



Katalon

Data Driven Testing

27–28 Apr., 4–5 May 2019

@IT MJU

Assoc. Rangsit Sirirangsi

www.indythaitester.com

Call Test Case

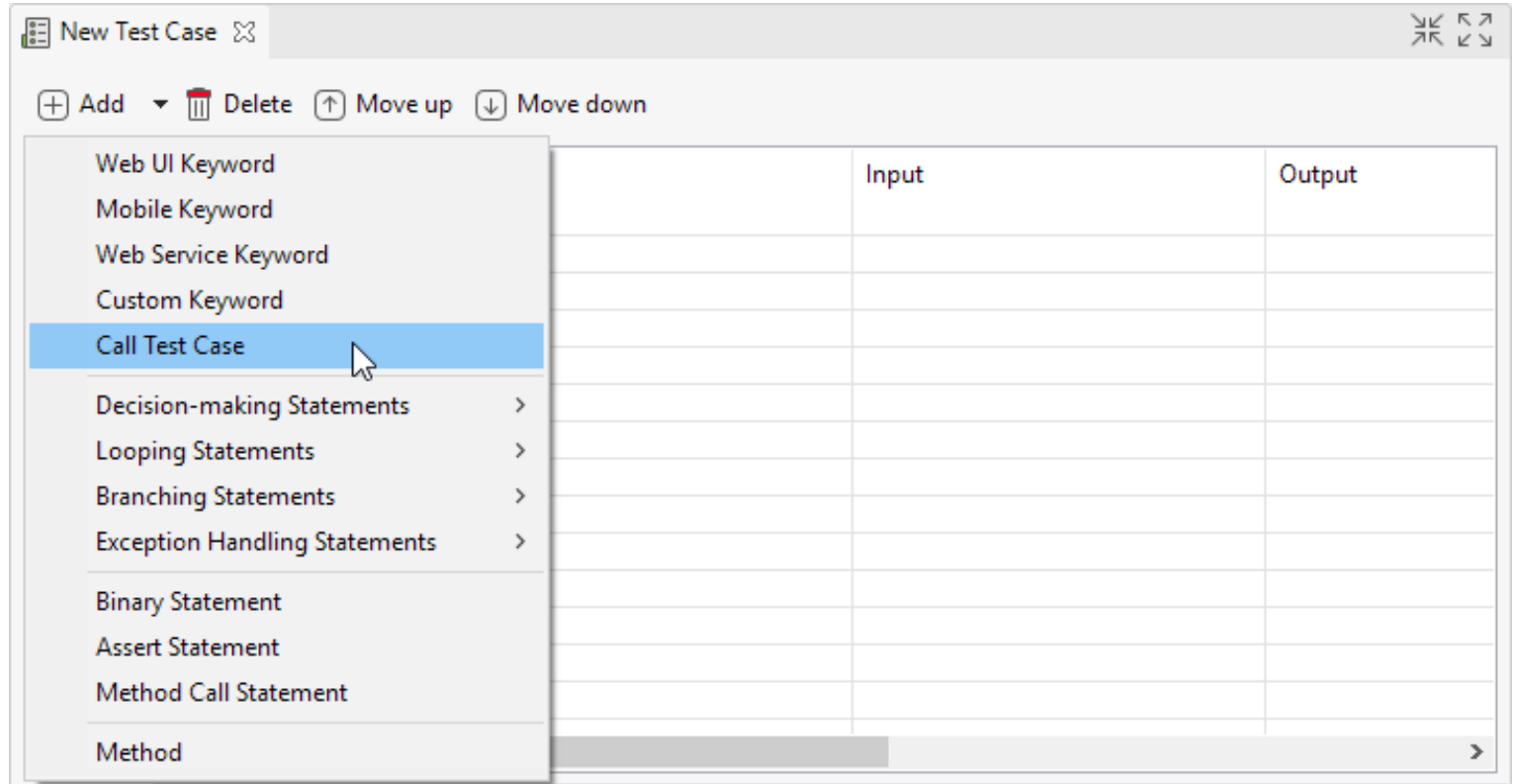
- ในกรณีที่ต้องการเรียกใช้ Test Case ที่มีอยู่เดิมเพื่อนำกลับมาใช้ใหม่ สามารถทำได้โดยการเรียกใช้คำสั่ง Call Test Case ซึ่งมีรูปแบบดังนี้

static Object callTestCase(TestCase calledTestCase, Map binding)

Param	Param Type	Mandatory	Description
arg0	TestCase	Required	Represent the called test case's path.
arg1	Map<String,Object>	Required	Represent the list of parameters will be used in the called test case.

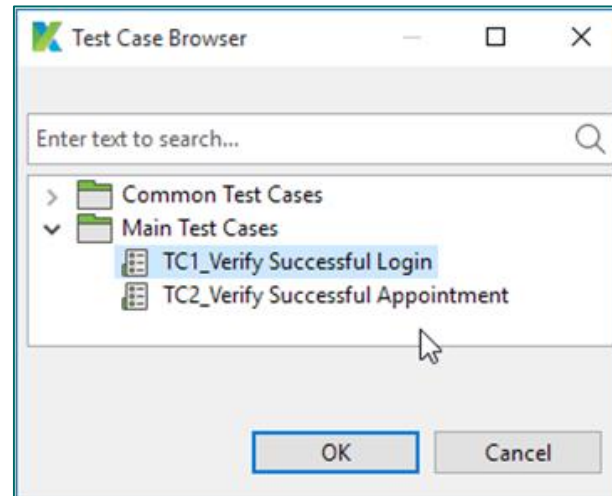
Call Test Case in Manual view

- การเรียกใช้ Test Case อื่นที่อยู่ภายใน **Manual view**:
 - เปิด test case ใน **Manual view** จากนั้นเลือกเมนูย่อย **Call Test Case** ดังรูป



