



Katalon

Custom Keyword

27–28 Apr., 4–5 May 2019

@IT MJU

Assoc. Rangsit Sirirangsi

www.indythaitester.com

Groovy : method

- เมธอดใน Groovy สามารถประกาศร่วมกับ return type หรือคือใช้วิธี def
- เมธอดสามารถรับค่าจำนวน arguments ได้ตามต้องการ โดยไม่จำเป็นต้องประกาศชนิดข้อมูลไว้สำหรับ arguments
- Modifiers เช่น public, private และ protected สามารถกำหนดไว้ได้ โดยค่า default ได้แก่ public ในกรณีที่ไม่มีการกำหนด Visibility ไว้
- การกำหนดเมธอดที่ไม่มีการรับค่าพารามิเตอร์สามารถทำได้ดังนี้

```
def methodName() {  
    //Method code  
}
```

Calling methods

- การเรียกใช้เมธอดใน Groovy ไม่จำเป็นต้องระบุสัญลักษณ์วงเล็บ ยกเว้นในกรณีที่ไม่มีการรับค่าพารามิเตอร์ต้องระบุวงเล็บเสมอ ตัวอย่างเช่น

```
def myMethod() {  
    // ...  
}
```

```
def myOtherMethod(someArg1, someArg2) {  
    // ...  
}
```

- ตัวอย่างการเรียกใช้เมธอด

myMethod()	// OK
myMethod	// error
myOtherMethod(2, 3)	// OK
myOtherMethod 4, 5	// OK

Groovy : method return value

- ตัวอย่างการสร้างเมธอดภายในคลาส Example

```
class Example {  
    public static def foo(int i) {  
        if ( i < 100 ) {  
            return -1  
        }  
        return 'Hello World'  
    }  
    static void main(String[] args) {  
        println foo(50)  
        println foo(200)  
    }  
}
```

