« ¿Cómo prepararse para un futuro en pruebas de software? ¿Cómo crear una matriz de rastreabilidad de requisitos (R

Selenium Core Extensions - Selenium Tutorial

En este capítulo, vamos a hablar sobre las extensiones principales de Selenium y las extensiones definidas por el usuario. Antes, podemos comenzar, vamos a hacer una recapitulación rápida de las extensiones presentes en Selenium IDE, como discutimos en el capítulo 2 de este tutorial.

1) Acción: ordena a Selenium IDE realizar una acción, es decir, acceder a la interfaz de usuario (interfaz de usuario) en la página web. Por ejemplo, doubleClick, ClickAndWait, dragAndDrop, dragAndDropAndWait, etc.

2) Evaluadores / Afirmación: Define las verificaciones que se realizarán en los datos recibidos de la IU después de ejecutar los siguientes comandos, como verifyElementPresent, verifyElementWidth, verifyExpression, verifyHTMLSource, verifyLocation, verifyMouseSpeed, etc.

3) Estrategia de localización: define una estrategia para ubicar un elemento web en la página web. Podría ser por nombre, ID, CSS, etiqueta, XPath, etc.

Hay muchas cosas que se pueden hacer usando Selenium IDE, pero aún así, se requiere agregar algunas funcionalidades adicionales en las pruebas para nuestro proyecto. En tales escenarios, podemos agregar nuestras extensiones personalizadas en la parte superior de la biblioteca que también se conocen como 'Extensiones de usuario'.

Por ejemplo, si necesitamos realizar una acción mientras probamos en un elemento web para convertir el texto a minúsculas antes de con los detalles. No existe tal acción en la biblioteca Action para Selenium IDE. Por lo tanto, podemos crear nuestra extensión de usuario para convertir el texto a minúsculas.

Creación de extensión de usuario:

Antes de que podamos comenzar a trabajar en la extensión de usuario para Selenium IDE, deberíamos estar bien versados en JavaScript conceptos de objetos prototipo, como se muestra a continuación:

```
1 Selenium.prototype.doTextLowerCase = function(locator, text)
```

To create a customized 'User Extension' in IDE, we first need to create a JavaScript method and later we need to add this method to the Se object prototype and PageBot object prototype. Once added we can restart Selenium IDE and we can see all these extensions that were a JavaScript prototype are recognized by Selenium IDE with their name (here ***doTextLowerCase***, etc..). User-defined extension code for '***doTextLowerCase***' is shown below.

A continuación se detallan las formas en que podemos definir las extensiones de usuario en Selenium IDE.
1) Acción:
todas las acciones, comenzamos por "do", es decir, si la acción es crear una extensión para el texto en minúsculas, su nombre hará ***TextLowerCase*** . Después de agregar esta función a Selenium IDE, creará automáticamente una función de espera para esta acción. Por tanto, en este caso tendremos dos métodos conocidos como ' ***doTextLowerCase*** ' y ' ***doTextLowerCaseAndWait*** '.

```
Selenium.prototype.doTextLowerCase = function(locator, text) {
 //Here findElement is handling all type of locators i.e. by CSS, XPAth, NAME, ID, etc.
 var element = this.page().findElement(locator);

 // Create the text to Lower Case
 text = text.toLowerCase();

 // Replace the web element text to the Lower case text.
 this.page().replaceText(element, text);
 };
```

Explicación del código de JavaScript:

- Aquí, estamos llamando al objeto prototipo Selenium para crear una nueva acción 'doTextLowerCase', que es un método que acept parámetros, a saber, el *localizador* y el *texto* .
- El objeto localizador puede contener el nombre del elemento web, ID, XPath, CSS, etiqueta, etc. para ubicar el elemento web utiliza método findElement del objeto de la página JavaScript como se muestra arriba.
- Texto como argumento, estamos convirtiendo a minúsculas con el método de JavaScript.
- Por último, estamos utilizando el método ' *replaceText'* del objeto de página para reemplazar el texto de la página de los elementos seleccionados en minúsculas.