Call Test Case in Manual view

- **Test Case Browser** จะแสดงชื่อ Test Case เพื่อให้ผู้ใช้เลือกได้ตามต้องการ



- **Call Test Case** จะเพิ่ม test case ที่ถูกเลือกไว้ดังรูป

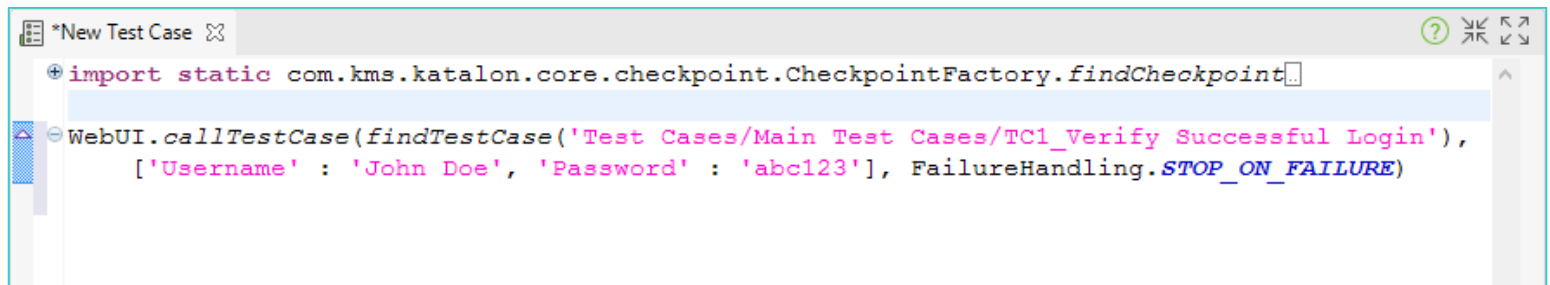
*New Test Case			
Add Delete Move up Move down			
Item	Object	Input	Output
1 - Call Test Case	TC1_Verify Successful Login	["Username":'John Doe', "Password":	

Call Test Case in Scripting view

- ส่วนภายใน **Scripting view** การเรียกใช้ Test Case อื่นสามารถทำได้โดยการเรียกใช้เมธอด *callTestCase* ซึ่งยอมให้ผู้ใช้กำหนดชื่อและพารามิเตอร์ที่จำเป็น ดังรูปแบบการทำงานต่อไปนี้

//call test case using WebUI Class

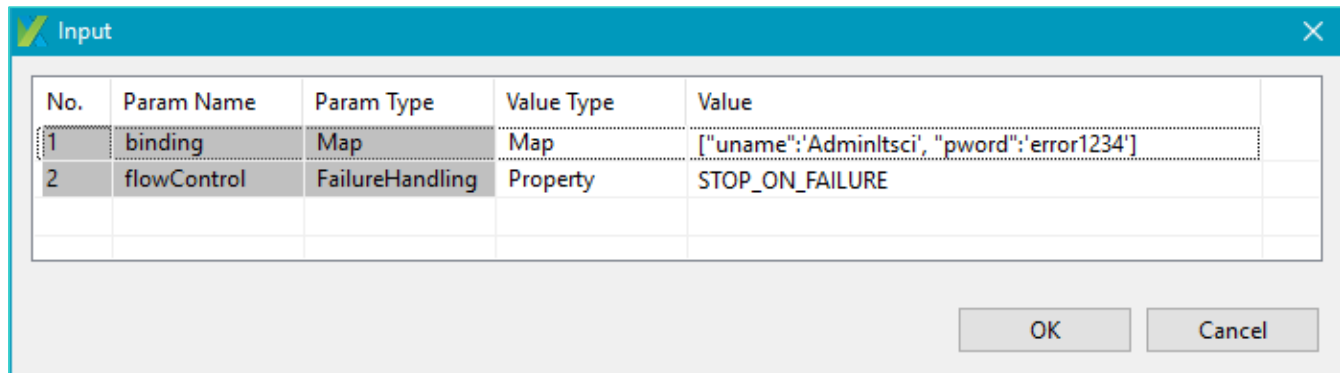
WebUI.callTestCase(findTestCase({Test Case ID}), [key1:value1, key2:value2, ... , keyN:valueN], FailureHandling.option)



```
*New Test Case
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
WebUI.callTestCase(findTestCase('Test Cases/Main Test Cases/TC1_Verify Successful Login'),
    ['Username' : 'John Doe', 'Password' : 'abc123'], FailureHandling.STOP_ON_FAILURE)
```

Call Test Case with Parameters

- การเรียกใช้ Test Case ผู้ใช้สามารถผ่านพารามิเตอร์ที่ต้องการได้

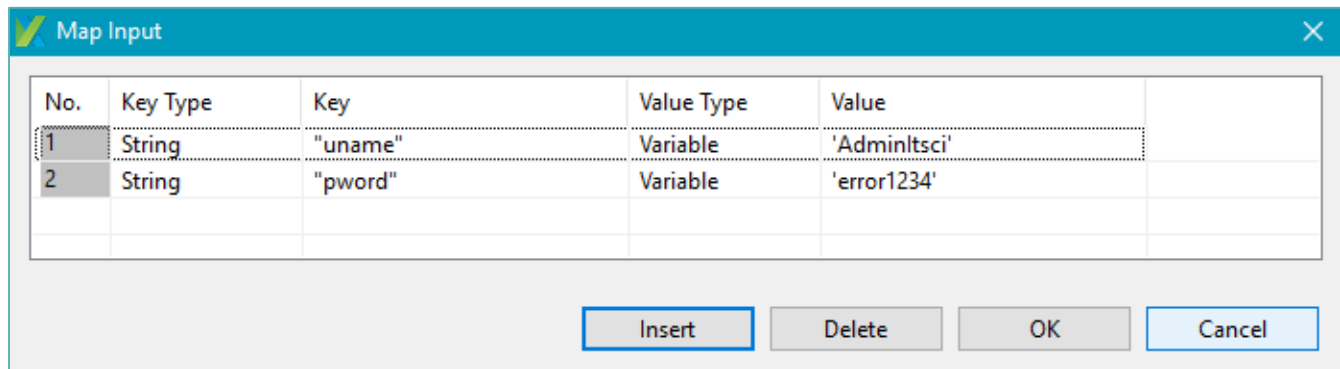


The 'Input' dialog box contains a table with the following data:

No.	Param Name	Param Type	Value Type	Value
1	binding	Map	Map	{"uname":'Adminltscl', "pword":'error1234'}
2	flowControl	FailureHandling	Property	STOP_ON_FAILURE

Buttons: OK, Cancel

- โดยกำหนดค่าในรูปของ Map<String,Object> ตามลำดับค่าที่ต้องการเรียกใช้










The 'Map Input' dialog box contains a table with the following data:

No.	Key Type	Key	Value Type	Value
1	String	"uname"	Variable	'Adminltscl'
2	String	"pword"	Variable	'error1234'

Buttons: Insert, Delete, OK, Cancel

Call Test Case with Parameters

- ตัวอย่างการเรียกใช้ Test Case อื่นภายใน Test Case ปัจจุบัน เพื่อจุดประสงค์ในการลดจำนวนโค้ด โดยการนำ Test Case ที่มีอยู่แล้วกลับมาใช้งานใหม่

+ Add Recent keywords Delete Move up Move down Edit tags		
Item	Object	Input
 1 - Call Test Case	Login	["uname":"Somchai101", "pword":"Somchai101"]
 2 - Click	editBtn	
 3 - Set Text	input_txtname	"สมชาย ใจดีมาก"
 4 - Click	editBtn	
>  5 - If Statement		WebUI.waitForAlert(10)
 6 - Verify Match		temp; "แก้ไขข้อมูลเรียบร้อยแล้ว"; true
 7 - Close Browser		

Ex : Call Test Case

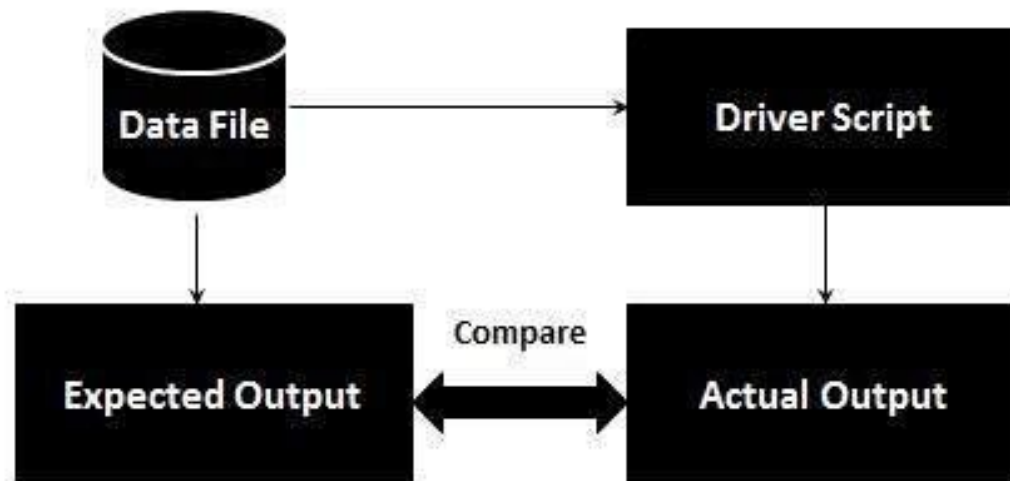
- จากโปรแกรม QtpRegisterDemo สร้าง Test Case : Login โดยกำหนดชื่อผู้ใช้งานและรหัสผ่านตามที่ได้ระบุใน FillRegisterProfile
- จากนั้นสร้าง Test Case : Delete Invalid Register โดยการทำงานจะต้องเรียกใช้ Login ก่อนเสมอ โดยการผ่านค่าพารามิเตอร์ภายใน Login ได้แก่ username : AdminItsci และ password : error1234
- กำหนด Locator ในรูปของ XPath เพื่อลบข้อมูลเฉพาะรายแรกเพียงรายเดียว
- ส่วนผลลัพธ์ของการทดสอบได้จากการเปรียบเทียบข้อมูลที่อ่านจาก PopUp (Actual Result) กับใช้ข้อความว่า ‘ลบข้อมูลเรียบร้อยแล้ว’ (Expected Result)
 - ในกรณีที่ถูกต้องแสดงข้อความ Actual Result matches Expected Result
 - ส่วนในกรณีที่ล้มเหลวแสดงข้อความ Actual Result does not match Expected Result

Limitation of Record & Playback

- มีลักษณะเป็น Hard Coded ต้องอัปเดตสคริปต์ทุกครั้งที่มีการเปลี่ยนแปลงข้อมูลเกิดขึ้น
- ไม่มีการกลไกในการจัดการกับความผิดพลาดที่อาจเกิดขึ้น บ่อยครั้งจะล้มเหลวเนื่องมาจากการ Playback เช่น Pop-Up หรือ Error Message เป็นต้น
- การเปลี่ยนแปลงสคริปต์ตามการทำงานของโปรแกรมการทดสอบจะต้องถูกบันทึกใหม่เสมอ ส่งผลทำให้ค่าใช้จ่ายในการบำรุงรักษาสคริปต์สูงขึ้น
- ในระหว่างการบันทึกหากมีข้อผิดพลาดใด ๆ เกิดขึ้นจะต้องบันทึกขั้นตอนทั้งหมดใหม่ทุกครั้ง
- สคริปต์สำหรับ Record/Playback จะทำงานได้ดีเฉพาะในกรณีที่โปรแกรมที่ถูกทดสอบได้ผ่านการทำงานมาแล้วนั่นเอง
- ไม่สามารถนำกลับมาใช้ใหม่ได้