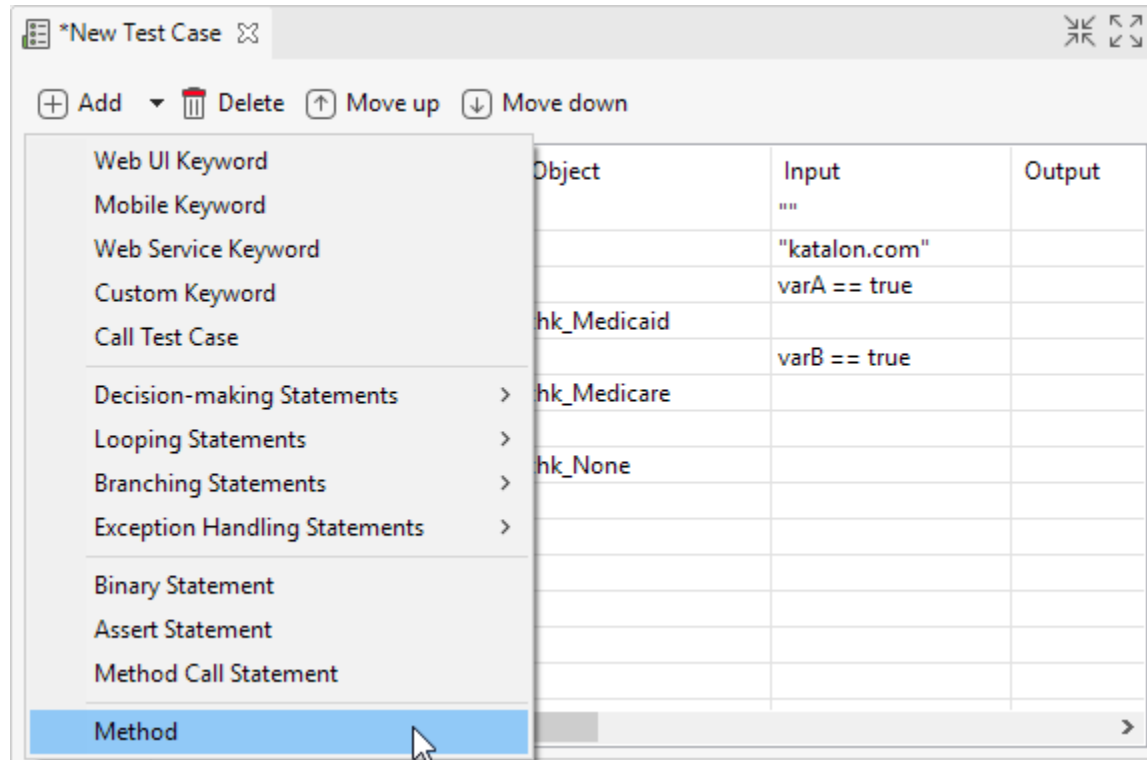
output : -1

Hello World

- ชนิดการคืนค่าที่ระบุเป็น def สามารถคืนค่าข้อมูลได้หลายชนิด

Define a method in Manual view

- ขั้นตอนการสร้างเมธอดเริ่มต้นจากเปิด test case ใน **Manual** view จากนั้นเลือก **Method** จาก Add ดังรูป



Method builder

- ผู้ใช้สามารถเลือกกำหนดค่าต่าง ๆ ที่จำเป็นสำหรับการสร้างเมธอด ได้แก่ Name, Return type, Setup, Teardown options และ Parameter list ลงใน **Method builder** ดังต่อไปนี้

Method builder

Name: myMethod1

Return type: java.lang.Integer

☐ Set up ☐ Tear Down ☐ Tear Down If FAILED ☐ Tear Down If ERROR ☐ Tear Down If PASSED

Param Type	Param Name
java.lang.Object	param1
java.lang.Object	param2

Insert Delete OK Cancel

Manual Mode : Method

- หลังจากเมธอดถูกสร้างขึ้นเรียบร้อยแล้ว ขั้นตอนถัดไปเป็นการเรียกใช้เมธอด โดยกำหนดพารามิเตอร์ตามที่กำหนดไว้ดังรูป

Item	Object	Input
✗ 1 - Open Browser		""
✗ 2 - Navigate To Url		"http://jqueryui.com/datepicker/"
✗ 3 - Switch To Frame	iframe	3
✗ 4 - Click	dateBox	
<i>f_x</i> 5 - Method Call Statement		selectDate(findTestObject("Object Re
> <i>f_x</i> selectDate()		

- ขั้นตอนสุดท้ายบันทึก test case
- เมื่อ test step ถูกประกาศไว้ในรูปของเมธอดแล้ว Katalon จะไม่ยอมให้มีการเปลี่ยนแปลงแก้ไขได้ในอีกเวอร์ชัน

Classes, Methods and Objects Declaration

- คลาสและเมธอดใน Groovy ใช้รูปแบบเช่นเดียวกับจาวา ตัวอย่างเช่น

```
class Product {  
    private String name  
    private def price  
    Product(name, price, String vendor) {  
        this.name = name  
        this.price = price  
    }  
    def setName(name){  
        this.name = name  
    }  
    public String getPrice(){  
        return price  
    }  
    def toString(){  
        return "Name = $name, Price =  
        $price, Vendor = $vendor";  
    }  
}
```

```
def p1 = new Product("Mobile", "10000")  
println(p1.toString())  
def p2 = new Product(name: 'Laptop',  
price: "540000")  
println(p2.toString())
```

