Miscellaneous

27–28 Apr., 4–5 May 2019

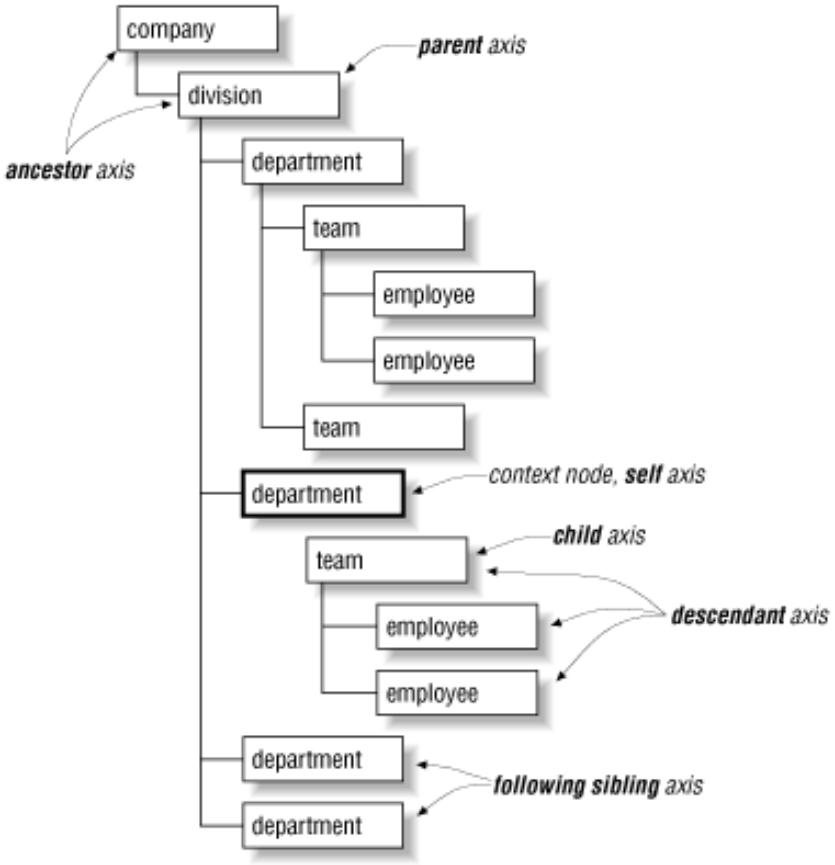
@IT MJU

Assoc. Rangsit Sirirangsi

www.indythaitester.com

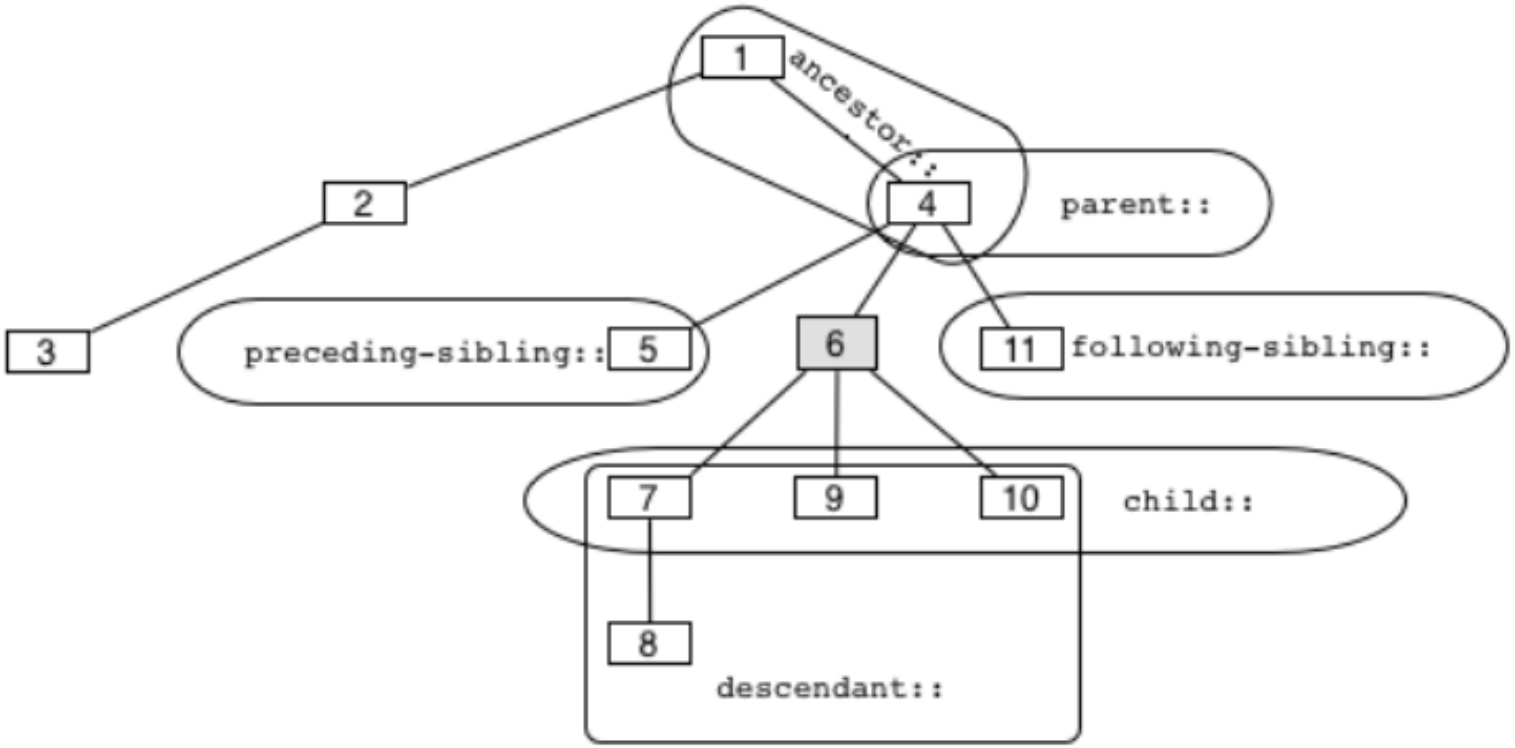
Xpath Axes

- เนื่องจากปัญหาในการระบุตำแหน่งด้วย Absolute Xpath จึงได้มีการพัฒนาการใช้ Axes ดังต่อไปนี้
  - **ancestor**
  - **descendant**
  - **following-sibling**
  - **preceding-sibling**
  - **child**
  - **parent**



Selenium : Xpath Axes

- รูปแบบของ Xpath Axes มีดังต่อไปนี้
  - //*[@id='day']/following-sibling::*



Neighbor XPath

- เป็น Locator ที่ถูกพัฒนาเพื่อใช้งานร่วมกับ Katalon Studio โดยมีรูปแบบดังนี้

1	2	3	4	5
xpath=(.//*[normalize-space(text()) and normalize-space(.)='neighbor_text'])	[n]/	preceding:: following::	tag_name	[m]

- ต้องใช้กับ neighbor ของ element เป้าหมายที่ชัดเจนและถูกระบุโดย text เป็นหลัก โดยมีหลักการทำงานดังนี้

Column No.	Discrete description
1	Retrieve the element using the text of its neighbor
2	Retrieve the neighbor among those retrieved in step 1 using the neighbor's index.
3	From the neighbor, going forward or backward
4	Retrieve all elements with the same tag name as the target element, then
5	Retrieve the target element

Send Keys : Simulate Keyboard Event

- คำสั่ง sendKeys() นอกจากจะใช้ในการ input ข้อมูลลงใน Text Box แล้วยังสามารถใช้ในการจำลองการทำงานของคีย์ต่าง ๆ ได้โดยมีรูปแบบดังนี้

static void sendKeys(TestObject **to,** String **strKeys)**

Param	Param Type	Mandatory	Description
to	TestObject	Required	Represent a web element.
strKeys	String	Required	The combination of keys to type.

WebUI.*sendKeys(findTestObject('textArea'), Keys.chord(Keys.**LEFT_CONTROL, 'a'))*