What is DDF & DataTable

- Data Driven Framework ใช้สำหรับการทดสอบโปรแกรมด้วยข้อมูลทดสอบหลายชุด ข้อมูลดังกล่าวถูกผ่านค่าไปยังสคริปต์จากแหล่งจัดเก็บต่าง ๆ เช่น excel files, csv files, ODBC และ ADO (ActiveX Data Objects) objects




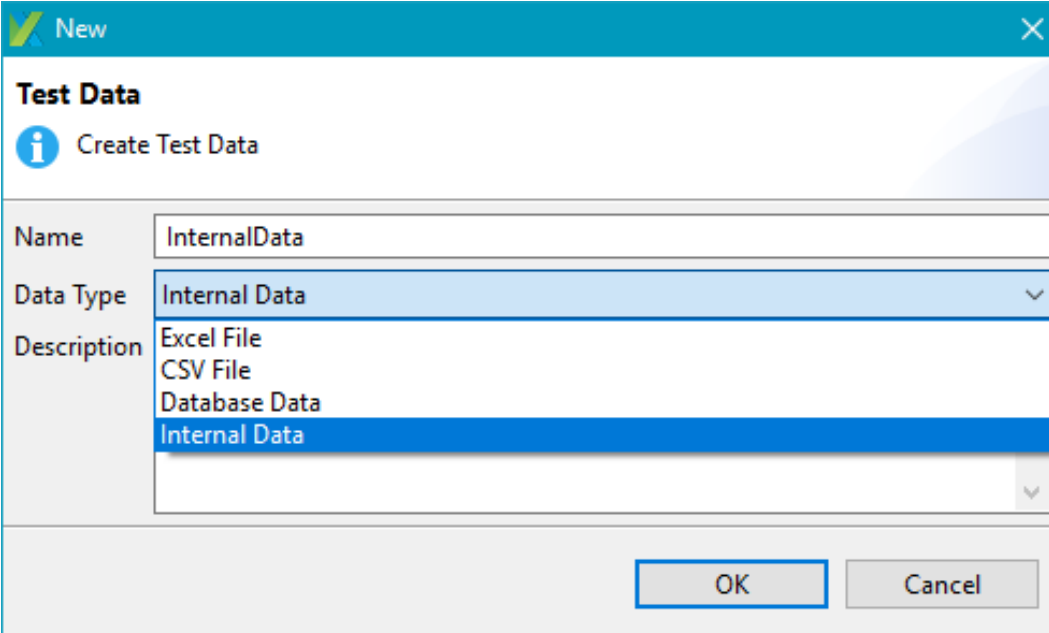
katalon : Manage Test Data

- Katalon สนับสนุนการจัดการข้อมูลสำหรับการทดสอบได้ 4 รูปแบบดังนี้
 - Internal Test Data – เป็นการสร้างชุดข้อมูลทดสอบภายใน Project ไฟล์ที่มีอยู่แล้ว
 - Excel Test Data – เป็นการสร้างข้อมูลทดสอบจากไฟล์ Excel ที่มีอยู่แล้ว
 - CSV Test Data - เป็นการสร้างข้อมูลทดสอบจากไฟล์ CSV ที่มีอยู่แล้ว
 - Database Data - เป็นการสร้างข้อมูลทดสอบจากฐานข้อมูลที่มีอยู่แล้ว




Create an Internal Data

- คลิกเลือกจาก  > New > Test Data หรือ
- File > New > Test Data จากเมนูหลัก ซึ่งจะปรากฏ New Test Data เพื่อให้ผู้ใช้ระบุชื่อและเลือก Data Type ในรูปของ Internal Data



New

Test Data

 Create Test Data

Name: InternalData

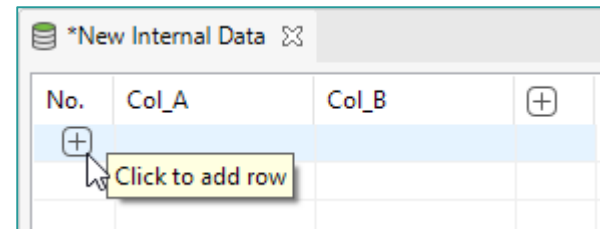
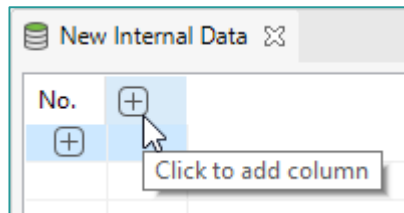
Data Type: Internal Data