Name = Mobile, Price = 10000
Name = Laptop, Price = 540000

Steps to Create Custom Keyword

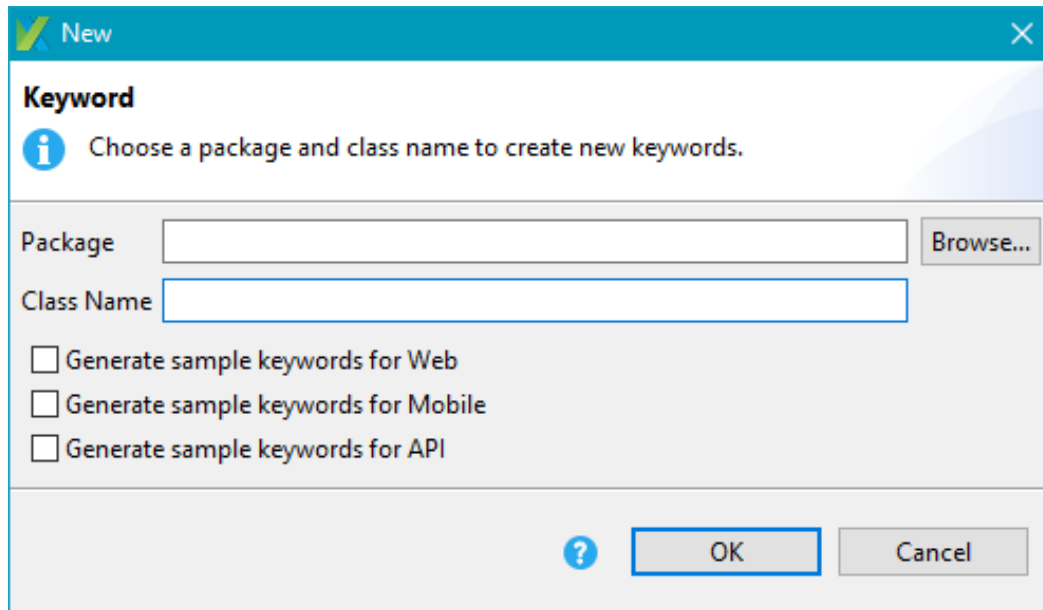
- สร้างแพ็คเกจใหม่ภายในโฟลเดอร์ keyword
- ภายในแพ็คเกจดังกล่าวสร้างคลาสที่ต้องการ
- ภายในคลาสเมธอดสร้างเมธอดที่ต้องการ
- กำหนด @Keyword ไว้ที่เมธอด
- ภายใน Test Case เพิ่ม Custom Keyword เพื่อเรียกใช้ keyword ที่กำหนดไว้



**CUSTOM
KEYWORD**

Create a Package & Class

- เพื่อกำหนดหน้าที่จัดกลุ่มของ custom keywords ไว้เป็นหมวดหมู่ต่าง ๆ
- คลิกเลือกที่ **File > New > Package** จากเมนูหลัก จากนั้นจะปรากฏหน้าต่าง **New Keyword Package** เพื่อให้ผู้ใช้ระบุชื่อของแพ็คเกจและคลาสดังรูป



New

Keyword

i Choose a package and class name to create new keywords.