2) Evaluadores / Afirmación:

Podemos crear nuestra propia función personalizada para los usuarios de Selenium IDE. Todos los accesores en Selenium IDE están regi con *prototipo de objeto de selenio* que siempre irá precedido de "get" o "is", por ejemplo, getValueFromCompoundTable, isValueFromCompoundTable, etc. Puede aceptar dos parámetros. El primer parámetro es el objetivo y el segundo parámetro es el campo valor en el caso de prueba. Vamos debajo de esto con la ayuda de seguir el código JavaScript.

```
Selenium.prototype.assertTextLowerCase = function(locator, text) {

//findElement method to locate element.
 var element = this.page().findElement(locator);

// To verify text convert to Lower case
 text = text.toLowerCase();

// To get actual element value
 var actualValue = element.value;

// Make sure the actual value matches the expected
 Assert.matches(expectedValue, actualValue);
 };
 Selenium.prototype.isTextASEqual = function(locator, text) {
 return this.getText(locator).value===text;
 };

 Selenium.prototype.getTexAstValue = function(locator, text) {
 return this.getText(locator).value;
 };
```

Explicación del código de JavaScript:

- Aquí, estamos llamando al objeto prototipo Selenium para crear como nuevo acceso ' *assertTextLowerCase',* que es un método que acepta dos parámetros: *localizador* y *texto* .
- El objeto localizador puede contener el nombre del elemento web, ID, XPath, CSS, etiqueta, etc. para ubicar el elemento web utiliza método findElement del objeto de la página JavaScript como se muestra arriba.
- Texto como argumento, estamos convirtiendo a minúsculas con el método de JavaScript.
- A continuación, estamos almacenando el valor real del elemento web en el campo 'realValue'.
- Usando el objeto prototipo de Selenium ' *isTextAsEqual* ', estamos comparando el valor real del selector de elemento web con un localizador con el valor de texto más bajo. Si ambos valores son iguales, el método devolverá true else false.
- Usando el objeto prototipo de Selenium ' *getTextAsValue',* estamos devolviendo el valor de texto del elemento web ubicado por el m findElement del objeto de página.

3) Estrategia de localización:

Podemos crear nuestra propia función personalizada para ubicar un elemento extendiendo el ***prototipo de PageBot*** con una función que prefijada como " *locateElementBy* ". Esta función aceptará dos parámetros. El primer parámetro es la cadena de localización ' *text'* y el seg parámetro es el documento ' *inDocument'* donde buscará esta cadena de localización como se muestra a continuación. A continuación se muestra el ejemplo del Localizador de texto en minúsculas.

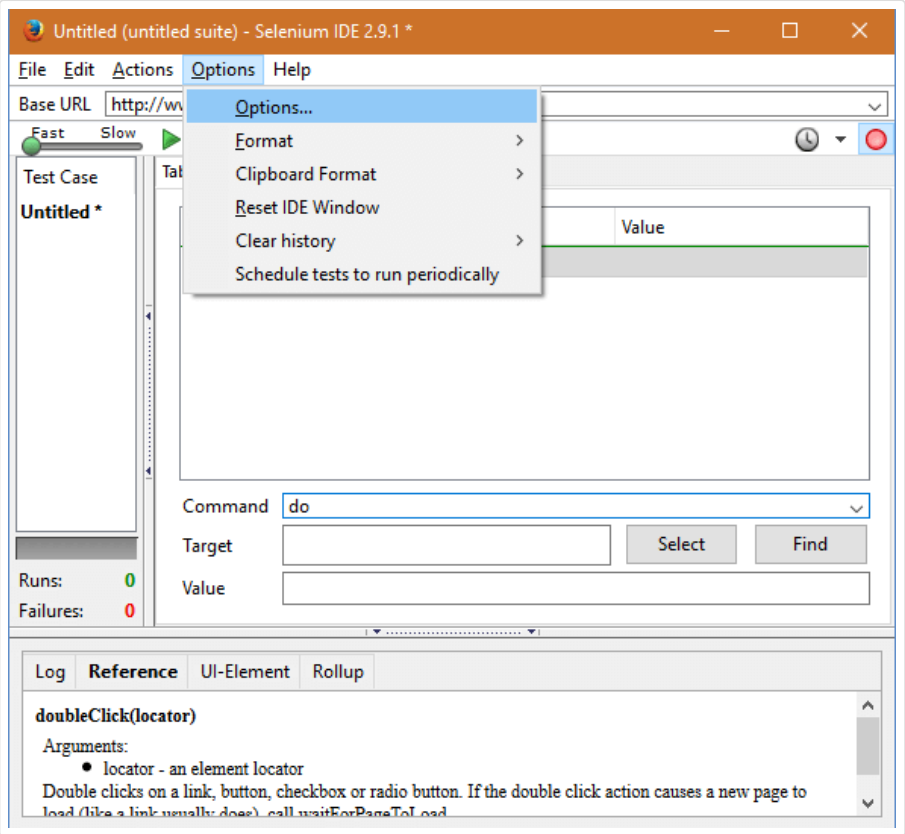
```
01 // The "inDocument" is a document you are searching.
02  PageBot.prototype.locateElementByLowerCase = function(text, inDocument) {
03
04 // Create the text to search for
05  var expectedValue = text.toLowerCase();
06
07  // Loop through all elements, looking for ones that have
08  // a value === our expected value
09  var allElements = inDocument.getElementsByTagName("*");
10
11 // This star '*' is a kind of regular expression it will go through every element (in HTML DOM every elem
   surely have a tag name like<body>,<a>,<h1>,<table>,<tr>,<td> etc. ). Here our motive is to find an elemer
   which matched with the Upper Case text we have passed so we will search it with all elements and when we
   match we will have the correct web element.
12  for (var i = 0; i < allElements.length; i++) {
13  var testElement = allElements[i];
14  if (testElement.innerHTML &amp;amp; testElement.innerHTML === expectedValue) {
15  return testElement;
16  }
17  }
18  return null;
19  };
```