Description: Excel File
CSV File
Database Data
Internal Data

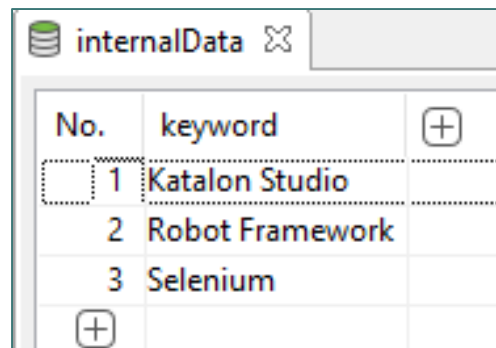
OK Cancel

Add Data to Internal Data

- ใน Editor View เลือกเครื่องหมายบวกเพื่อเพิ่มแถวและคอลัมน์ใหม่

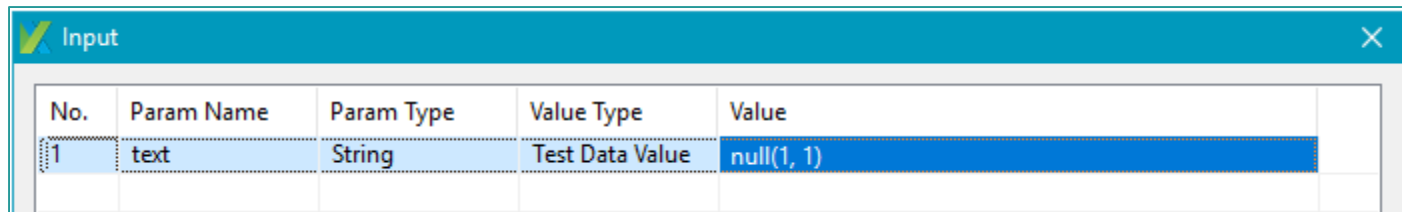


- จากนั้นเลือกแต่ละเซลล์เพื่อ input แต่ละค่าตามต้องการ



Select Data from Internal Data

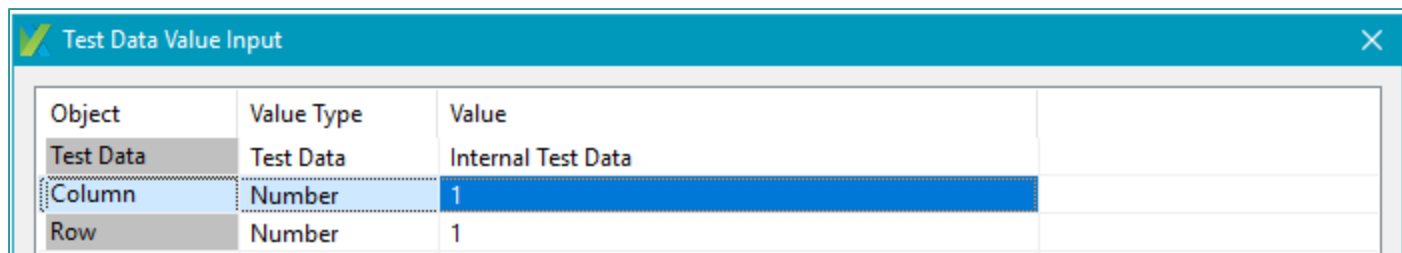
- จากโค้ดใน Manual Mode ดับเบิลคลิกในช่อง Input เพื่ออ่านค่าจาก Internal Data โดยการกำหนดค่า Input ในคอลัมน์ Value Type ให้เป็น Test Data Value



The screenshot shows a dialog box titled 'Input' with a table containing one row of test data. The table has five columns: No., Param Name, Param Type, Value Type, and Value. The first row has the following values: 1, text, String, Test Data Value, and null(1, 1).

No.	Param Name	Param Type	Value Type	Value
1	text	String	Test Data Value	null(1, 1)

- จากนั้นเลือก Value เพื่อกำหนดค่า Test Data Value Input โดยระบุชื่อ Internal Data และค่าภายในคอลัมน์และแถวตามลำดับ



The screenshot shows a dialog box titled 'Test Data Value Input' with a table containing three rows of test data. The table has three columns: Object, Value Type, and Value. The first row has the following values: Test Data, Test Data, and Internal Test Data. The second row has the following values: Column, Number, and 1. The third row has the following values: Row, Number, and 1.

Object	Value Type	Value
Test Data	Test Data	Internal Test Data
Column	Number	1
Row	Number	1

Loops: *while*

- คำสั่ง while loop มีรูปแบบและวิธีการเรียกใช้เช่นเดียวกับภาษา C หรือ Java

```
while (condition){  
    // Repeat the loop till the condition remains true.  
}
```

- ตัวอย่างเช่น

```
def x = 0  
def y = 5  
while ( y-- > 0 ) {  
    x++  
}
```

```
def singers = ["Adele", "Byonce", "Rhianna", "Elton John"]  
def i = 0  
println ("Using a while loop:")  
while (i < singers.size()) {  
    println ("${singers[i++]}")  
}
```

Loops: *for*

- คำสั่ง `for` ใน Groovy จะมีลักษณะง่ายและสะดวกต่อการใช้งาน โดยสามารถลดจำนวนการเขียนโค้ดลงแต่สามารถทำงานได้เช่นเดียวกับการใช้ Java
- ตัวอย่างเช่น Loop ใน Java

```
for(int i = 0; i < 3; i++) {  
    System.out.println("United United..." )  
}
```

- ส่วน Loop ใน Groovy ซึ่งสามารถทำงานได้เหมือนกัน

```
3.times {  
    println "United United..."  
}
```


Collection Data Types

- Groovy สนับสนุนการทำงานของ Collection โดยมีรูปแบบการประกาศตัวแปรอีอบเจกต์ดังต่อไปนี้

```
// Define a Collection called numbers holding 1, 2 and 3
def numbers = [10, 2, 33]
// Add two more entries to the Collection.
numbers.add(14)
numbers.add(57)
```

- การประกาศตัวแปร collection จะใช้ร่วมกับเครื่องหมาย [] เพื่อระบุสมาชิกที่อยู่ภายใน collection
- ส่วนการเข้าถึงสมาชิกภายใน collection สามารถดำเนินการผ่าน ***Enhanced For Loop*** (หรือเรียกว่า for-each)

Enhanced For Loop

- รูปแบบการทำงานของ Enhanced For Loop มีดังต่อไปนี้

```
for(Object in IterableObject){  
    // Set of Statements.  
}
```

- ตัวอย่างเช่น

```
def hello = "Hello"  
for(aChar in hello){  
    println aChar  
}  
  
// A Collection object holding the four seasons.  
def seasons = ["Winter", "Season", "Spring", "Autumn"]  
for(season in seasons){  
    println season  
}
```

Output

```
H  
e  
l  
l  
o  
Winter  
Season  
Spring  
Autumn
```

Loops: *for*

```
def x = 0
for ( i in 0..9 ) {
  x += i
}
```

```
x = 0
for ( i in [0, 1, 2, 3, 4] ) {
  x += i
}
def array = (0..4).toArray()
x = 0
for ( i in array ) {
  x += i
}
```

```
def map = ['abc':1, 'def':2, 'xyz':3]
x = 0
for ( e in map ) {
  x += e.value
}
```

```
x = 0
for ( v in map.values() ) {
  x += v
}
```

```
def text = "abc"
def list = []
for ( c in text ) {
  list.add(c)
}
```