Package

Class Name

☐ Generate sample keywords for Web

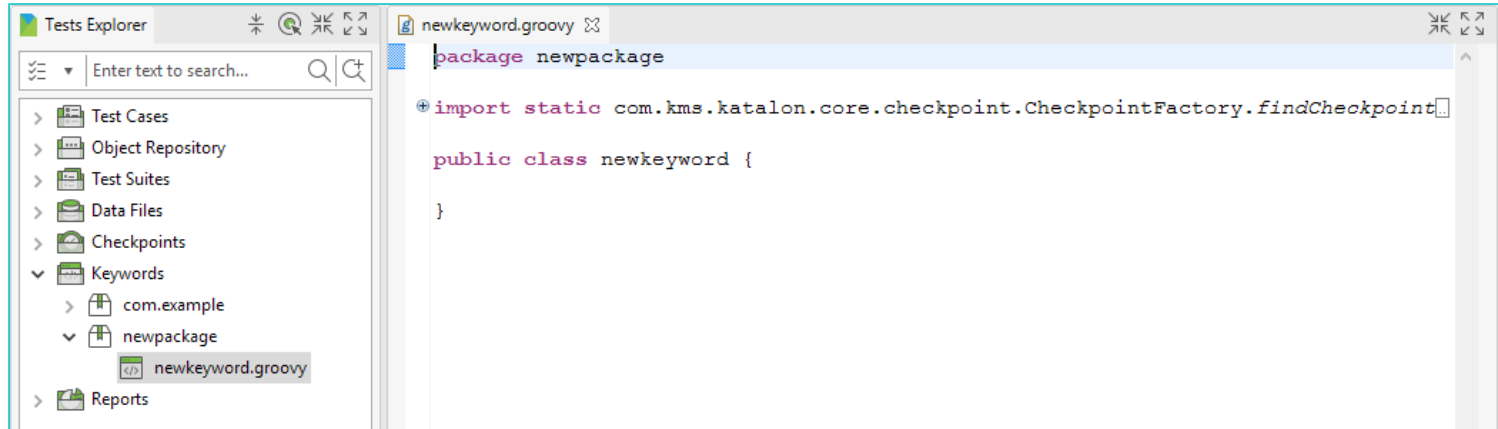
☐ Generate sample keywords for Mobile

☐ Generate sample keywords for API

?

Keyword Class

- keyword ใหม่ถูกสร้างขึ้นภายในแพ็คเกจที่กำหนดไว้ ผู้ใช้สามารถตรวจสอบได้จากโค้ดที่ปรากฏขึ้นภายใน Script View



Keyword : method

- ขั้นตอนต่อไปเป็นการโค้ดรายละเอียดของเมธอดภายในคลาสที่กำหนดไว้ โดยระบุ `@Keyword` เพื่อสร้างคีย์เวิร์ดที่พร้อมสำหรับการเรียกใช้

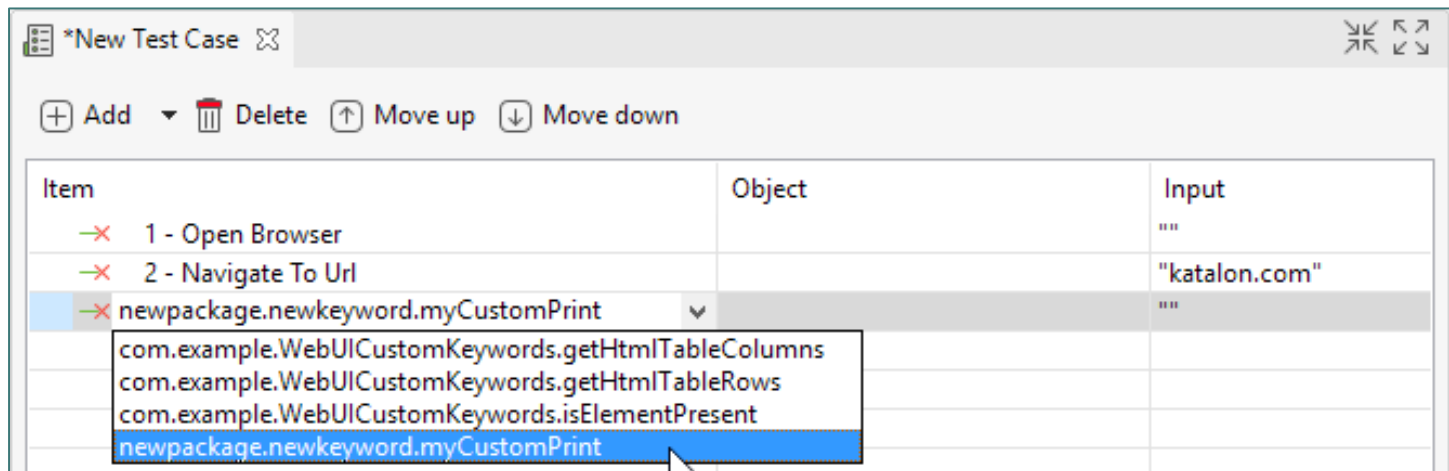
`@Keyword`

```
def keywordName(parameters...) {  
    // enter your code here  
    // you can use either Groovy or Java  
}
```

