Selenium IDE
Webdriver

Aan de slag

Het Wielsem 10, 5231 BW ’s-Hertogenbosch, telefoon 073-6409311
e-mail office@testwork.nl internet http://www.testwork.nl

1 Inhoudsopgave

2 Drie werkwijzen

Er zijn drie primaire methoden om testcases te ontwikkelen. Vaak heeft een ontwikkelaar of tester alle drie de technieken nodig.

2.1 Recording

Vele first-time gebruikers beginnen met het opnemen van een test case van hun interactie met een website. Wanneer Selenium IDE voor het eerst wordt geopend, is de record-knop ('red-dot') standaard ON. Als u niet wilt dat Selenium automatisch begint met opnemen dan kunt u dit uitschakelen door te gaan naar Options, Options… en dan de-selecteren "Start recording immediately on open".

Tijdens het opnemen zal Selenium opdrachten automatisch invoegen in uw testcase op basis van uw acties. Typische voorbeelden hiervan zijn:

- klikken op een link - click or clickAndWait commands
- waarden invoeren - type command
- opties selecteren in een drop-down listbox - select command
- klikken op selectievakjes of keuzerondjes - click command

Hier zijn enige "weetjes" om bewust van te zijn:

- Bij het commando Type zou het nodig kunnen zijn dat u op enige andere gebieden van de webpagina moet klikken om het op te nemen.
- Als u een koppeling volgt wordt er meestal een Click commando geregistreerd . Vaak moet u dit wijzigen in clickAndWait om ervoor te zorgen uw testcase pauzeert totdat de nieuwe pagina volledig geladen is. Anders blijft uw testcase opdrachten uitvoeren voordat de pagina is geladen met alle UI elementen. Dit zal leiden tot onverwachts mislukken van de testcase.

2.2 Assert en Verify commando's

Testcases zullen ook de eigenschappen van een web-pagina moeten kunnen controleren/valideren. Dit vereist *assert* en *verify* commando's.:

- Assert maakt een test mogelijk, welke controleert of een element op de web pagina is. Als dit element niet beschikbaar is, dan zal de test stoppen op de stap die faalt.
- Verify maakt ook een test mogelijk om te controleren of een element op de web pagina is, maar als dit element er niet is, dan zal de test niet stoppen, maar verder gaan met de uitvoering er van.

We zullen hier niet verder de specifieke kenmerken van deze opdrachten beschrijven; dat wordt in het hoofdstuk over Selenium commando's – "Selenese" gedaan. Hier zullen we eenvoudig beschrijven hoe deze opdrachten zijn toe te voegen aan een testcase.

Met Selenium IDE op opnemen, ga naar de browser waarin uw test applicatie wordt getoond en klik met de rechter muisknop ergens op de pagina. U ziet een context menu, welke weergeeft: verify en/of assert commando's.

De eerste keer dat u Selenium gebruikt wordt er alleen een Selenium commando getoond. Als u echter de IDE gebruikt, merkt u dat extra opdrachten snel zullen worden toegevoegd aan dit menu. Selenium IDE zal proberen te voorspellen welk commando, samen met de parameters, u voor een geselecteerd element van de UI op de huidige web-pagina nodig zou kunnen hebben.

Laten we eens kijken hoe dit werkt. Open een web-pagina van uw keuze en selecteer een tekst blok op de pagina. Een alinea of een kop voldoen. Klik nu met de rechter muisknop op de geselecteerde tekst. Het contextmenu moet u een verifyTextPresent commando geven en de voorgestelde parameter zou de tekst zelf moeten zijn.

Merk ook de menu-optie “Show All Available Commands” op. Deze toont vele meer commando’s, samen met de voorgestelde parameters voor het testen van uw geselecteerde UI-element.