Run Internal Test Data

- ภายหลังจากกำหนดค่า Internal Data เรียบร้อยแล้ว ค่า Input ที่ถูกระบุจะถูกนำเสนอผ่านโค้ดภายใน Manual Mode ดังนี้

+ Add Recent keywords Delete Move up Move down Edit tags		
Item	Object	Input
1 - For Loop Statement		(i = 1; i <= 3; (i++))
1.1 - Open Browser		""
1.2 - Navigate To Url		"https://www.google.co.th"
1.3 - Wait For Page Load		3
1.4 - Set Text	input_q	internalData("keyword", i)
1.5 - Click	firstKeyword	
1.6 - Delay		4
1.7 - Wait For Element Visi	firstLink	3
1.8 - Delay		4
1.9 - Close Browser		

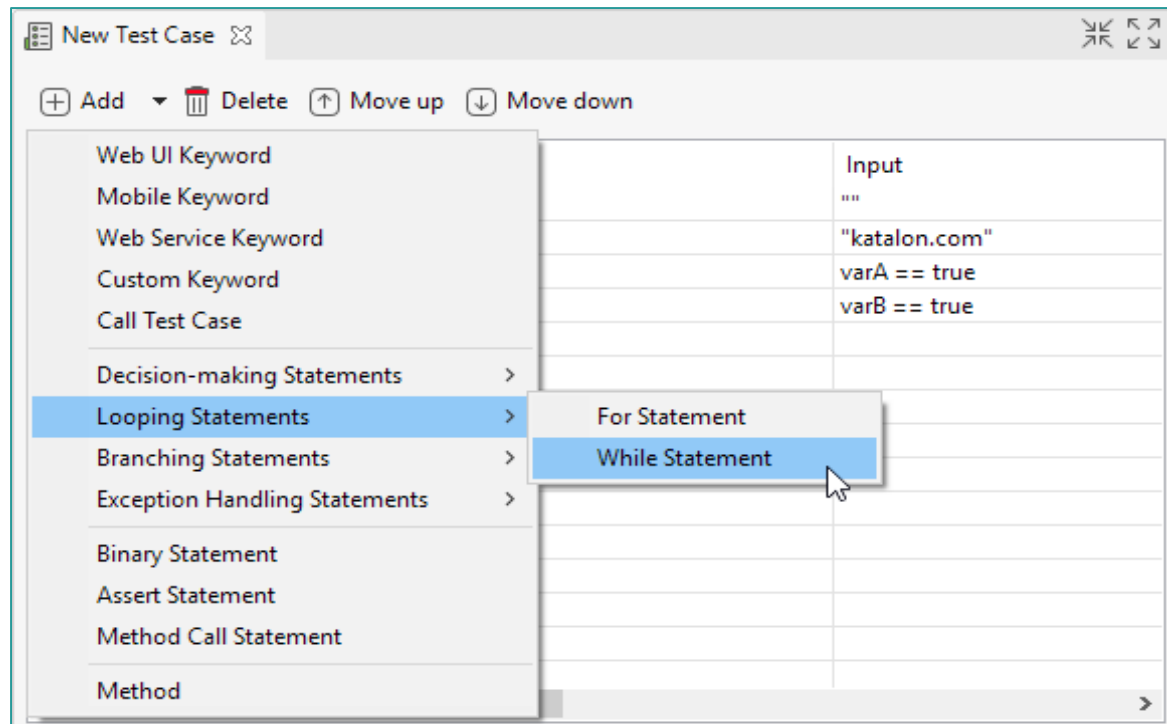
Demo : SearchSelenium

- จงสร้างสคริปต์ด้วยมือ โดยการ Add Property ของ TestObject ด้วยมือเพื่ออ่านข้อมูลจาก Internal Data โดยมีการทำงานแบบวนซ้ำชนิดต่าง ๆ เพื่อเรียกใช้ค่าในแต่ละแถวและคอลัมน์จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Open Browser	Chrome
2	Go to the Application URL	www.google.co.th
3	Search keyword	Selenium
4	Click first Keyword	Yes
5	Wait for first Link	4
6	Close Browser	Yes
7	Repeat until end	Yes

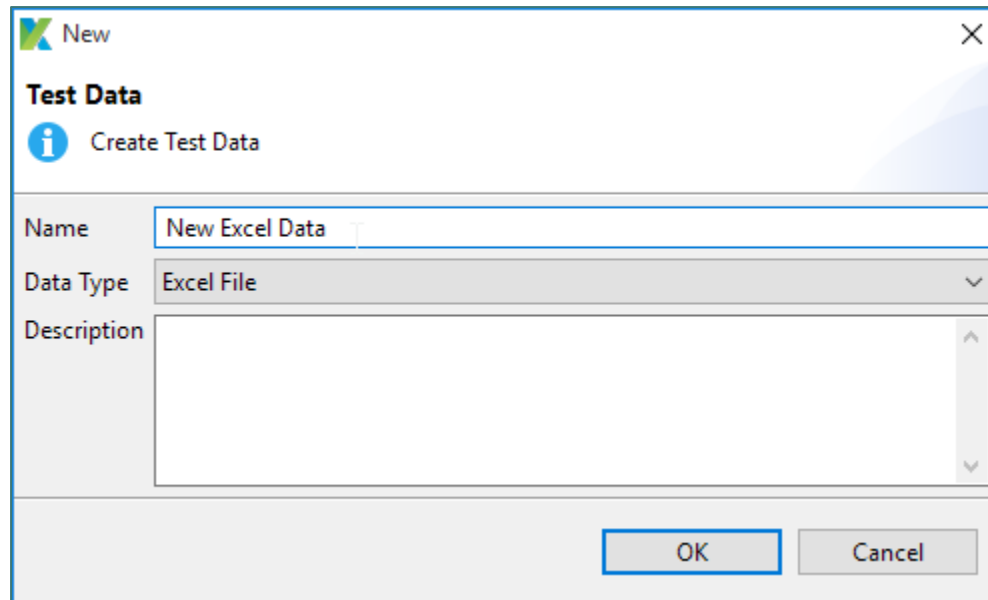
Looping statements

- กลไกควบคุมที่เหมาะสมในการทำงานร่วมกับ Data Driven Testing ได้แก่ การทำงานแบบวนซ้ำชนิดต่าง ๆ เพื่อเรียกใช้ค่าในแต่ละแถวและคอลัมน์
- ใน Katalon Studio สามารถเรียกใช้ได้จาก Manual view ดังรูป