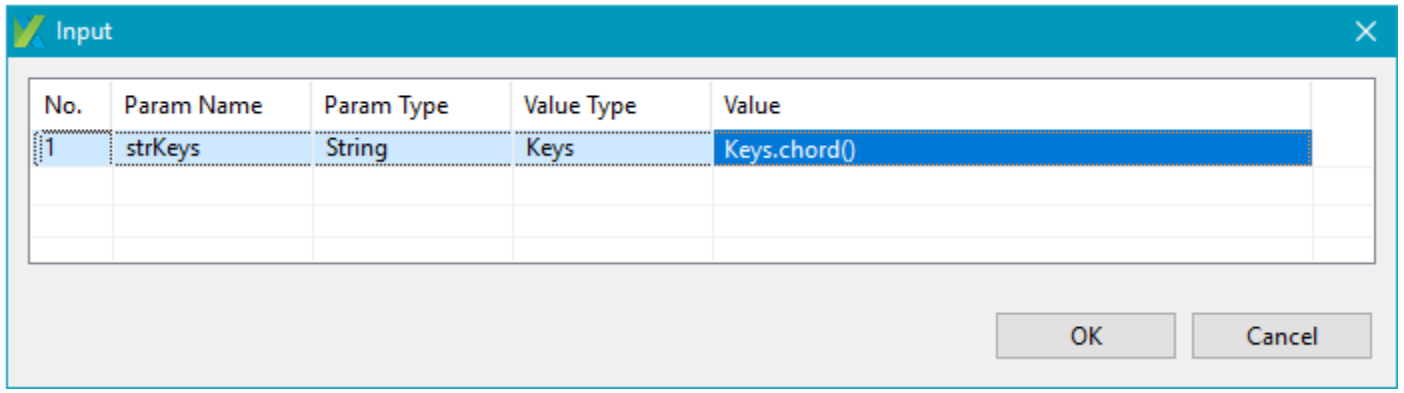
WebUI.*sendKeys(findTestObject('textArea'), Keys.chord(Keys.**LEFT_CONTROL, Keys.DELETE))*

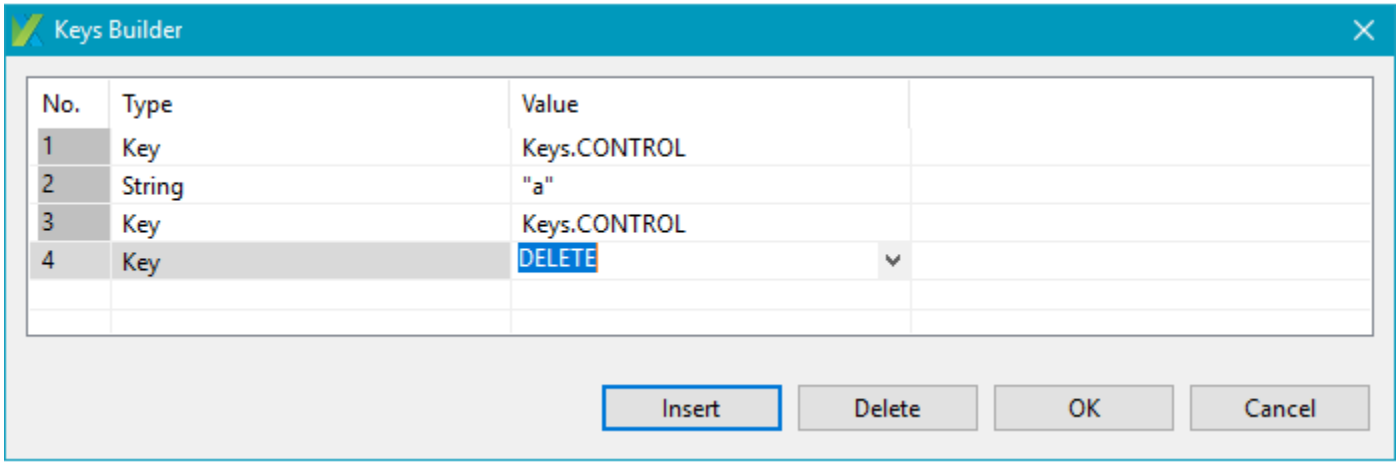
Keys.chord : Parameters

Keyboard's Key	Keys enum's value
Backspace	Keys.BACK_SPACE
Ctrl Key	Keys.CONTROL
Alt key	Keys.ALT
DELETE	Keys.DELETE
Enter Key	Keys.ENTER
Shift Key	Keys.SHIFT
Spacebar	Keys.SPACE
Tab Key	Keys.TAB
Equals Key	Keys.EQUALS
Esc Key	Keys.ESCAPE
Home Key	Keys.HOME
Insert Key	Keys.INSERT
PgUp Key	Keys.PAGE_UP
PgDn Key	Keys.PAGE_DOWN

