

MODUL 5

RESULT AND REPORTING

A. Tujuan

1. Mengetahui apa itu test execution reports pada katalon.
2. Mengetahui bagaimana cara melihat hasil test execution pada katalon.
3. Mengetahui bagaimana cara melihat hasil report pengujian dalam format tertentu.
4. Mengetahui bagaimana cara melihat hasil report pengujian pada email.
5. Mengetahui apa itu analytic katalon pada kataon studio.

B. Dasar Teori

Test execution report merupakan cara untuk melihat hasil pengujian yang telah dilakukan, dimana hasil pengujian ini akan ditampilkan dalam bentuk laporan.

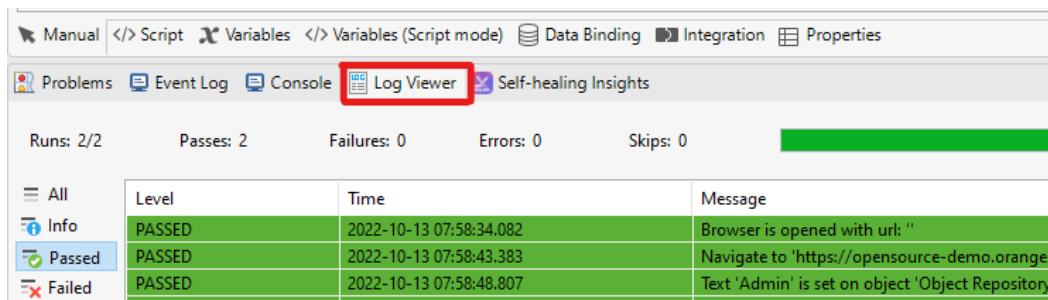
5.1 TEST EXECUTION REPORTS

Didalam katalon studio terdapat dua fitur yang menyajikan hasil eksekusi dari setiap test yang dijalankan.

- Log : untuk menyajikan hasil eksekusi dari test case, test suite, dan test suite collection
- Report : untuk menyajikan hasil eksekusi dari test suite dan test suite collection

5.1.1. Log Viewer

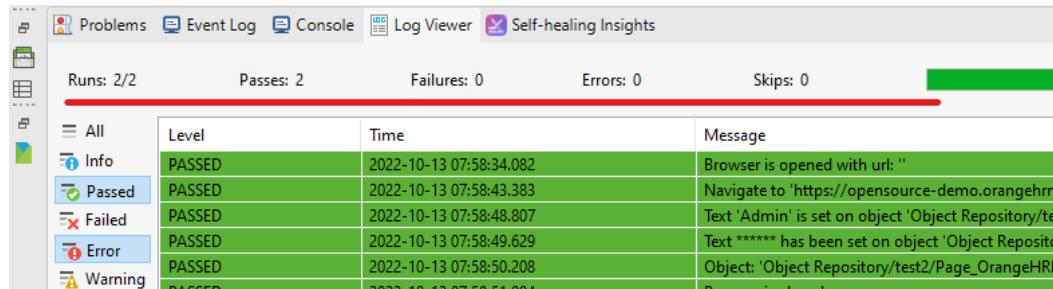
1. Kita dapat menggunakan tools log viewer untuk melihat log dari setiap test case.



The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab highlighted in red. The top navigation bar includes 'Manual', '</> Script', 'Variables', 'Variables (Script mode)', 'Data Binding', 'Integration', and 'Properties'. Below the navigation bar, there are tabs for 'Problems', 'Event Log', 'Console', 'Log Viewer' (which is active), and 'Self-healing Insights'. A summary bar displays 'Runs: 2/2', 'Passes: 2', 'Failures: 0', 'Errors: 0', and 'Skips: 0'. A green progress bar is shown next. The main area is a table with columns 'Level', 'Time', and 'Message'. The table has three rows: one for 'All' (Info level), one for 'Passed' (Info level), and one for 'Failed' (Info level). The 'Failed' row is partially cut off at the bottom.