Excel Test Data

- เช่นเดียวกับ Internal Data ขั้นตอนนี้เลือก File > New > Test Data จากเมนูหลัก
- จากนั้นหน้าต่าง New Test Data ปรากฏขึ้น เพื่อให้ผู้ใช้ระบุชื่อของ test data และเลือก Data Type ในรูปของ Excel File. ตามลำดับ



The screenshot shows a 'New' dialog box for creating test data. It includes a title bar, a 'Test Data' section with an information icon and 'Create Test Data' text, and three input fields: 'Name' (containing 'New Excel Data'), 'Data Type' (a dropdown menu set to 'Excel File'), and 'Description' (a large empty text area). The 'OK' and 'Cancel' buttons are at the bottom right.

Select Excel file

- จากนั้นดับเบิลคลิกที่ชื่อของ Test Data ภายใน Data Files เพื่อ import ไฟล์ Excel เข้าสู่ Project ใน Katalon Studio โดยการระบุ Path และ Sheet ตามลำดับ
- จากนั้นข้อมูลจากไฟล์ Excel เลือกไว้จะถูกโหลดเข้าสู่ preview section ดังรูป

File Name

../../MyTestProject/GasData2.xlsx

Browse...

Sheet Name

Sheet1

▼

☒ Use first row as header
 ☒ Use relative path

No.	endMiles	startMiles	gallon	price	Miles_Gallon	Result
1	15000	10000	100	4	50.00 miles per gallon	
2	35000	26000	24	3.5	363.64 miles per gallon	
3	40000	30000	26	3.5	384.62 miles per gallon	

Data Driven Testing : Test Case Level

- ขั้นตอนการสร้าง Data Driven Testing ในระดับ Test Case
 - สร้าง Project และ Test Case ตามลำดับ
 - สร้างสคริปต์สำหรับการทดสอบไว้
 - สร้างไฟล์ข้อมูลทดสอบที่มีนามสกุล .xlsx
 - Import ไฟล์ข้อมูลลงใน Project โดยการระบุ Path และ Sheet
 - อ้างอิงค่า input ภายในสคริปต์กับไฟล์ข้อมูล โดยการระบุชื่อพร้อมแถวและคอลัมน์
 - ในกรณีที่มีข้อมูลมากกว่า 1 แถว สร้าง Loop และกำหนดตัวแปรเพื่อรองรับแถวและคอลัมน์
 - บันทึกข้อมูล Test Case และรันการทดสอบ

Demo : Gas-Mileage-Calculator

- สร้าง Test Script จาก Test Plan ที่กำหนด โดยใช้ Chropath ช่วยเขียนสคริปต์ เพื่อเรียกใช้ข้อมูลจากไฟล์ Excel จาก Test Case ดังต่อไปนี้

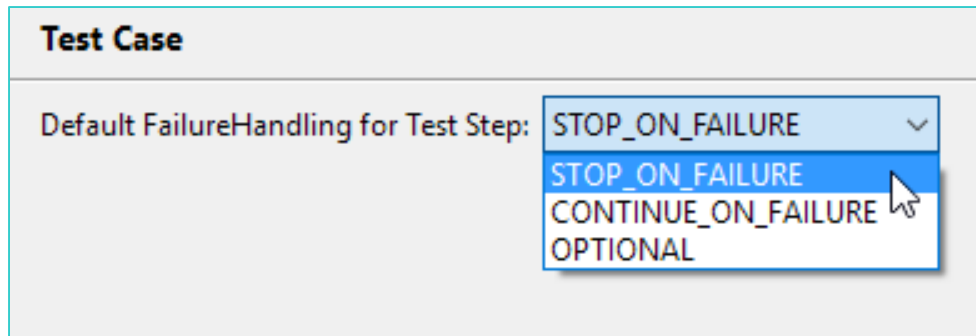
Step #	Step Detail	Detail
1	Open Browser	
2	Go to the Application URL	http://www.calculator.net/ /gas-mileage-calculator.html
3	Input current odometer reading	12000
4	Input Previous Odometer	10000
5	Input gas put in the tank	40
6	Input gas price	4.5
7	Click Calculate image	Yes
8	Verify Match	50.00 miles per gallon
9	Close Browser	Yes

Failure handling

- ผู้ใช้สามารถตั้งค่า Failure handling เพื่อให้ Katalon Studio ทำงานต่อเนื่องหรือไม่ในกรณีที่มีความผิดพลาดเกิดขึ้นระหว่างการประมวลผล
- Katalon Studio สนับสนุนการจัดการ failure handling ใน 3 ทางเลือก ได้แก่
 - STOP_ON_FAILURE โดยหยุดการประมวลผลในขั้นตอนการทดสอบนั้น ๆ เมื่อมีความผิดพลาดเกิดขึ้น
 - CONTINUE_ON_FAILURE ประมวลผลต่อเนื่องแม้ว่าจะมีความผิดพลาดเกิดขึ้นก็ตาม
 - OPTIONAL ประมวลผลต่อเนื่องแม้ว่าจะมีความผิดพลาดเกิดขึ้น แต่ขั้นตอนที่ผิดพลาดจะมีสถานะเป็น warning