Item	Description
@Keyword	The annotation to indicate that the block of code below is the definition of a keyword.
parameters	The list of parameters that would be used in the custom keyword
keywordName	The name that you would like to use for your custom keyword

Custom keywords in Manual view

- การเรียกใช้ Custom Keyword ภายใน Manual View สามารถทำได้โดยการเลือก Add => Custom Keyword จะปรากฏรายชื่อภายใน Test Step เพื่อให้ผู้ใช้สามารถเลือกใช้ได้ดังรูป



Demo : AutoComplete

- สร้าง Test Script ที่กำหนดโดยใช้ ChroPath ช่วยในการ Add Property โดยใช้คำสั่ง Selenium ร่วมกับ Katalon เพื่อแสดงผล AutoComplete จาก Test Case ดังต่อไปนี้

TC #	Test Step	Test Data
1	Go to the Application URL	http://www.google.co.th
2	Input search query	Katalon Studio
3	Get name of Autocomplete	xpath
4	Display names from Autocomplete	
5	Compare names with Test Data	katalon studio vs selenium
6	Click Test Data to search	
7	Close Browser	

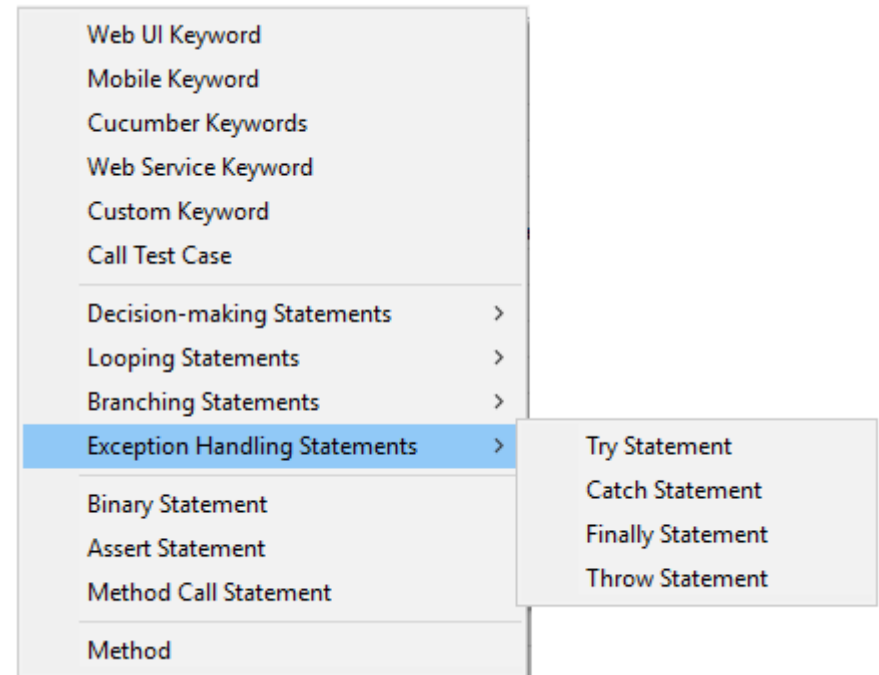
- สร้าง Custom Keyword เพื่อทำงานแทนเมธอดที่ถูกสร้างไว้แล้ว

What is Exception handling?

- เป็นความผิดปกติที่เกิดขึ้นระหว่างการประมวลผล และมีผลทำให้ลำดับการทำงานของชุดคำสั่งภายในโปรแกรมเปลี่ยนไป ซึ่งความผิดปกติดังกล่าวอาจก่อให้เกิด error ภายในโปรแกรม
- Error ในการโปรแกรมแบ่งออกได้เป็น 2 กลุ่ม ได้แก่
 - **Compile-time errors** เกิดขึ้นจากการเขียนโปรแกรมที่ผิดไปจากรูปแบบของโปรแกรมภาษาที่ใช้
 - **Run-time errors** เกิดขึ้นระหว่างการประมวลผลโปรแกรม
- Exception เป็นความผิดปกติที่เกิดขึ้นในช่วงเวลารันไทม์ เพื่อที่จะจัดการความผิดปกติดังกล่าวจำเป็นต้องใช้กลไก Exception Handling โดยการเรียกใช้คีย์เวิร์ด Throw

