



Katalon

Miscellaneous

27–28 Apr., 4–5 May 2019

@IT MJU

Assoc. Rangsit Sirirangsi

www.indythaitester.com

Diff : Scripting and Programming Language

Scripting Language

1. ใช้กับงานขนาดเล็ก ส่วนใหญ่ใช้ร่วมกับแอปพลิเคชัน และบราวเซอร์
2. ใช้ Interpreter
3. ใช้ทรัพยากรของระบบน้อย
4. ส่วนใหญ่ใช้งานฝั่งไคลเอนต์

Programming Language

1. ใช้สำหรับพัฒนาแอปพลิเคชัน
2. ใช้ Compiler
3. ใช้ทรัพยากรของระบบจำนวนมาก
4. ส่วนใหญ่ใช้งานฝั่งเซิร์ฟเวอร์

Groovy

- Groovy เป็นภาษาสคริปต์เชิงวัตถุที่มีความสามารถในการทำงานแบบไดนามิก เช่นเดียวกับ JavaScript หรือ Python
- Groovy มีลักษณะเหมือนเป็น Superset ของ Java ซึ่งหมายความว่าจาวาโค้ดส่วนใหญ่ถือเป็น Groovy โค้ดได้เช่นกัน
- Groovy เพิ่มรูปแบบการทำงานใหม่ ๆ ลงบน Java ตัวอย่างเช่น คำสั่ง

```
System.out.println("Hello World");
```

- ถือเป็นคำสั่งที่สามารถใช้งานได้ทั้ง Java และ Groovy แต่ใน Groovy สามารถใช้คำสั่งที่สั้นกว่าและสามารถทดแทนได้โดยไม่ต้องสิ้นสุดด้วยเครื่องหมาย semi-colon เช่น

```
println "Hello World"
```

def

- การประกาศตัวแปรใน Groovy สามารถใช้คีย์เวิร์ด def ที่ย่อมาจากคำว่า definition ได้ ตัวอย่างเช่น

```
def name = "Johnny"
```

- ในทางปฏิบัติสามารถประกาศตัวแปรโดยไม่ระบุชนิดข้อมูลได้เช่นกัน เช่น

```
name = "Johnny"
```

- ในกรณีที่ต้องการอ่านค่าจากตัวแปรให้กำหนดสัญลักษณ์ '\$' ไว้ก่อนหน้าชื่อตัวแปรเสมอ ตัวอย่างเช่นต้องการแสดงข้อความ "Hello Johnny" ดังนี้

```
def name = "Johnny"  
print "Hello $name \n"
```

def

- def ใช้แทนชนิดข้อมูลได้หรืออีกนัยหนึ่งคือการใช้แทนอ็อบเจกต์
- การไม่ระบุคีย์เวิร์ด "def" ไว้ในการใช้งานร่วมกับตัวแปรสำหรับสคริปต์
ปัจจุบัน groovy จะใช้งานตามขอบเขตของตัวแปรแบบ global
- ในกรณีที่ใช้งานกับสคริปต์เดียวจะไม่มี
ความแตกต่างใด ๆ

```
def x = 1  
println x
```

```
x = new java.util.Date()  
println x
```

```
x = -3.1499392  
println x
```

```
x = false  
println x
```

