroberfläche von Webanwendungen unterscheiden. (K2)
LO2	Sie verstehen die Abhängigkeit der Web-Benutzeroberfläche zu dem zugrunde liegenden HTML-, JavaScript- und CSS-Code mittels DOM Inspektion. (K2)

Übersicht:

- Einführung
- UI-Automatisierung mit aktuellen Browsern
- UI-Automatisierung mit aktuellen Browsern mit simulierter Bildschirmgröße
- Verwendung von Headless Browsern
- Web-UI: Perspektive Benutzer vs. Browser

Grundlage:
Webanwendungen müssen für eine Vielzahl von Benutzern über eine Vielzahl von Geräten verfügbar sein. Viele von ihnen verwenden fluid interfaces, so dass die Anwendungen für verschiedene Browser unterschiedlich dargestellt werden. Die Browser selbst verhalten sich in bestimmten Situationen unterschiedlich, da sie verschiedene Darstellungsgrößen und Technologien verwenden.

Daher ist die Browserabdeckung ein wesentlicher Aspekt beim Testen der Benutzeroberfläche von Webanwendungen, unabhängig davon, ob sie von einem Menschen oder einem Programm durchgeführt wird. Ein Selenium-Testautomatisierungsentwickler soll verstehen, wie man mit Selenium unterschiedliche Browser automatisieren kann, sowie die Möglichkeit Browser zu simulieren. Eine weitere Option, die der Entwickler kennen sollte, ist die Verwendung eines headless Browser der schneller und ressourcenschonender bei der Automatisierung arbeitet.

Die Browser stellen UI-Elemente dar, die in HTML definiert werden, um sie in ein DOM-Objekt (Document Object Model) zu laden und im Browserfenster

darzustellen, wobei Browser-Standards und Style-Overrides in CSS verwendet werden. JavaScript wird für die benutzerdefinierte Eventbehandlung, das Senden von AJAX-Anfragen an den Server und die DOM-Manipulation verwendet. Ein Testautomatisierungs-Engineer soll in der Lage sein, dass, was auf dem Bildschirm zu sehen ist, mit Hilfe der DOM-Inspektion der zugrundeliegenden Definition in HTML/DOM zuzuordnen. Diese Informationen werden als Input für den Testautomatisierungscode zur Identifizierung und Interaktion mit solchen Elementen verwendet.

Webanwendungen haben verschiedene Arten von UI Elementen, die verschiedenen Arten von Aktionen unterstützen, die ein Benutzer ausführen kann, z.B. kann ein Benutzer auf eine Schaltfläche (Button) klicken, Text in ein Textfeld eingeben, eine Option aus einer Auswahlliste auswählen, usw. Ein Benutzer kann auch den Zustand eines bestimmten UI Elements visuell überprüfen, z.B. ob es aktiviert ist, welchen Text es enthält, etc. Ein Testautomatisierungs-Engineer sollte in der Lage sein, alle derartigen Anfragen zu stellen und alle Arten von Maßnahmen in einem automatisierten Test durchzuführen.

Ein Testautomatisierungs-Engineer soll auch das CSS-Design kennen und wissen, in welchen Teilen von HTML es spezifiziert werden kann. Dies gilt auch für JavaScript, die primäre clientseitige Skriptsprache, die von modernen Webanwendungen verwendet wird.

Kapitel 2 - Einführung in Selenium

LO3	Sie kennen die Entstehungsgeschichte von Selenium und können die verschiedenen Werkzeuge der Suite und deren Zweck erklären. (K1)
LO4	Sie verstehen die Selenium-Architektur in Bezug auf Sprachimplementierung, Kommunikationsprotokolle und Treiber. (K2)

Übersicht:

- Entstehungsgeschichte von Selenium
- Welche Möglichkeiten bietet Selenium
- Selenium Suite
- Vereinfachte Selenium-Architektur

Grundlage:
Mit seinem Beginn bei ThoughtWorks und späteren Beiträgen aus der Community hat sich Selenium an die Spitze der Automatisierung von Web-Oberflächen (Web UI) gestellt. Die Selenium WebDriver API, die als W3C-Standard eingereicht wurde, wird auch von verschiedenen anderen Tools wie z.B. Appium verwendet, um die Schnittstelle zu ihrer zugrunde liegenden Implementierung bereitzustellen.