Select Keys.chord : Parameters

- Manual Mode ในส่วนของ input ดับเบิลคลิกเพื่อเลือก Input และ Keys ดังรูป



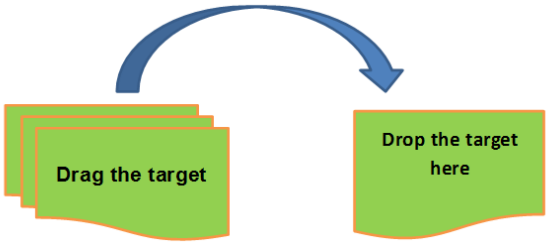


dragAndDropToObject()

- โดยปกติแล้วเว็บแอพ ฯ ประกอบไปด้วยคุณสมบัติที่เรียกว่า Drag and Drop
- Katalon Studio สนับสนุนการทำงานดังกล่าวโดยการเรียกใช้คำสั่งสองรูปแบบ
- แบบแรกเป็นการลาก TestObject หนึ่งไปวางยังอีก TestObject หนึ่ง โดยมีรูปแบบดังนี้

static void dragAndDropToObject(TestObject sourceObject, TestObject destinationObject)

  - พารามิเตอร์ประกอบด้วย TestObject ที่ต้องการ Drag และ Drop ตามลำดับ



Demo : Drag & Drop

- สร้าง Test Script ที่กำหนดโดยใช้ ChroPath ช่วยในการเขียนสคริปต์
- เพื่อคลิกปุ่มโดยใช้คุณสมบัติ Drag and Drop จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	https://demos.telerik.com/kendo-ui/dragdrop/index
3	Get Text and display in comment	to (Before text)
4	drag and drop	from => to
5	Wait for…	?
6	Get Text and display in comment	to (After text)
7	Close Browser	close

DragAndDropTo

Verify Element Present

- ตรวจสอบว่าส่วนประกอบของ HTML ถูกนำเสนอภายใน DOM หรือไม่ คำสั่งหรือเมธอดนี้คืนค่า true ในกรณีที่เป็นจริงหรือ false ในกรณีที่เป็นเท็จ

static boolean verifyElementPresent(TestObject to, int timeOut, FailureHandling flowControl)

actualText	String	Required	Represent the actual text.
to	TestObject	Required	Represent a web element.
timeout	int	Required	System will wait at most timeout (seconds) to return result

dragAndDropByOffset()

- Katalon ใช้คำสั่งในรูปของเมธอดนี้เพื่อลาก TestObject และวางลงในตำแหน่ง offset ที่ถูกระบุ

static void dragAndDropByOffset(TestObject sourceObject, int xOffset, int yOffset)

- พารามิเตอร์ "SourceObject" เป็น TestObject ที่ต้องการ drag
- พารามิเตอร์ตัวที่ 2 และ 3 เป็นตำแหน่งแกน x และ y ในรูปของ pixel ของ element ปลายทางที่ต้องการ drop element ต้นทาง first element.

Switch Windows

- ในเว็บแอพพลิเคชั่นที่ประกอบไปด้วยหลาย ๆ เฟรมหรือหลายวินโดวส์ เมื่อสลับการทำงานไปยังวินโดวส์อื่นจำเป็นต้องผ่านการควบคุมการทำงานตามไปด้วย เมื่อทำงานเสร็จสิ้นสามารถสลับกลับมายังวินโดวส์เดิมได้
- Katalon Studio ได้ออกแบบคำสั่งสำหรับการสลับวินโดวส์ไว้ 3 รูปแบบ
- รูปแบบแรกเป็นการสลับการทำงานของวินโดวส์ตามลำดับค่า index

static void switchToWindowIndex(Object **index)**

```
WebUI.switchToWindowIndex(1)
```

- พารามิเตอร์:
  - index - เป็นค่าตัวเลขที่กำหนดไว้ตามลำดับการเปิดวินโดวส์เป็นหลัก โดยเริ่มต้นจากค่า 0 เป็นลำดับแรกและ 1, 2 …

Switch Windows

- รูปแบบที่สองเป็นการสลับการทำงานของวินโดวส์ตาม title ที่กำหนดไว้

```
static void switchToWindowTitle(String title)
WebUI.switchToWindowTitle('Documentation')
```

  - พารามิเตอร์ :
    - title – ชื่อ title ของวินโดวส์ที่ต้องการสลับการทำงาน
- รูปแบบที่สามเป็นการสลับการทำงานของวินโดวส์ตาม Url ที่กำหนดไว้

```
static void switchToWindowUrl(String url)
WebUI.switchToWindowUrl('https://docs.katalon.com/')
```

  - พารามิเตอร์ :
    - url - ชื่อ url ของวินโดวส์ที่ต้องการสลับการทำงาน

Demo : Switch Windows

- สร้าง Test Script ที่กำหนดโดยใช้ ChroPath ช่วยในการเขียนสคริปต์ เพื่อแสดงการสลับหน้าจอด้วย switchToWindowIndex() จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	https://line.me/th/
3	Click Menu Game	Yes
4	Click Add Friend	First Icon
5	switch to Window index	1
6	Close Window index	1
7	Switch to Window index	0
8	Close Browser	

LineSwitchTo

What is Alert ?