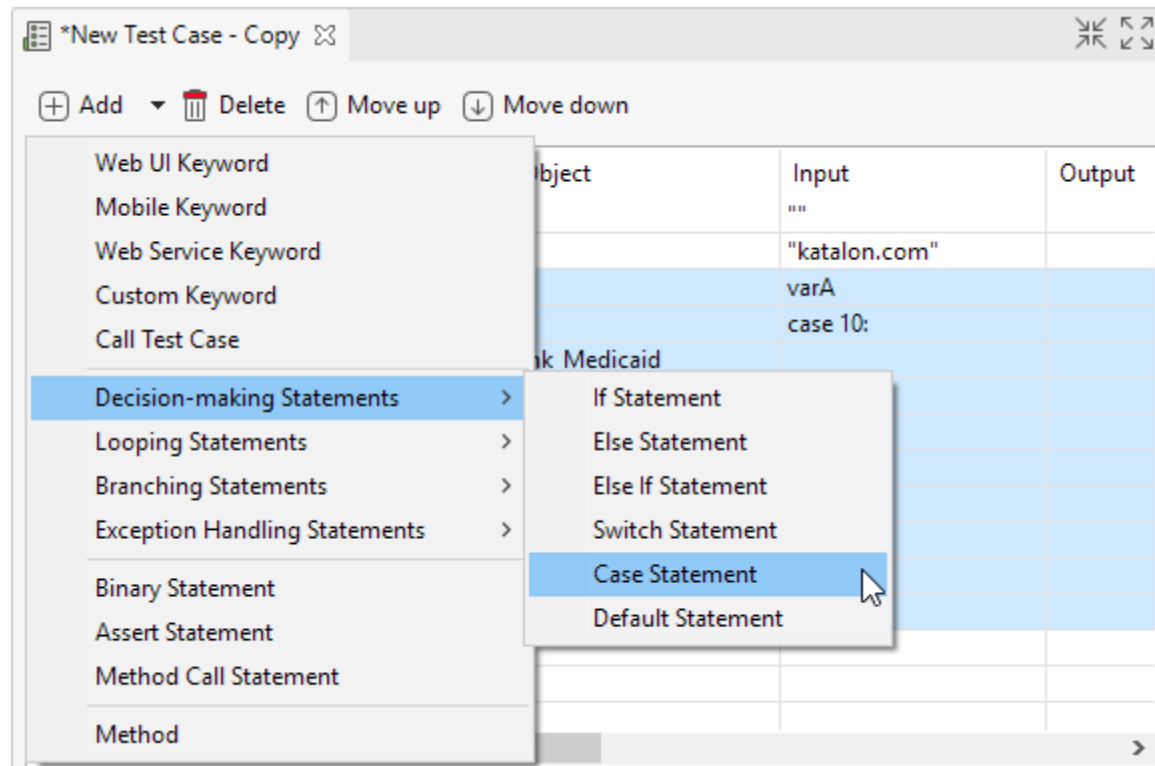
```
x = "Groovy!"  
println x
```

Statements and Relational & Logical Operators

- ใช้สำหรับการให้คอมพิวเตอร์ทำงานตามที่กำหนดไว้
- โดยปกติแล้วใช้ 1 ประโยคคำสั่งต่อ 1 บรรทัด
 - < Less than
 - <= Less than or equal to
 - > Greater than
 - >= Greater than or equal to
 - = Equal to
 - <> Not equal to
 - NOT = boolean NOT
 - AND = boolean AND
 - OR = boolean OR

Decision-making statements

- เป็นคำสั่งในการตัดสินใจหรือการทำงานแบบมีเงื่อนไขประเภทต่าง ๆ เช่น if-else, switch ซึ่งสามารถเรียกใช้ภายใน Manual view ได้ดังรูป



If..Else

STATEMENT	DESCRIPTION
If	This statement requires a boolean condition as input value. Katalon Studio will execute all the steps within once the condition is triggered.
Else If	Using Else If after If, you can create a combination of conditions where the steps within the first satisfied condition will be executed.

Item	Object	Input	Output
→× 1 - Open Browser		""	
→× 2 - Navigate To Url		"katalon.com"	
▼ IF 3 - If Statement		varA == true	
→× 3.1 - Click	chk_Medicaid		
▼ ELSE IF 4 - Else If Statement		varB == true	
→× 4.1 - Click	chk_Medicare		
▼ ELSE 5 - Else Statement			
→× 5.1 - Click	chk_None		

Print result in the log viewer

- นอกจากคำสั่งพื้นฐาน `println()` และ `comment()` ที่ใช้ในการแสดงผลผ่าน Log Viewer แล้ว katalon Studio ยังได้สร้างคลาส `KeywordLogger` เพื่อใช้งานในลักษณะเดียวกัน

```
import com.kms.katalon.core.logging.KeywordLogger
KeywordLogger log = new KeywordLogger()
log.logInfo("yourMsg")
```

- นอกจากนั้นยังสามารถใช้กับเมธอดอื่น ๆ ได้ดังนี้

```
log.logError("")
log.logFailed("")
log.logInfo("")
log.logNotRun("")
log.logPassed("")
log.logWarning("")
```

Marking tests as Passed, Failed or in Error

- Katalon Studio ทำเครื่องหมายการทดสอบในกรณีที่ failed หากคำสั่ง fails ประมวลผล แต่มีวิธีการในการ fail การทดสอบ เนื่องจาก error อื่น ๆ อาจจำเป็นต้องมีการหาคำสั่งที่ถูกต้อง แต่ error ยังคงเกิดขึ้นแล้ว
- การทำเครื่องหมายที่ผลลัพธ์การทดสอบสามารถทำได้โดยการ import
`import com.kms.katalon.core.util.KeywordUtil`
- จากนั้นผู้ใช้สามารถทำเครื่องหมายผลการทดสอบได้หลายรูปแบบ อาทิ
 - `KeywordUtil.markPassed`
 - `KeywordUtil.markFailed`
 - `KeywordUtil.markError`
 - `KeywordUtil.markWarning`