Es ist gut, die Entstehung von Selenium zu kennen und die laufende Entwicklung zu verfolgen damit ein Testautomatisierungs-Engineer weiß, wann und warum bestimmte Funktionen eingeführt wurden und wohin die Entwicklung von Selenium und den dazugehörigen Werkzeugen zukünftig geht.

Die Architekur von Selenium ermöglicht Implementierungen in unterschiedlichen Programmiersprachen. Dies ist möglich durch die Aufteilung in eine browserspezifische Komponente, die das JSON Wireprotokoll verwendet und die Komponente für die jeweilige Programmiersprache.

Das Verständnis dieser Architektur ist notwendig, um zu wissen, welche Komponenten welche Funktionalitäten bereitstellen.

Kapitel 3 - Automatisierung der Web-Oberfläche mit Selenium

LO5	Sie verstehen die Zweckmäßigkeit und die API der Web UI Automation auf Browser-, Seiten- und Elementbasis. (K2)
LO6	Sie können den Aufbau und die Notwendigkeit des Aufbaus eines Testautomationsframeworks verstehen und erklären. (K3)
LO7	Sie können verschiedene Identifizierungsstrategien für UI-Elemente anwenden. (K3)
LO8	Sie können verschiedene Anfrage- und Interaktionsmethoden für UI-Elemente anwenden. (K3)

Übersicht:

- Einführung
  - Automatisierung auf Browser-Ebene
  - Starten/Schließen verschiedener Browser
  - Browsernavigation
  - Fenster- und URL-Informationen abfragen
- Automatisierung auf Seitenebene
  - Informationen auf Seitenebene abfragen
  - Elementidentifikation im Detail
- Automatisierung auf Basis eines WebElements
  - Ermitteln der Eigenschaften
  - Grundlegende Aktionen

Grundlage:
Für die Automatisierung von Web UI benötigen Sie Zugriff auf die Automatisierungs-API auf drei unterschiedlichen Ebenen:

1. Browser-Ebene - Diese Aktionen sind unabhängig von einer bestimmten Webseite und werden über eine Schnittstelle in Selenium bereitgestellt.
2. Seitenebene - Diese Aktionen hängen von der aktuell im Browser geladenen Seite ab und werden auch vom WebDriver-Objekt bereitgestellt.

3. UI-Elementebene - Diese Aktionen beziehen sich auf ein bestimmtes UI-Element. Diese werden meist durch das WebElement-Objekt bereitgestellt.

Ein Testautomatisierungs-Engineer soll wissen, welche Aktionen Selenium über seine API bzw. über diese Ebenen hinweg unterstützt. Es wird nicht erwartet, dass sich der Testautomatisierungs-Engineer an die gesamte API erinnert. Jedoch muss er in der Lage sein, sich auf die Selenium-API zu beziehen und den Zweck verschiedener API-Aufrufe zu erkennen und sollte in der Lage sein, sie beim Schreiben eines Testautomatisierungscodes zu verwenden.

In diesem Prozess sollte man wissen, wie man eine Testautomatisierungs-Engine verwendet, um verschiedene Standardfunktionen wie Test Representation (Testdarstellung), Test Fixtures (Testvor- und Nachbereitung), Assertions (Wertvergleiche), die zusammen mit dem Selenium-Code verwendet werden können, um einen automatisierten Test zu schreiben.