Probeer meer van de UI elementen uit. Probeer met de rechter muisknop op een afbeelding of een besturingselement, zoals een knop of een selectievakje, te klikken. U moet mogelijk “Show All Available Commands” gebruiken om andere opties te zien dan “verifyTextPresent”. Zodra u deze andere opties selecteert, zullen de meer in het gebruik liggende opdrachten verschijnen in het primaire context menu. Bijvoorbeeld, het selecteren van “verifyElementPresent” voor een afbeelding zal later met zich mee brengen, dat die opdracht als u de volgende keer een afbeelding selecteert en klikt met de rechter muisknop beschikbaar is in het primaire context menu.

Nogmaals, deze opdrachten worden in detail behandeld in het hoofdstuk over de Selenium opdrachten. Gebruik voor nu echter gerust de IDE om opdrachten op te nemen en te selecteren in een testcase om deze vervolgens uit te voeren. Je kunt veel leren over de Selenium opdrachten door gewoon te experimenteren met IDE.

Tijd voor actie – Maak opdracht 1. Vul een testcase aan om items op een pagina te verifiëren en opdracht 2. Herontwerp de testcase met Verify en Assert commando’s.

2.3 Editing

2.3.1 Insert Command

Table View

Selecteer het punt in uw testcase waar u de opdracht wilt invoegen. U doet dit in het Test Case deelvenster, klik met de linker muisknop op de regel waar u een nieuwe opdracht wilt invoegen. Klik met de rechter muisknop en selecteer het Insert Command. IDE zal een lege regel toevoegen vlak voor de geselecteerde regel. Gebruik nu de bewerkings-/tekstvelden om het commando van het nieuwe commando en de bijbehorende parameters op te geven.

Source View

Selecteer het punt in uw testcase waar u de opdracht wilt invoegen. Dit kan in het Test Case deelvenster door links-klik tussen de commando's waar u een nieuwe opdracht wilt invoegen en voer de HTML-codes die nodig zijn voor het maken van een 3-kolom rij welke de opdracht, de eerste parameter (indien er een is vereist door de opdracht) en de tweede parameter (nogmaals, als vereist is) bevat. Zorg ervoor dat u uw test opslaat voordat u terug naar de Table view.

2.3.2 Insert Comment

Opmerkingen (Comments) kunnen worden toegevoegd aan uw testcase beter om deze (beter) leesbaar te maken. Deze opmerkingen worden genegeerd wanneer de test case wordt uitgevoerd.

Opmerkingen kunnen ook worden gebruikt om verticale ‘witte ruimte’ toe te voegen (een of meer lege regels) in uw tests; creëer gewoon lege opmerkingen. Een lege opdracht veroorzaakt een fout tijdens de uitvoering; een leeg commentaar niet.

Table View

Selecteer de regel in uw testcase waar u de opmerking wilt invoegen. Klik met de rechter muisknop en kies Insert Comment. Gebruik nu het Command veld om de opmerking in te geven. Uw opmerking verschijnt in paarse tekst.

Source View

Selecteer het punt in uw testcase waar u de opmerking wilt invoegen. Voeg een HTML-stijl commentaar toe, bijvoorbeeld `<!-- your comment here -->`.

Tijd voor actie – Maak opdracht 3. Voeg Selenium IDE comments toe.

2.3.3 Bewerken van een Command of Comment

Table View

Klik op de regel die moet worden gewijzigd en bewerk deze met behulp van Command, Target en Value velden.

Source view

Aangezien Source view het equivalent biedt van een WYSIWYG (What You See is What You Get) editor, wijzigt u simpelweg de gewenste regel – Command, Parameter of Comment.

3 Openen en bewaren van een testcase

Net als de meeste programma's zijn er “Save” en “Open” opdrachten in het menu Bestand (F). Selenium maakt echter onderscheid tussen testcases en testsuites. Om uw Selenium IDE tests op te slaan voor later gebruik kunt u ofwel individuele testcases opslaan of de testsuite opslaan. Als de testcases van uw testsuite nog niet zijn opgeslagen, wordt u gevraagd om hen te “saven” voordat u de testsuite opslaat. Als u een bestaande testcase of suite opent, toont Selenium IDE de Selenium-opdrachten in het Test Case deelvenster.