All	Level	Time	Message
Info	PASSED	2022-10-13 07:58:34.082	Browser is opened with url: "
Passed	PASSED	2022-10-13 07:58:43.383	Navigate to 'https://opensource-demo.orangehr...
Failed	PASSED	2022-10-13 07:58:48.807	Text 'Admin' is set on object 'Object Repository/

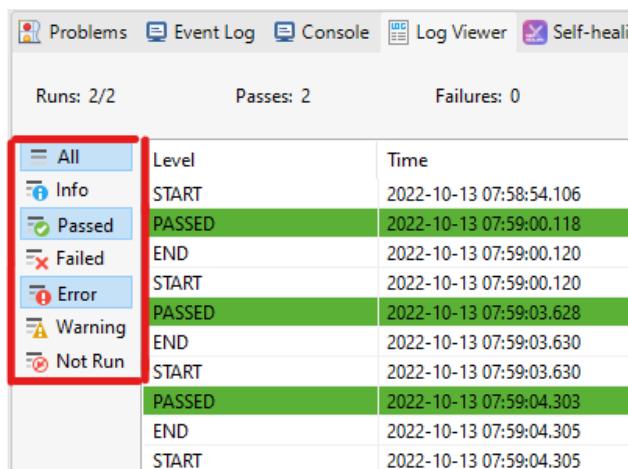
2. Dalam jendela log viewer kita dapat melihat tes yang dijalankan, tes yang lulus uji, kesalahan, error, dan test yang dilewati.



The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab selected. At the top, it displays statistics: Runs: 2/2, Passes: 2, Failures: 0, Errors: 0, and Skips: 0. Below this is a table with columns for Level, Time, and Message. The table contains six rows, all of which are green, indicating they are 'PASSED'. The messages in the table relate to navigating to URLs and setting object repository values.

Level	Time	Message
PASSED	2022-10-13 07:58:34.082	Browser is opened with url: ''
PASSED	2022-10-13 07:58:43.383	Navigate to 'https://opensource-demo.orangehrmlive.com/'
PASSED	2022-10-13 07:58:48.807	Text 'Admin' is set on object 'Object Repository/test2/Page_OrangeHRM/Text_Admin'
PASSED	2022-10-13 07:58:49.629	Text ***** has been set on object 'Object Repository/test2/Page_OrangeHRM/Text_*****'
PASSED	2022-10-13 07:58:50.208	Object: 'Object Repository/test2/Page_OrangeHRM/Button_Login'
PASSED	2022-10-13 07:58:51.001	Object: 'Object Repository/test2/Page_OrangeHRM/Text_Error'

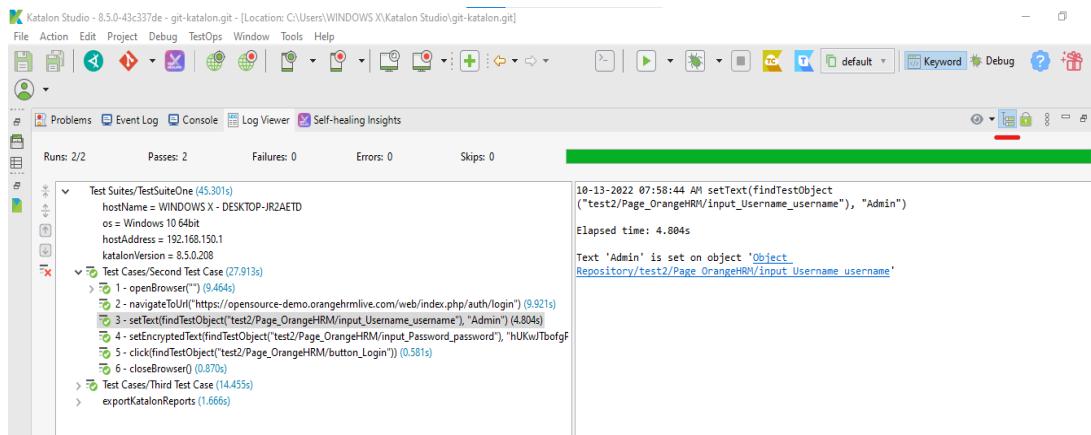
3. Jika ingin memfilter log berdasarkan hasil kita dapat menggunakan menu pada jendela log sebelah kiri



This screenshot shows the same Katalon Studio interface as above, but with the 'Passed' filter selected from the left sidebar. The table now includes several additional rows that were previously hidden, such as 'START' and 'END' entries for each step in the test script. The 'PASSED' status is consistently shown for these additional entries.