- เป็น JavaScript ฟังก์ชันที่ใช้แจ้งให้ผู้ใช้ทราบในรูปของ dialog ที่ประกอบไปด้วยข้อความที่ถูกระบุ รวมถึงปุ่ม OK/Cancel
- ใช้เพื่อบังคับผู้ใช้ให้มีการกระทำที่ Dialog ก่อนทำสิ่งอื่น ทั้งนี้เพื่อป้องกันไม่ให้ผู้ใช้สามารถเข้าถึงส่วนประกอบอื่น ๆ ได้
- Alert สามารถแบ่งออกได้เป็น 3 แบบดังนี้
  - **Java Script Alert Box**: เมื่อเกิด alert ผู้ใช้ต้องคลิกปุ่ม ok เพื่อทำงานต่อ
  - **Java Script Confirm Box**: คืนค่า true เมื่อผู้ใช้คลิกปุ่ม "OK" และคืนค่า false หากคลิกปุ่ม "Cancel“
  - **Java Script Prompt Box**: หากผู้ใช้คลิกปุ่ม "OK" หลังจาก input ค่าจะคืนค่าดังกล่าวในรูปของ Output

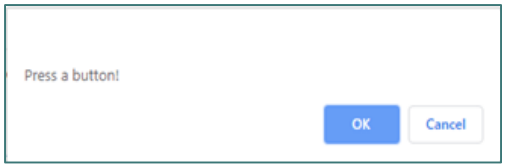
Alert : method

- Accept Alert: ใช้สำหรับควบคุมการคลิกปุ่ม Ok ภายใน Alert

  static void acceptAlert()

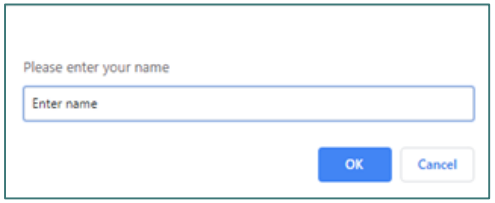
- Dismiss Alert : ใช้สำหรับควบคุมการคลิกปุ่ม Cancel ภายใน Alert

  static void dismissAlert()



- การผ่านข้อมูลยัง Prompt Alert สามารถทำได้โดยใช้คำสั่งดังนี้

  static void setAlertText()



- การอ่านข้อมูลจาก alert message : สามารถทำได้โดยเรียกใช้คำสั่งดังนี้

  static String getAlertText()

Demo : Popup Handling

- สร้าง Test Script ที่กำหนดไว้ โดยใช้ ChroPath ช่วยในการ Add Test Object Property เพื่อทดสอบการจัดการ Pop Up จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	https://www.seleniumeasy.com/test/javascript-alert-box-demo.html
3	Click me !	Yes
4	Alert Pop Up	Yes
5	Delay	4 seconds
6	Click Alert Ok button	Yes
7	Close Browser	

AlertDemo

Script Mode

- Katalon studio ยอมให้ผู้ใช้สามารถเขียนสคริปต์ได้โดยตรง โดยใช้โปรแกรมภาษา Java/Groovy ช่วยในการทำงาน
- ขั้นตอนการสร้าง Test Case ด้วยสคริปต์
  - ใช้คีย์เวิร์ด (Built-in/Custom) ในการสร้างสคริปต์
  - โดยปกติจะใช้คำสั่งประเภท WebUI เป็นหลัก
  - รัน Test Case
- ในกรณีที่ต้องการ Skip ขั้นตอนในการรันการทดสอบ
  - ใช้คำสั่ง not_run: เพื่อข้ามการทำงานของคำสั่งที่ถูกระบุ
  - ใน Manual mode ใช้การคลิกขวาที่คำสั่งเลือก => disable

Katalon : WebUiBuiltInKeywords as WebUI

- คำสั่ง Katalon ที่ถูกเรียกใช้บ่อย ๆ ใน Script Mode ซึ่งคำสั่งเหล่านี้จะขึ้นต้นด้วย WebUI ตามด้วยชื่อของเมธอดที่ต้องการเรียกใช้ เช่น