4 Meerdere browser vensters

Web applicaties gebruiken helaas niet altijd één venster van uw browser. Een voorbeeld hiervan zou kunnen zijn een site die rapportages weergeeft. De meeste rapporten hebben hun eigen venster zodat gebruikers gemakkelijk tussen deze schermen kunnen ‘switchen’.

Helaas, in test termen kan dit heel moeilijk zijn om uit te voeren, maar in deze paragraaf geven we een eerst kennismaking met het creëren van een test die zich tussen vensters verplaatst.

Werken met meerdere browser vensters kan een van de moeilijkste dingen zijn om te doen binnen een Selenium test. Dit ligt aan het feit dat de browser Selenium moet toestaan om programmatisch te weten hoeveel onderliggende browser processen worden voortgebracht.

In de volgende opdrachten zullen we zien, dat de test klikt op een element op de pagina en dat dat leidt tot het verschijnen van een nieuw venster. Als u een pop-up blocker actief heeft, terwijl u deze opdrachten uitvoert, dan kan het een goed idee zijn om deze uit te schakelen voor deze test site.

Tijd voor actie – Maak opdracht 4. Werken met een meerdere browser vensters (1) en opdracht 5. Werken met een meerdere browser vensters (2).

5 De waitFor Commands in AJAX applicaties

In AJAX driven web applicaties worden gegevens opgehaald van de server zonder de pagina te vernieuwen. De andWait opdrachten zullen niet werken als de pagina niet daadwerkelijk vernieuwd wordt. Onderbreken van de uitvoering van de test voor een bepaalde tijdperiode is ook niet een goede benadering, omdat het web element – afhankelijk van de systeem respons, belasting en andere ongecontroleerde factoren van het moment – eerder of later kan verschijnen dan de opgegeven tijdperiode , wat leidt tot storingen testen. De beste aanpak zou zijn om te wachten op het vereiste element in een dynamische periode en vervolgens verder te gaan met de uitvoering zodra het element wordt gevonden.

Dit wordt gedaan met behulp van “waitFor” opdrachten, zoals “waitForElementPresent” of “waitForVisible”, die wachten dynamisch, controleren elke seconde op de gewenste conditie en vervolgen met de volgende opdracht in het script zodra aan de conditie is voldaan.

De volgende commando’s zijn “waitFor” opdrachten (de lijst is niet uitputtend):

- waitForAlertNotPresent
- waitForAlertPresent
- waitForElementPresent
- waitForElementNotPresent
- waitForText,Present
- waitForTextNotPresent
- waitForPageToLoad
- waitForFrameToload

Een aantal van deze opdrachten worden impliciet uitgevoerd wanneer andere commando's worden uitgevoerd. Een voorbeeld hiervan is het commando clickAndwait. Dit zal een click opdracht afvuren en vervolgens een waitForPageToload afvuren. Een ander voorbeeld is de Open opdracht die alleen wordt voltooid wanneer de pagina volledig is geladen.

Tijd voor actie – Maak opdracht 6. Werken op pagina’s met AJAX.

6 Opslaan van informatie van de web pagina in de testcase

Soms is er een noodzaak voor het opslaan van elementen die op de pagina staan om deze later in een test te gebruiken. Dit zou kunnen zijn dat uw test een datum moet kiezen, die op de pagina staat, om deze later te gebruiken. Op deze wijze hoeft u niet deze waarden niet “hardcoded” in uw test op te nemen.

Zodra het element is opgeslagen zult u het opnieuw kunnen gebruiken door het aan te vragen uit een JavaScript dictionary, welke Selenium bijhoudt. Om de variabele te gebruiken is een van de volgende twee notaties nodig: het kan lijken op `${variableName}` of `storedvars ['variableName']`. Ik verkies de `storedvars` indeling, omdat het dezelfde indeling volgt als binnen de Selenium internals.