Hemos guardado el código JavaScript anterior en un archivo ' *CustomUserExtension.js'* en la ubicación ' *C:* \ *selenium_demo* \' en la máqu local. Agregaremos este archivo a Selenium IDE como extensión de usuario siguiendo los pasos.

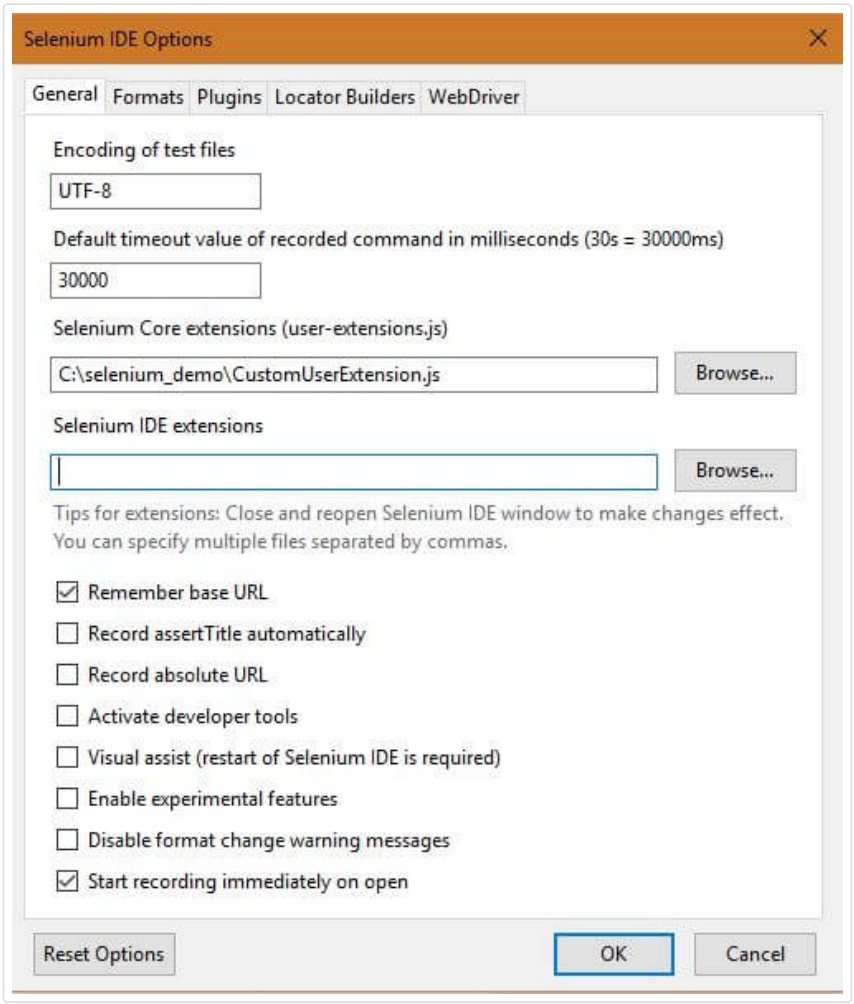
Pasos para agregar extensión al IDE de selenio:

Paso 1: abra el navegador Firefox donde el complemento IDE de Selenium ya está instalado. Abra Selenium IDE como una ventana emer