Method	Purpose
WebUI.openBrowser()	เปิดบราวเซอร์ตามชนิดที่ถูกระบุ
WebUI.navigateToUrl()	ไปที่ URL ที่ถูกระบุ
WebUI.setText()	จำลองการกรอกข้อมูลลงใน element
WebUI.sendKeys()	จำลองการกรอกข้อมูลลงใน element
WebUI.getText()	อ่านค่า text จาก Test Object ที่ถูกระบุ
WebUI.verifyTextPresent()	คืนค่า true หากผลพบว่า text ถูกนำเสนอภายในเว็บเพจ
WebUI.click()	จำลองการคลิก Test Object ที่ถูกระบุ
WebUI.closeBrowser()	ปิดการทำงานของบราวเซอร์

Demo : FindJobs

- จงสร้าง Test Script ที่กำหนดโดยใช้ ChroPath ช่วยในการ Add Property ของ Test Object ตามรายละเอียดจาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	chrome
2	Go to the Application URL	http://th.indeed.com/
3	Input what field	Selenium
4	input where field	Bangkok
5	click Find Job Button	Yes
6	get Job count	getText()
7	verify Job count	Pass/Fail
8	Close Browser	Yes

FindJob_JS

Execute JavaScript

- บางกรณี element ภายใน HTML อาจไม่ตอบสนองต่อคำสั่ง selenium เช่น คำสั่ง *click()* ดังนั้นจำเป็นต้องใช้ ***JavaScript*** เพื่อแก้ไขปัญหาดังกล่าว
- Katalon Studio ได้พัฒนาคำสั่งเพื่อใช้ร่วมกับ JavaScript โดยมีรูปแบบดังนี้

static Object **executeJavaScript(**String **script,** List **arguments,**
FailureHandling **flowControl)**

Param	Param Type	Mandatory	Description
script	String	Required	The JavaScript to execute.
argument	List	Required	The arguments to the script. Can be empty or null.

Return : Boolean, Long, Double, String, List, Web Element, or Null

JavaScriptExecutor : Cont.

- Katalon Studio เรียกใช้คำสั่งเพื่อการประมวลผล JavaScript ดังนี้
- ในกรณีที่ต้องการ click Button โดยใช้ JavaScript

```
WebElement element =
WebUiCommonHelper.findWebElement(findTestObject('your/object'),30)
WebUI.executeJavaScript("arguments[0].click()", Arrays.asList(element))
```

- ในกรณีที่ต้องการใช้แทนเมธอด sendKeys()

```
WebElement element = WebUiCommonHelper.findWebElement(findTestObject(‘xxx’), 30)
WebUI.executeJavaScript(“arguments[0].value='Chiangmai'", Arrays.asList(element))
```

- ในกรณีที่คืนค่า Element จาก id ที่ต้องการ

```
WebElement element = WebUI.executeJavaScript("return
                                document.getElementById('someId');", null)
```

Create a Dynamic Object at Runtime

- Test Objects เป็นส่วนประกอบที่สำคัญในการทดสอบอัตโนมัติ บางส่วนอาจอยู่ในรูป static (ไม่มีการเปลี่ยนแปลง) และบางส่วนอยู่ในรูป dynamic หรือในกรณีที่ต้องการลดขนาดของ Object Repository
- Katalon Studio ยอมให้ผู้ใช้สามารถสร้าง TestObject ขึ้นใหม่ในช่วงเวลารัน-ไทม์ได้ โดยการเรียกใช้เมธอดดังนี้

public TestObject **addProperty(**String **name,** ConditionType **condition,** String **value)**

  - พารามิเตอร์:
    - name - ชื่อของ property ที่ต้องการเรียกใช้ เช่น id หรือ xpath
    - condition – เงื่อนไขของ property ใหม่ เช่น equals
    - value – ค่า property ที่ต้องการกำหนดลงใน Test Object

TestObject : Add new Property

- การเพิ่ม Property ให้กับ TestObject ต้อง import Package และสร้างโค้ดดังต่อไปนี้

```
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.testobject.ConditionType
```

- เนื่องจาก TestObject ที่ถูกสร้างขึ้นจากการ addProperty() ไม่ได้เป็นส่วนหนึ่งของ Object Repository ดังนั้นจึงไม่สามารถใช้ร่วมกับคำสั่ง findTestObject() ได้ ดังการทำงานต่อไปนี้

```
String gender = 'Male'
String path = "//input[@value=‘ "+gender+" ']"
TestObject to = new TestObject().addProperty('xpath', ConditionType.EQUALS,
                                                                    path)

WebUI.click(to)
```