Maak, om te zien hoe dit werkt, de onderstaande opdracht.

Tijd voor actie – Maak opdracht 7. Opslaan van waarden vanuit de webpagina.

7 Het debuggen van tests

We hebben een aantal tests met succes gemaakt en heb gezien hoe we tegen AJAX-toepassingen kunnen werken maar helaas, het maken van tests die een eerste keer perfect draaien kan moeilijk zijn. Soms, als een testtool specialist moet u uw tests debuggen om te zien wat er mis is.

Als u door dit deel van het hoofdstuk wilt werken, zult u een testcase geopend moeten hebben in Selenium IDE.

Deze twee stappen zijn vrij nuttig wanneer uw tests niet draaien en u een specifieke opdracht wilt uitvoeren.

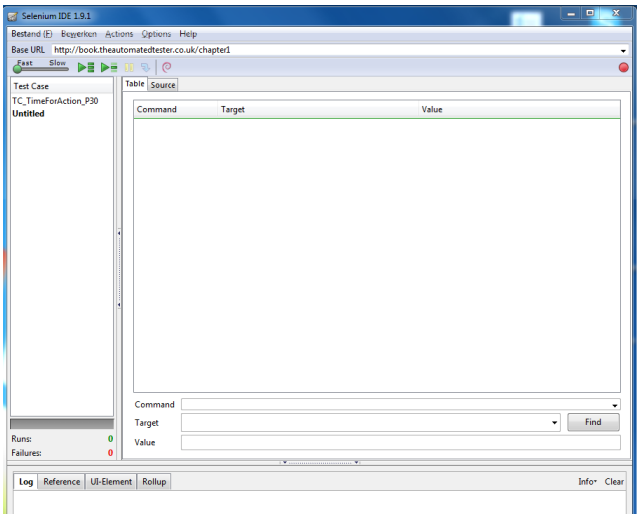
1. Highlight een commando.
2. Klik op de X toets. Hierdoor wordt de opdracht in Selenium IDE uitgevoerd.

8 Test suites

We hebben een aantal testcases kunnen maken met behulp van Selenium IDE en zijn erin geslaagd om deze met succes uit te voeren. Het volgende ding om te bekijken is hoe je een testsuite maakt, zodat we kunnen de testsuite openen en vervolgens hebben het uitvoeren van een aantal tests die we hebben gemaakt.

Als u Selenium IDE open heeft met een testcase, dan klikt u op het menu bestand:

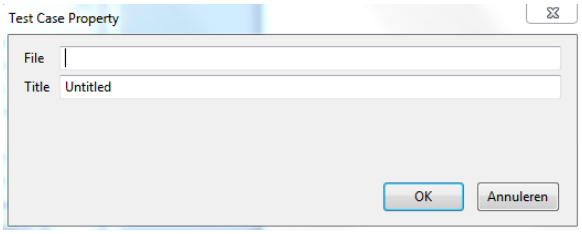
1. Klik “New Test Case”.
2. U zult zien, dat Selenium IDE een nieuw gebied opent aan de linkerzijde van IDE zoals (ook) te zien in het onderstaande screenshot:



U kunt dit zo vaak doen als u wilt en als u op de knop “Play entire test suite” klikt worden alle testcases uitgevoerd in de testsuite. Alle verwerken en fouten zullen worden gelogd in het onderste gedeelte van de testcase box.

Om dit op te slaan dit, klik op het menu Bestand (F) en klik op “Save Test Suite” en sla het Test Suite bestand op. Een ding kan hierbij worden opgemerkt en dat is, dat met het opslaan van een testsuite *niet* de testcase wordt opgeslagen. Zorg ervoor dat u niet alleen de test suite op slaat, maar dat u de test case opslaat elke keer als u een verandering heeft aanbracht.