KeywordUtil

- คำสั่งในรูปแบบของเมธอดสำหรับการทำเครื่องหมายดังกล่าวนั้นจะถูกกำหนดไว้ในคลาส KeywordUtil และในแพ็คเกจ `com.kms.katalon.core.util.KeywordUtil` โดยมีรายละเอียดโดยย่อดังนี้

Method	Description
<code>logInfo(String message)</code>	Log message as info
<code>markError(String message)</code>	Mark keywords as errors
<code>markErrorAndStop(String message)</code>	Mark the keyword as an error and stop executing
<code>markFailed(String message)</code>	Mark the keyword as failed and continue execution
<code>markFailedAndStop(String message)</code>	Mark the keyword as failed and stop executing
<code>markPassed(String message)</code>	Mark keywords as passed
<code>markWarning(String message)</code>	Mark keywords as warnings

KeywordUtil.mark...

- ด้วยคำสั่งภายในคลาส KeywordUtil ส่งผลให้เกิดความยืดหยุ่นในการควบคุมการประมวลผลการทดสอบ แทนที่จะเขียนผลลัพธ์ 2 ค่าลงใน log ไฟล์ ผู้ใช้สามารถใช้ KeywordUtil ทำเครื่องหมายผลการทดสอบ passed หรือ failed ในกรณีที่สองค่าตรงกันหรือไม่

```
KeywordUtil.logInfo('Actual result is: ' + result)
```

```
if(result == expected) {
```

```
    KeywordUtil.markPassed('Pass: Actual matches Expected Result')
```

```
}
```

```
else {
```

```
    KeywordUtil.markFailed('Fail: Actual does not matches Expected Result')
```

```
}
```

Demo : if-else

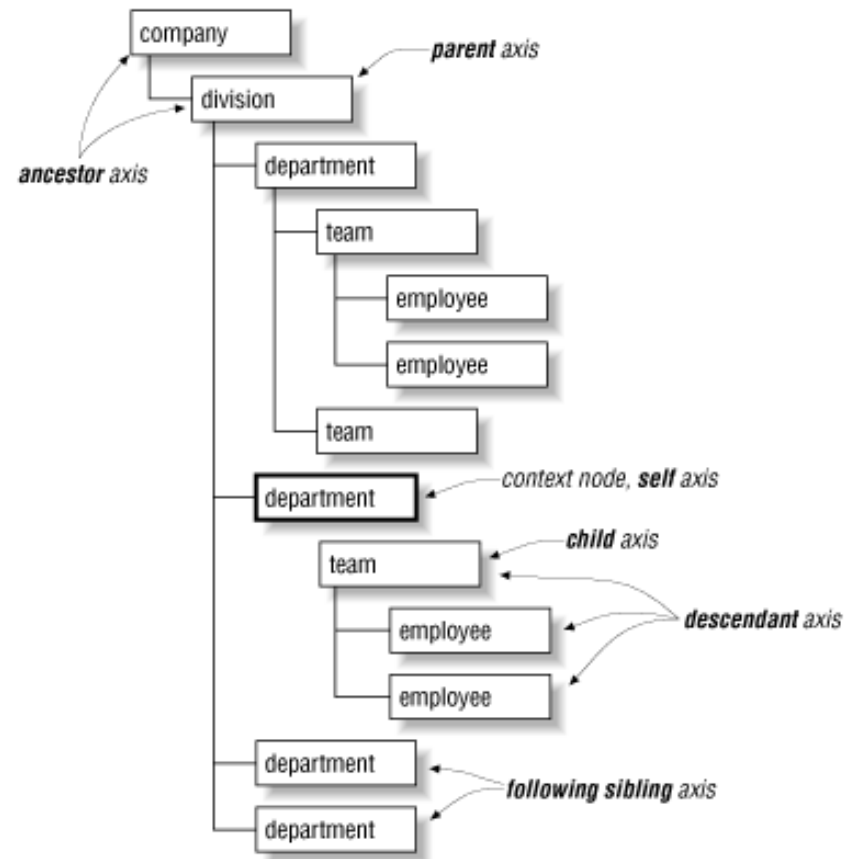
- สร้างสคริปต์สำหรับการทดสอบ โดยอาศัย ChroPath ช่วยในการหา Locator และ Add TestObject Property จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	https://percentagecalculator.net/
3	create var percent, number	30, 75
4	Enter What is	percent
5	Enter % Of	number
6	Click Calculate	Yes
7	get Attribute value save to	result
8	if result equals expected	Pass
9	else	Fail
10	Close Browser	Yes