Demo : IFrame with Dynamic Property

- สร้าง Test Script แบบ Manual โดยอาศัย ChroPath ช่วยในการ Add Property ของ Test Object แบบไดนามิคจาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	http://jqueryui.com/selectable/
3	set var	?
4	Add Property To Test Object	yes
5	Click Item index	Yes
6	Get Text & save to	output
7	display Result	comment
8	end Loop	Yes

FrameSelected

Modify Object Property

- การแก้ไข property ของ Test Object จะถูกใช้ในกรณีที่ต้องการเปลี่ยนแปลงค่า attribute ในช่วงรันไทม์ โดยการเรียกใช้ modifyObjectProperty()
- หากการแก้ไข property ที่ถูกระบุยังไม่เคยมีอยู่จะถูกสร้างขึ้น แต่หากค่า Property ที่ถูกแก้ไขมีค่าเป็น null ค่า property เดิมจะไม่มีการเปลี่ยนแปลง
- โดยมีรูปแบบการเรียกใช้งานดังต่อไปนี้

static TestObject **modifyObjectProperty(**TestObject **testObject,** String**propertyName,** String **matchCondition,** String **modifyValue, boolean isActive)**

- **isActive** - true ในกรณีที่ property ถูกใช้ในการค้นหา test object มิฉะนั้นเป็น false

Modify Object Property

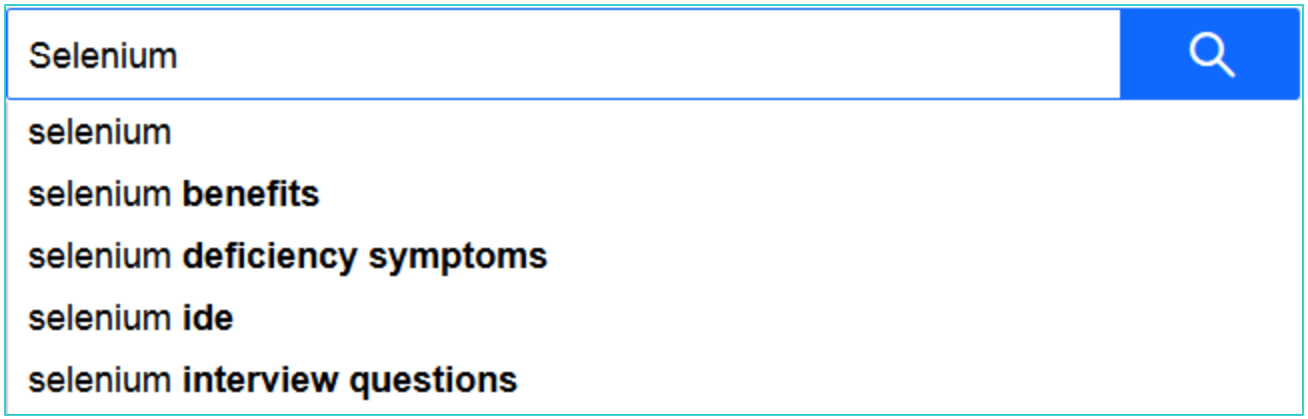
- การ Modify Object Property เป็นการแก้ไข Test Object ภายใน Object Repository ดังนั้นจึงจำเป็นต้องใช้งานร่วมกับ findTestObject() เสมอ
- หลังจากการแก้ไข property เรียบร้อยแล้วจะคืนค่า Test Object กลับไป
- ดังนั้นต้องมี Test Object ที่ถูกสร้างขึ้นใหม่เพื่อมารับค่าจากการเรียกใช้เสมอ เช่น

```
String path = "//ol[@id='selectable']/li["+var+"]“

TestObject to = WebUI.modifyObjectProperty(findTestObject('Selectable/selected’),
                                           'xpath', 'equals', path, true)
```

Dynamic Input : AutoComplete

- AutoComplete ใช้ในการคาดการล่วงหน้าของอักษร ข้อความหรือวลีที่เหลืออยู่จากการพิมพ์ของผู้ใช้ เพื่อให้เสร็จสมบูรณ์โดยอัตโนมัติ