Für die Identifizierung von Elementen unterstützt Selenium über sein By-Objekt verschiedene Identifizierungsstrategien:

- By Name
- By ID
- By Class Name
- By Tag Name
- By Link Text and Partial Link Text
- By CSS Selector
- By XPath

Ein Testautomatisierungs-Engineer muss den Zweck jeder dieser Identifizierungsstrategien kennen. Er/sie sollte in der Lage sein, die richtige Strategie für eine bestimmte Situation zu wählen. CSS Selectors und XPath sind extrem leistungsfähig und bieten umfangreiche Möglichkeiten, ein UI-Element zu identifizieren. Der Testautomatisierungs-Engineer soll CSS Selector und XPath eingehend verstehen und die Auswirkungen verschiedener Strategien auf Flexibilität und Leistung verstehen.

Interaktionen mit UI-Elementen erfordern auch Pre- und Post-Elementabfragen, um zu überprüfen, ob eine Maßnahme ergriffen werden kann bzw. ob eine Maßnahme erfolgreich durchgeführt wurde.

Kapitel 4 - Weiterführende Selenium-Code-Konstruktionen

LO9	Sie können verschiedene Selenium-Automatisierungskonstrukte, die die Grundlage für eine weitergehende Automatisierung bilden, anwenden. (K3)

Übersicht:

- Synchronisierungsstrategien
- Handhabung von Dropdown-Elementen
- Abgleich mehrerer Elemente
- Behandlung verschachtelter Elemente
- Hochladen einer Datei
- JavaScript-Ausführung
- Umgang mit Browserfenstern und Tabs
- Handhabung von Warnmeldungen (Alert)
- Umgang mit Frames
- Erstellung von Screenshots
- Ausführen von kombinierten Aktionen
- Tastaturaktionen
- Umgang mit Cookies
- Headless-Browser

Grundlage:
Aufgrund von Latenzzeit des Netzwerks und anderen clientseitigen Operationen befindet sich die Web-Benutzeroberfläche möglicherweise nicht in einem Zustand, in dem eine gewünschte Aktion durchgeführt werden kann. Beispielsweise könnte man auf eine Schaltfläche klicken, aber diese Schaltfläche wird noch nicht dargestellt, oder die Schaltfläche ist noch nicht anklickbar. Ein automatisierter Test enthält in der Praxis Code, um auf einen solchen gewünschten Zustand zu warten, bevor eine Aktion ausgeführt wird. Eine solche Wartezeit ist auch notwendig, bevor ein Element identifiziert werden kann, um auf das Laden der Seite zu warten. Anstatt hartkodiert statische Wartezeiten zu verwenden, sollten Tests einen dynamischen Wartemechanismus von Selenium verwenden.

Selenium stellt ein Select-Objekt mit einer erweiterten API für Dropdown-Listen zur Verfügung.

Es gibt Situationen, in denen man mehrere Elemente gemäß einer Locator-Strategie identifizieren, und darauf reagieren will. Es gibt andere Situationen, in denen Elemente nicht aus der Wurzel des DOM gefunden werden, sondern als untergeordnete Elemente innerhalb eines Elements. Selenium unterstützt die Automatisierung solcher Szenarien.

Das Hochladen einer Datei ist eine oft genutzte Funktion in Webanwendungen und der Automatisierungscode sollte in der Lage sein, diese zu simulieren.

Selenium unterstützt die Ausführung von JavaScript im Kontext eines Browsers. Dies ist sehr praktisch, denn damit kann der Testautomatisierungs-Engineer jede Art von benutzerdefiniertem JavaScript in den Browser einfügen und ausführen. Man kann auch Objekte an diesen JavaScript-Code aus Selenium heraus übergeben sowie ein Objekt aus dem ausgeführten JavaScript empfangen.

Selenium bietet weitere Funktionen, die für automatisierte Tests benötigt werden, wie die Handhabung von Fenstern/Tabs, Alerts und IFrames.

Wenn ein Fehler auftritt, kann ein automatisierter Test einen Screenshot des Inhalts des Browsers machen, so dass diese Informationen bei der Fehlersuche genutzt werden können. Selenium unterstützt die Erstellung solcher Screenshots in verschiedenen Formaten.