Default failure handling

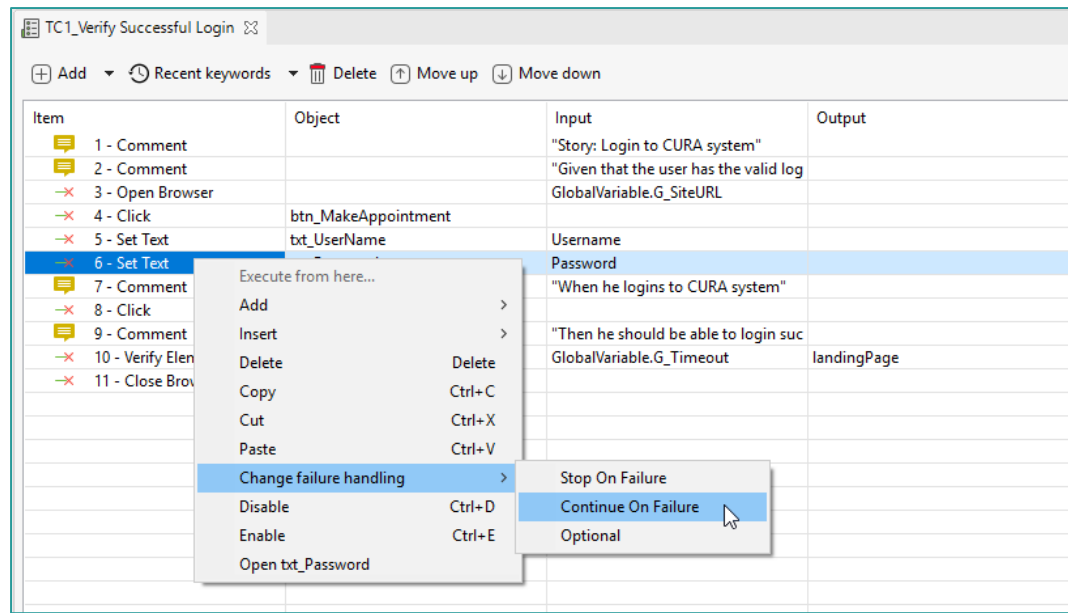
- การจัดการ Failure ในระดับ Project สามารถตั้งค่าได้จากการคลิกที่เมนูหลัก **Project > Settings > Test Case**



- โดยปกติแล้วค่า Default ในการจัดการ Failure ของ Katalon Studio จะอยู่ที่ STOP_ON_FAILURE ของทุก ๆ Test Case ในระดับ Project

Failure handling In Manual View

- นอกจากนั้นผู้ใช้อย่างยังสามารถกำหนดค่า failure ได้ในระดับคำสั่งโดยการกำหนดค่าได้ทั้งในระดับ **Manual view** หรือ **Scripting view** ของ test case.
- ใน Manual View สามารถทำได้โดยการคลิกเลือกที่คำสั่งดังรูป



Failure handling In Scripting View

- ทุกคำสั่ง built-in คีย์เวิร์ดใน Katalon Studio สามารถระบุการจัดการ *Failure Handling* ได้โดยระบุในค่าพารามิเตอร์ตัวสุดท้ายใน Scripting mode ในทางเลือกดังต่อไปนี้
 - FailureHandling.STOP_ON_FAILURE
 - FailureHandling.CONTINUE_ON_FAILURE
 - FailureHandling.OPTIONAL

```
WebUiBuiltInKeywords.openBrowser('', FailureHandling.STOP_ON_FAILURE)  
WebUiBuiltInKeywords.navigateToUrl('', FailureHandling.STOP_ON_FAILURE)  
WebUiBuiltInKeywords.setText(ObjectRepository.findTestObject(null), '', FailureHandling.STOP_ON_FAILURE)  
WebUiBuiltInKeywords.click(ObjectRepository.findTestObject(null), FailureHandling.STOP_ON_FAILURE)
```

Data Driven Testing : Test Suite Level

- Test Suite Level
 - สร้าง Test Case ใหม่
 - สร้างไฟล์ข้อมูลทดสอบนามสกุล .xlsx
 - สร้าง Variable ใน Manual Mode เพื่อใช้อ้างอิงภายใน Test Case
 - เพิ่มไฟล์ข้อมูลลงใน Katalon Studio
 - เพิ่ม Test Case ลงใน Test Suite ที่กำหนดไว้
 - คลิกเลือก Show Data Binding และ Map ข้อมูลเข้าด้วยกัน
 - บันทึกข้อมูล Test Case และรันการทดสอบ
- ความแตกต่างอยู่ตรงที่ไม่มีการใช้ Loop ในการทำงาน นอกจากนั้นการอ้างอิงในส่วนของ input จากตัวแปร เนื่องจาก Data Binding จัดการให้โดยอัตโนมัติ

Test Data

- หลังจากสร้าง Test Suite เสร็จแล้ว ขั้นตอนต่อไปเป็นการกำหนด Test Data และ Variable Binding เพื่ออ้างอิงระหว่างตัวแปรที่ใช้และ Test Data จากไฟล์ Excel ที่กำหนดไว้ โดยการเลือกค่าดังรูป

▼ Test Data

Add Delete Move Up Move Down Map All

	No.	ID	Data Iteration	Type
	1	Data Files/GasTest Data	All	One

< >

Variable Binding

Set Type Set Test Data ▼

	No.	Name	Default value	Type	Test Data
	1	endMiles	"	Data Column	1 - Data Files/Gas
	2	startMiles	"	Data Column	1 - Data Files/Gas
	3	gallon	"	Data Column	1 - Data Files/Gas
	4	price	"	Data Column	1 - Data Files/Gas
	5	Miles_Gallon	"	Data Column	1 - Data Files/Gas

< >

Variable Binding

- เลือก Test Data และ Choose column ตามลำดับ