Using Selenium commands in Katalon

- เนื่องจาก Katalon Studio ถูกสร้างขึ้นโดยใช้ Selenium เป็นหลัก ดังนั้นจึงยอมให้ผู้ใช้สามารถเรียกใช้เมธอดที่ถูกกำหนดไว้ใน WebDriver ได้ 2 วิธี
  - วิธีแรกแปลง Test Object ให้เป็น WebElement โดย import แพ็คเกจและเรียกใช้คำสั่งดังต่อไปนี้

```
import com.kms.katalon.core.webui.driver.DriverFactory
import com.kms.katalon.core.webui.common.WebUiCommonHelper as
                                                    WebUiCommonHelper
WebElement element =
WebUiBuiltInKeywords.findWebElements(findTestObject('xxx''), 5)
```

หรือ

```
WebElement element =
WebUiCommonHelper.findWebElement(findTestObject('xxx''), 5)
```

Selenium : Locating WebElement

- สามารถทำได้โดยการเรียกใช้เมธอด findElement() ที่คืนค่า WebElement
- เมธอดจะรับค่าพารามิเตอร์ร่วมกับคีย์เวิร์ด “By” ซึ่งมีรูปแบบการค้นหาดังนี้

Locator	Example (Java)
id attribute	By.id(“myElementId”)
name attribute	By.name(“myElementName”)
Tag Name	By.tagName(“td”)
Class name	By.className(“even-table-row”)
Link Text	By.linkText(“Click Me!”), By.partialLinkText(“ck M”)
CSS Selector	By.cssSelector(“h1[title]”)
Xpath	By.xpath(“//input[@id=’myElementId’]”)

Using Selenium commands in Katalon

- วิธีที่สองเรียกใช้ผ่าน DriverFactory เพื่อให้ใช้งานร่วมกับ WebDriver ได้ โดย import แพ็คเกจและเรียกใช้คำสั่งดังต่อไปนี้
- โดยการ import แพ็คเกจและเรียกใช้คำสั่งดังต่อไปนี้

```
import org.openqa.selenium.WebDriver as WebDriver
import org.openqa.selenium.WebElement as WebElement
import org.openqa.selenium.By as By

WebDriver driver = DriverFactory.getWebDriver()

WebElement tab = driver.findElement(By.cssSelector(
                                    '//table[@class="k-selectable"]/tbody[1]'))
```

Selenium Basic Command

- Selenium นำเสนอส่วนประกอบ HTML ของเว็บเพจในรูปของ WebElement
- เมธอด findElement() ใช้ค้นหาส่วนประกอบ HTML ที่ต้องการ โดยคืนค่าในรูปของ WebElement เช่น

```
WebDriver driver = new ChromeDriver();
driver.get("http://google.com");

WebElement searchBox = driver.findElement(By.name("q"));
searchBox. sendKeys("Selenium");

driver.findElement(By.name("btnG")).click();
```

เมธอด findElements()

- ใน Selenium ส่วนประกอบของ HTML จะถูกเรียกว่า WebElement ทั้งหมด
- ใช้สำหรับการค้นหาทุก ๆ ส่วนประกอบของ HTML ที่อยู่ภายในเว็บเพจปัจจุบัน ตามวิธีการค้นหาที่กำหนดไว้
- ตัวอย่าง การใช้งานของเมธอด findElements() มีดังต่อไปนี้

```
List <WebElement> list = driver.findElements(By.tagName("a"));
```

Loops: *for*

- คำสั่ง for ใช้งานในลักษณะที่มีการวนซ้ำตามจำนวนรอบที่กำหนดไว้
- ตัวอย่างเช่น Loop ใน Java

```
char[] vowels = {'a', 'e', 'i', 'o', 'u'};
    for (int i = 0; i < vowels.length; ++ i) {
      System.out.println(vowels[i]);
    }
```

- ส่วน For each Loop ใน Java มีรูปแบบดังนี้

```
char[] vowels = {'a', 'e', 'i', 'o', 'u'};

// foreach loop
for (char item: vowels) {
      System.out.println(item);
}
```

Demo : AutoComplete

- สร้าง Test Script ที่กำหนดโดยใช้ ChroPath ช่วยในการ Add Property
- เพื่อแสดงผล AutoComplete ต้องการค้นหา Selenium จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Go to the Application URL	http://www.yahoo.com
2	Input search query	Selenium
3	Get name of Autocomplete	xpath
4	Display names from Autocomplete	
5	Compare names with Test Data	seleniumhq
6	Click Test Data to search	
7	Close Browser	

yahooTestAuto