Einige Aktionen sind miteinander verknüpft, um eine einzige Aktion durchzuführen. Beispielsweise könnte man mit der Maus über eine Navigationsleiste fahren, die wiederum ein Untermenü anzeigt, um dann auf ein Untermenüelement zu klicken. Diese können in Selenium mit Hilfe einer Selenium-Action simuliert werden. Diese kommen auch zum Einsatz, um komplexe Tastaturaktionen zu simulieren.

In der Regel speichert eine Webanwendung ihren Status sowie Benutzereinstellungen in Cookies (siehe dazu https://de.wikipedia.org/wiki/HTTP-Cookie). Selenium bietet Zugriff auf Cookies und ähnliche Funktionen können in einem automatisierten Test implementiert werden.

Des Weiteren unterstützt Selenium sogenanntes Headless-Browsing, falls der verwendete Browser dies unterstützt. Chrome und Firefox haben eine Headless Option. HtmlUnitDriver ist ein Browser, der speziell als Headless-Browser entwickelt wurde. Dies ist sehr nützlich und wird oft von Automatisierungstechnikern genutzt, um viele Browser parallel zu betreiben, mit reduziertem Ressourcenverbrauch pro Browser.

Kapitel 5 - Zusammenstellung eines grundlegenden Rahmens

LO10	Sie können End-to-End-Benutzerszenarien automatisieren, indem Sie gute Programmiermethoden und objektorientierte Prinzipien verwenden, um Selenium-Code in verschiedenen Klassen und Methoden aufzuteilen. (K3)
LO11	Sie können Probleme beheben, sowie Verbesserungspotentiale erkennen und diese in kurzen Code-Beispielen beschreiben. (K3)

Übersicht:

- Ausführliche Übung: Automatisierung von End-to-End-Szenarien
- Refactoring des Programmcodes zur Erstellung und Verwendung einer WebAutomator-Klasse, die die Initialisierung und Interaktionen mit dem WebDriver abstrahiert.
- Nächste Schritte: (Übersicht und Demo)
    - Implementierung des Event Listeners
    - Page Object Design Pattern
    - Verwendung von Page Factories und Pages als ladbare Komponenten (in einigen Programmiersprachen verfügbar)
    - Selenium-Grid

Grundlage:
Ohne die Umsetzung guter Programmierpraktiken und Designmuster wird ein automatisierter Code schnell mit vielen doppelten Anweisungen unübersichtlich und schwer zu warten. Ebenso entstehen Inkonsistenzen im Code wenn verschiedene Programmierer in ihrem eigenen spezifischen Stil programmieren.

Es ist wichtig, alle mehrfach genutzten Funktionen in einem zentralen Framework bereitzustellen, das zum Schreiben von Tests verwendet wird. Mit den objektorientierten Prinzipien kann man sehr einfach ein Grundgerüst zusammenstellen und den Funktionsumfang im weiteren Projektverlauf weiter ausbauen.

Obwohl es nicht der direkte Themenbereich dieses Kurses ist, ist es dennoch wichtig zu verstehen, dass es viele Maßnahmen gibt, die ein Testautomatisierungs-Engineer ergreifen kann, um ein robustes Framework

zusammenzustellen. Man soll also Konzepte von Page Object Model (POM), Event Listener, Selenium Grid etc. kennen, die die Eigenschaften und Robustheit eines Frameworks weiter verbessern können. Einige Sprachimplementierungen bieten auch zusätzliche Funktionen wie Page Factory und Loadable Component, um die einfache Entwicklung des Page Object Model zu unterstützen. Diese Konzepte werden in dem CSE oberflächlich behandelt, so dass die Teilnehmer über diesen Kurs hinaus weiterlernen können. Die CSE-Prüfung deckt diese Konzepte nicht ab.

Referenzen:

- Selenium: https://www.selenium.dev/
- Die Inhalte wurde aus erster Hand auf Basis der Erfahrungen entworfen und erstellt, die von den Unternehmen, die an der Gründung von Selenium United beteiligt waren, gesammelt wurden.