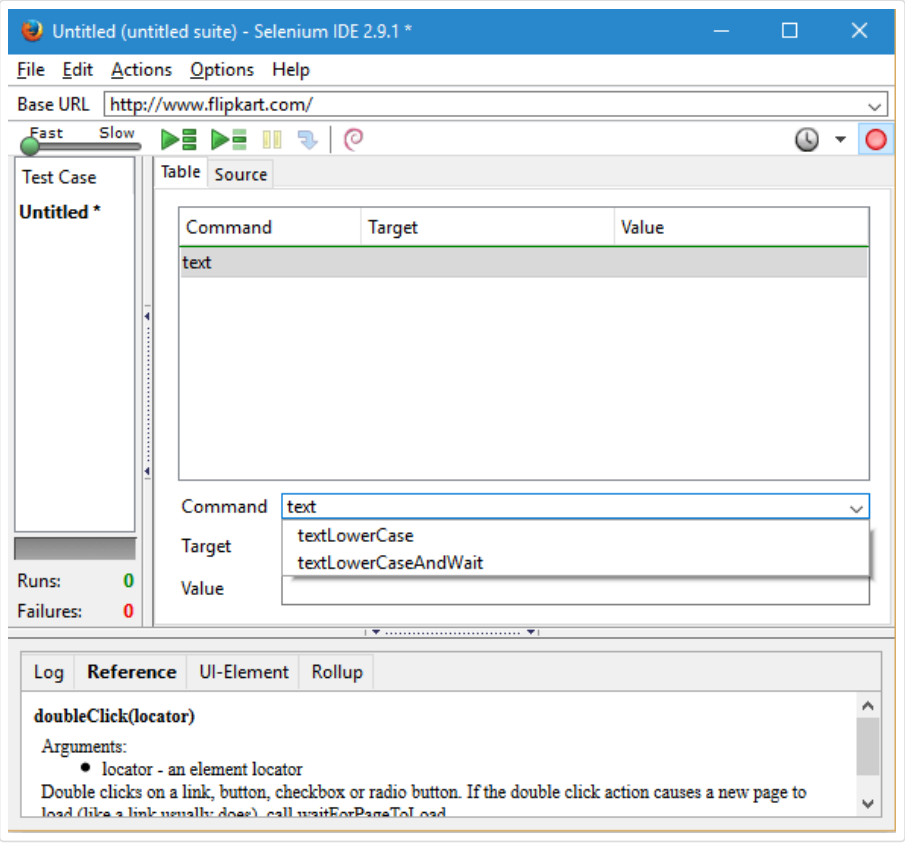
Paso 2: Navegue al menú superior del IDE de Selenio como Opciones a Opciones ... como se muestra a continuación.



Paso 3: Esto abrirá un cuadro de diálogo pidiendo que proporcione la ruta local del archivo **Selenium Core Extensions** (user-extensions. Proporcione la ruta como ubicación ' *C: \ selenium_demo \ CustomUserExtension.js' y* haga clic en el botón Aceptar como se muestra a continuación.



Paso 4: la extensión de usuario de selenio se ha agregado correctamente. Podemos, por ejemplo, la extensión en Selenium ID después d búsqueda del método en el campo 'Command' de IDE de selenio, como se muestra a continuación. Podemos observar que los nuevos mé y " que creamos como extensión de usuario están presentes en Selenium IDE.



A ti:

En el tutorial ***Selenium Core Extensions*** , hemos aprendido acerca de la creación de extensiones definidas por el usuario de Selenium q son directamente compatibles con la biblioteca IDE de Selenium. Hemos creado una extensión de acción, accesos o aserciones y estrateg localización utilizando el prototipo de objeto Selenium y el prototipo de objeto PageBot en JavaScript. Usando el mismo enfoque, podemos más extensiones definidas por el usuario según la necesidad de prueba de la aplicación bajo prueba por Selenium IDE.

Si usted no es un lector habitual de este sitio web, le recomendamos encarecidamente que se registre para recibir nuestro boletín informat gratuito por correo electrónico. Regístrese simplemente proporcionando su dirección de correo electrónico a continuación:

Ingrese su dirección de correo electrónico:

Suscribir

Revise el correo electrónico en su bandeja de entrada para obtener confirmación de las últimas actualizaciones de Software Testing de t gratuita.

Pruebas felices!

Artículos Relacionados:

1. **Selenium Training Series - Introducción al Selenium IDE**
2. **Diferencia entre el selenio IDE, RC y WebDriver**
3. **¿Cómo crear Selenium WebDriver Test usando Selenium IDE? - Tutorial de Selenio**
4. **Tutorial de selenio: Introducción a la red de selenio**
5. **Cómo manejar las cookies en Selenium WebDriver**
6. **Cómo acceder a enlaces y tablas con Selenium Webdriver?**
7. **Instalación de TestNG en Eclipse - Tutorial Selenium WebDriver**
8. **Inicie Selenium WebDriver Test Script por su cuenta - Tutorial de Selenio**
9. **Cómo ejecutar su primera secuencia de comandos Selenium WebDriver - Selenium WebDriver Tutorial**
10. **Marco JUnit e híbrido (Keyword + Data Based) - Parte 1**

19 de marzo de 2018 | Etiquetas: Acción , Afirmación , Evaluadores , Estrategia de localización , Selenium ide , Creación de extensión de usuario | Cate
Pruebas de automatización, Selenio