Als u de naam van de test case wilt veranderen in een meer zinvolle naam, dan kunt u dit doen door rechts te klikken op de test en te klikken op de optie “Properties” het context menu:



U kunt nu betekenisvolle namen aan uw tests geven en deze zal dan verschijnen in Selenium IDE.

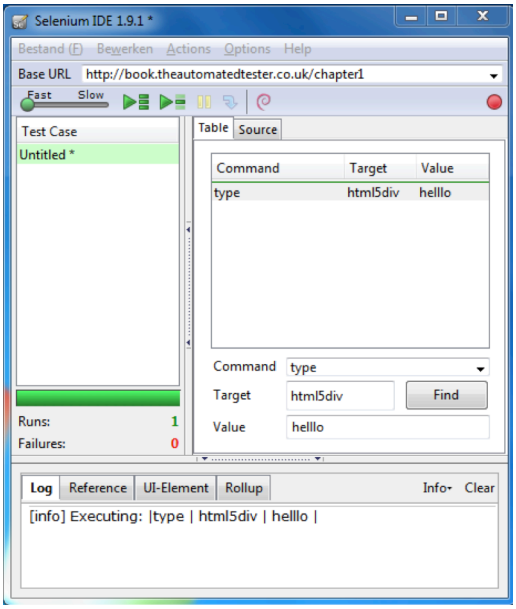
9 Testcases opslaan

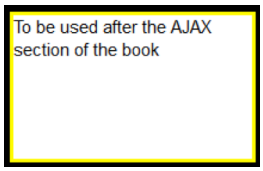
Opslaan van een testcase gebeurt op dezelfde wijze als het opslaan van een test suite. Klik op het menu Bestand (F) en klik vervolgens op “Save Test Case”. Dan krijgt u een “save” dialoog, waarmee u de testcase ergens kunt opslaan. Wanneer u uw testcases en uw testsuite opslaat, dan zal Selenium IDE proberen de relaties/volgorde te behouden tussen de testcases in de test suite.

10 Wat je niet kunt opnemen met Selenium

We hebben gezien dat onze tests goed werken als ze worden opgenomen en ze vervolgens wordne afgespeeld. Helaas zijn er een aantal dingen die Selenium niet kan doen. Aangezien Selenium werd ontwikkeld in JavaScript, probeert het te synthetiseren wat de gebruiker doet met JavaScript events. Jammer genoeg betekent dit, dat Selenium door te functioneren binnen de sandbox gebonden is aan dezelfde regels die JavaScript in alle browsers heeft.

- Silverlight en Flex/Flash toepassingen kunnen bij het schrijven van document niet worden opgenomen met Selenium IDE. Deze beide technologieën werken in hun eigen sandbox en werken niet met het DOM om hun werk te doen.
- HTML 5 wordt op het moment van schrijven niet volledig ondersteund met Selenium IDE. Een goed voorbeeld hiervan zijn elementen die de `contentEditable=true` attribuut hebben. Als u dit wilt zien, dan kunt u het “Type” commando gebruiken om iets te typen iets in het `html5div` element. De test zal u vertellen dat het de opdracht heeft voltooid, maar de UI zal niet zijn gewijzigd, zoals in de volgende afdruk van het scherm is te zien:





(Deel van scherm)

- Selenium IDE werkt ook niet met Canvas elementen op de pagina, dus is het voor u niet mogelijk om uw tests items te laten verplaatsen op een pagina.
- Selenium kan geen bestand uploads aan. Dit is te wijten aan het feit, dat de JavaScript sandbox niet toestaat dat JavaScript communiceert met `<input type=file>` elementen op een pagina. Terwijl het wellicht mogelijk is dat u de tekst verstuurt naar de box zal het niet altijd doen wat u verwacht, dus ik zou adviseren om het niet te doen.

Wij zijn in staat om een aantal van deze elementen met Selenium WebDriver in latere hoofdstukken van dit boek te automatiseren.

11 Aan de slag

Geïnteresseerd in de opdrachten? E-mail ons dan even: info@testwork.nl.