Katalon : Exception Handling

- การจัดการ Exception ใน Katalon Studio สามารถทำได้ทั้งใน Manual View และ Script View โดย Manual View จะเรียกใช้จากเมนูดังรูป
- ส่วนใน Script view ผู้ใช้สามารถเขียนโปรแกรมเพื่อจัดการ Exception ได้โดยตรงผ่านโปรแกรมภาษา Groovy หรือ Java



Handling Exception

- เป็นการจัดการความผิดพลาดเพื่อให้การทำงานของโปรแกรมสามารถดำเนินการต่อเนื่องไปได้ โดยปกติมีอยู่ 2 รูปแบบ ได้แก่
 - ใช้ throws statement เพื่อจัดการกับ exception ภายในเมธอดที่ต้องการ

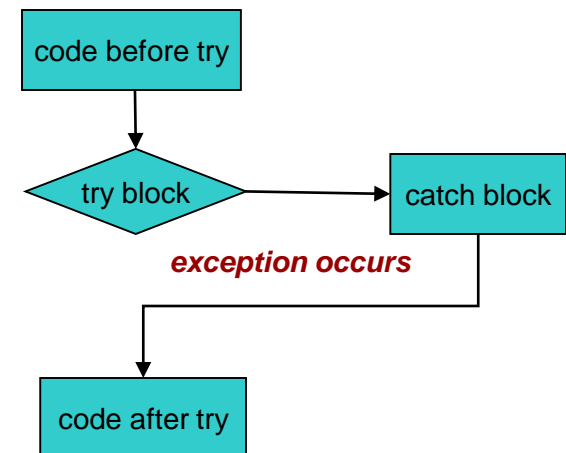
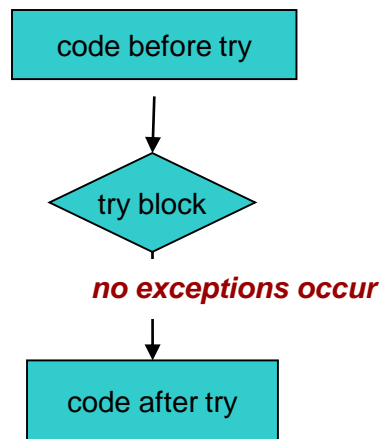
```
if( condition)  
    throw new Exception("Message to display");
```

- ใช้กลไกที่เรียกว่า try-catch ภายในเมธอด

```
try{  
    // code that may throw exceptions  
}  
  
catch (ExceptionType ref) {  
    exception handling code  
}
```

Handling Exception : Mechanism

- กลไกการจัดการ Exception มีลักษณะการทำงานเป็นสองส่วน ได้แก่
 - กรณีที่ไม่มีความผิดปกติเกิดขึ้น โค้ดสำหรับจัดการ Exception จะถูกข้ามไปโดยอัตโนมัติ
 - กรณีที่มีความผิดปกติเกิดขึ้น โค้ดสำหรับจัดการ Exception จะถูกเรียกใช้โดยอัตโนมัติ



Exception Handling

- throws statement หากใช้จัดการ exception ร่วมกับการกำหนดเงื่อนไขเพียงอย่างเดียวจะส่งผลให้การทำงานสิ้นสุดลงทันที
- ดังนั้นส่วนใหญ่จึงมักใช้ร่วมกับกลไกแบบ try-catch เพื่อแสดงข้อความแจ้งเตือนผู้ใช้ และสามารถทำงานได้ต่อไป

```
String date = '38'
try {
    if (Integer.parseInt(date) > 31)
        throw new Exception("Date should less than 31");
}
catch (Exception e) {
    println(e)
}
```