Xpath Axes

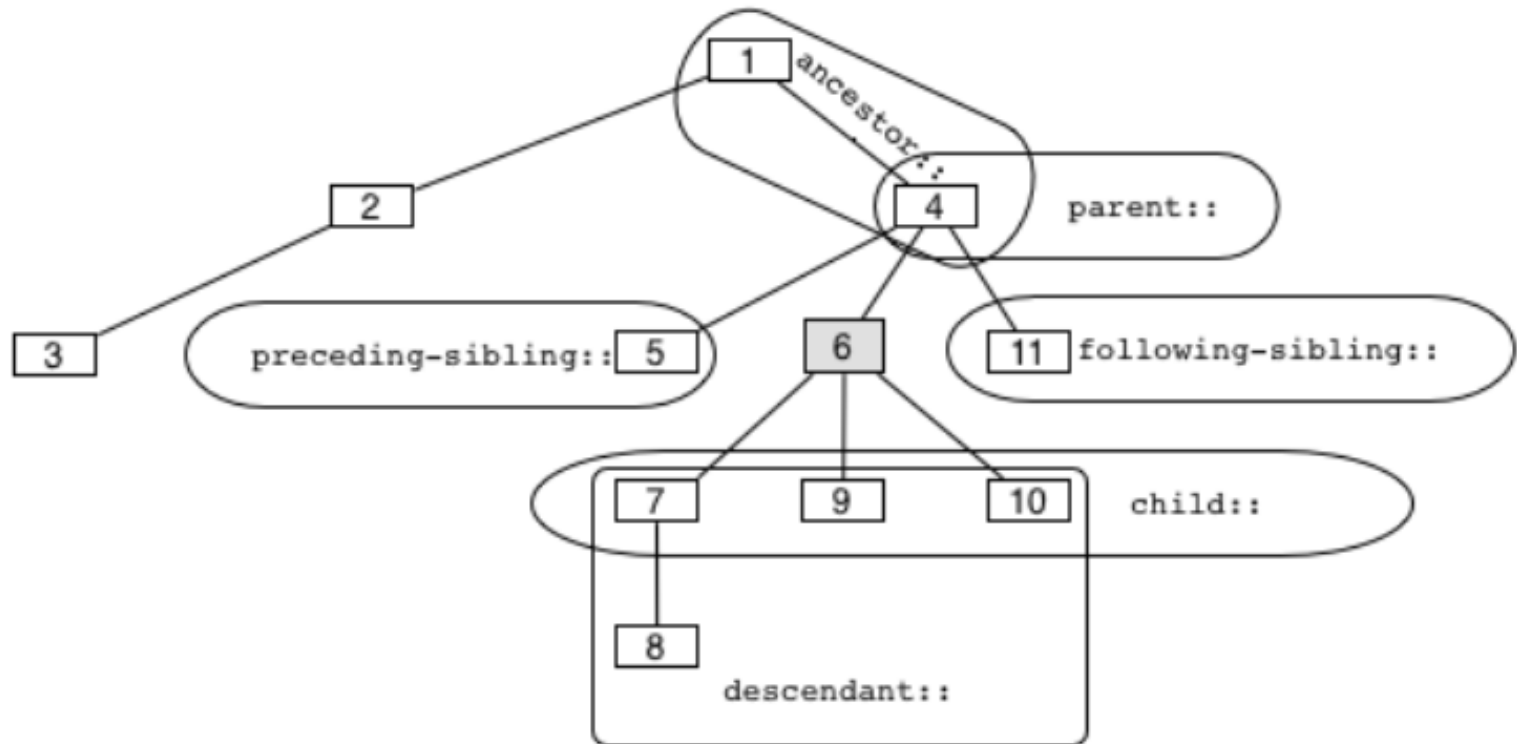
- เนื่องจากปัญหาในการระบุตำแหน่งด้วย Absolute Xpath จึงได้มีการพัฒนาการใช้ Axes ดังต่อไปนี้

- 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::	tag_name	[m]
		following::		