Exception Occurred

Exception Handling : Code

```
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import org.openqa.selenium.TimeoutException
import com.kms.katalon.core.logging.KeywordLogger
import com.kms.katalon.core.util.KeywordUtil
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI

    try{
        WebUI.click(findTestObject("Page_Register/btn_register"))
    } catch (TimeoutException toe) {
        WebUI.waitForElementClickable(findTestObject("Page_Register/btn_register"), 20)
        WebUI.click(findTestObject("Page_Register/btn_register"))
    }catch (Exception e) {
        KeywordUtil.markError("Register button is not found.")
        throw(e);
    }
```

WebTable

- Tables เป็นเครื่องมือสำคัญในการออกแบบด้วย HTML โดย Tables จะถูกใช้รวบรวมส่วนประกอบอื่น ๆ ของ HTML อื่น ๆ ไว้ด้วยกัน
- โดยปกติแล้ว Table ประกอบด้วยแถวต่าง ๆ โดยแต่ละแถวอาจประกอบไปด้วยคอลัมน์จำนวนตั้งแต่หนึ่งหรือมากกว่า การทำงานกับ Table โดยปกติจะเริ่มตั้งแต่การนับจำนวนแถวและคอลัมน์ตามลำดับ

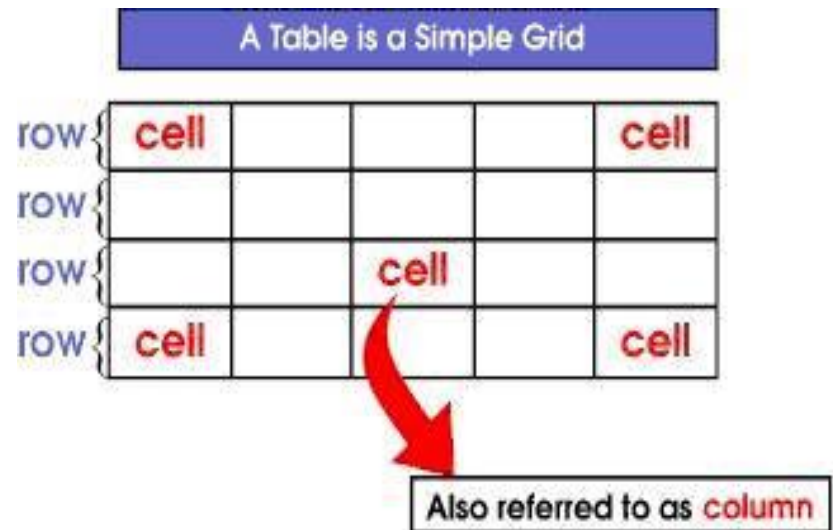


Table Concepts

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
</table>
```

Month	Savings
January	\$100
February	\$80

Tag	Description
<table>	ใช้สร้างตารางข้อมูล
<th>	สำหรับข้อความที่เป็น Head จะเป็นตัวหนา
<tr>	สำหรับแถวของตาราง
<td>	สำหรับข้อมูลในแต่ละ cell
<caption>	คำอธิบายตาราง
<thead>	กำหนดส่วน head ของตาราง
<tbody>	กำหนดส่วน body ของตาราง

Ex : Training System

- จากการทำงานของ FillRegisterProfile, Login และ Delete Trainee จาก QtpRegisterDemo
 - อ่านข้อมูลจากไฟล์ Excel เดียวกันแต่ Sheet แตกต่างกัน
 - เรียกใช้การทำงานตามลำดับโดยอาศัย Call Test Case
 - สร้าง Test Suite เพื่อแสดงผลลัพธ์การทดสอบ
 - สร้าง Custom Keyword ในฟังก์ชันการทำงานที่จำเป็น ได้แก่ การจัดการ PopUp เพื่อแสดงการ Pass หรือ Fail ของการทำงาน