- 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'))`

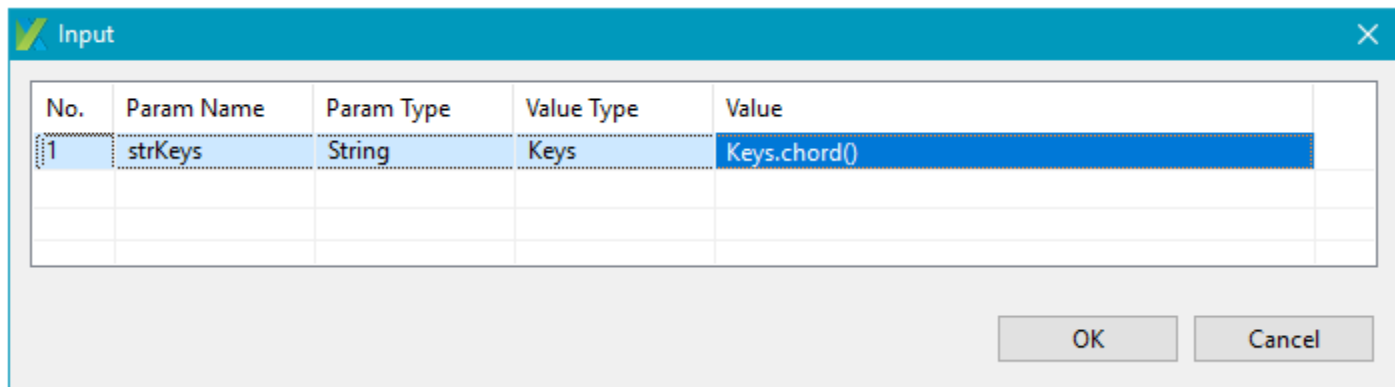
`WebUI.sendKeys(findTestObject('textArea'), Keys.chord(Keys.LEFT_CONTROL, Keys.DELETE))`

Keys.chord : Paramters

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

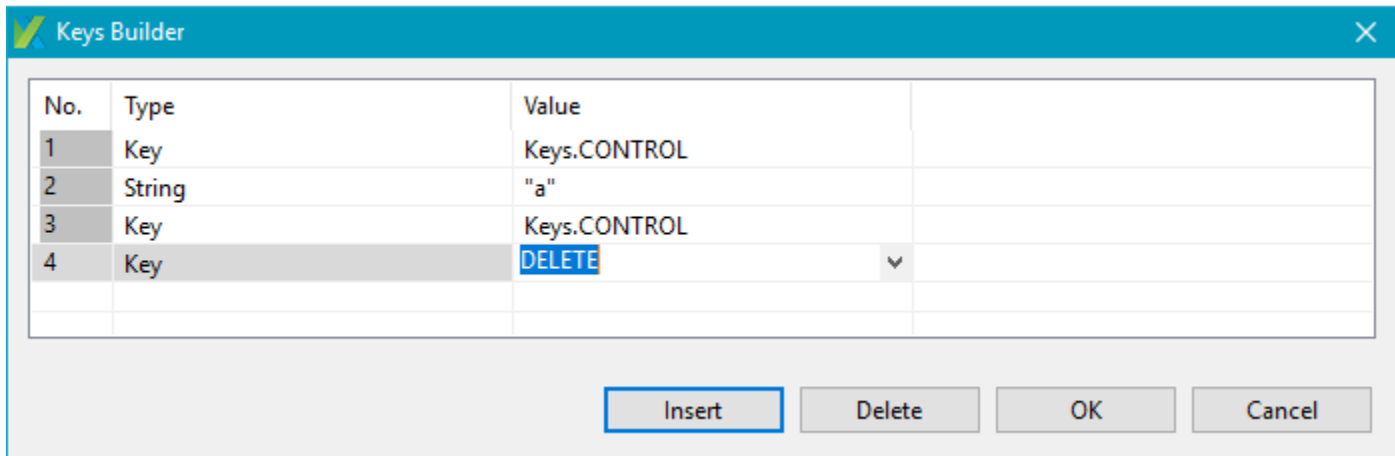
Select Keys.chord : Paramters

- Manual Mode ในส่วนของ input ดับเบิลคลิกเพื่อเลือก Input และ Keys ดังรูป



No.	Param Name	Param Type	Value Type	Value
1	strKeys	String	Keys	Keys.chord()

OK Cancel



No.	Type	Value	
1	Key	Keys.CONTROL	
2	String	"a"	
3	Key	Keys.CONTROL	
4	Key	DELETE	

Insert Delete OK Cancel