Level	Time
START	2022-10-13 07:58:54.106
PASSED	2022-10-13 07:59:00.118
END	2022-10-13 07:59:00.120
START	2022-10-13 07:59:00.120
PASSED	2022-10-13 07:59:03.628
END	2022-10-13 07:59:03.630
START	2022-10-13 07:59:03.630
PASSED	2022-10-13 07:59:04.303
END	2022-10-13 07:59:04.305
START	2022-10-13 07:59:04.305

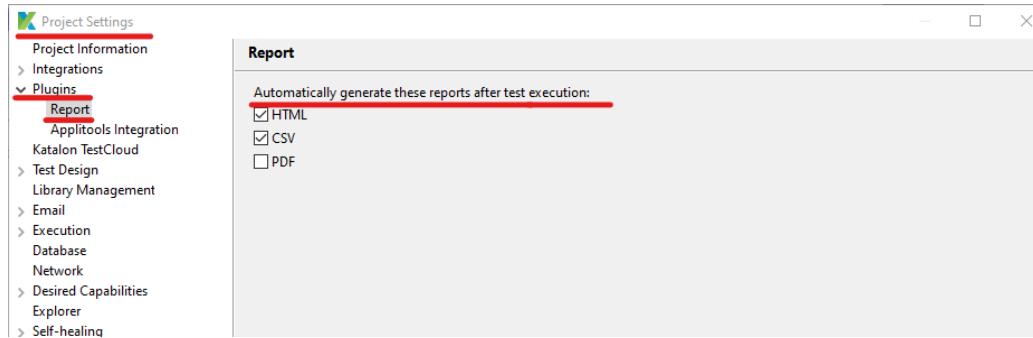
4. Katalon juga menyediakan tampilan hirarki yang didalamnya terdapat jendela log dan jendela detail dari setiap tahapan pengujian



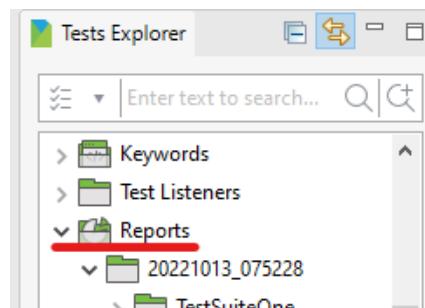
5.1.2. Report

1. Dalam katalon studio kita dapat mengatur format report yang digunakan untuk menyajikan report melalui menu project > setting > plugins > report

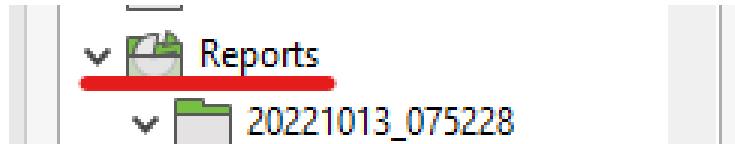
Ada tiga jenis format report yang disajikan yaitu html, csv, dan pdf.



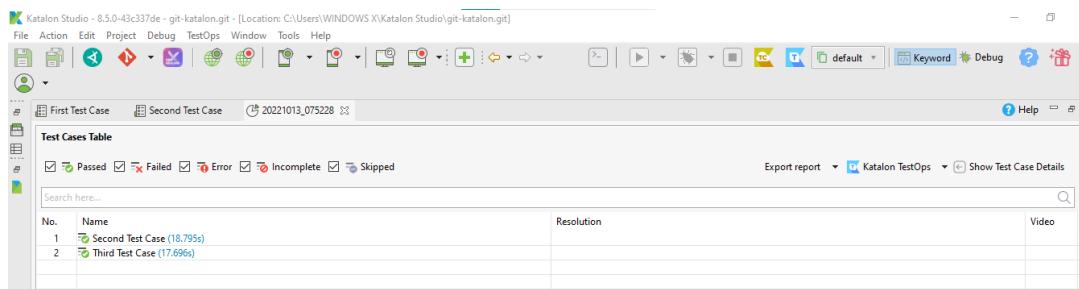
2. Untuk melihat report kita dapat menuju jendela explorer dan memilih menu report



3. Report yang disajikan sesuai dengan jumlah eksekusi yang dilakukan terhadap test suite ataupun test suite collection. Penamaan report secara default diawali oleh tahun, bulan, dan tanggal atau sesuai dengan waktu eksekusi



4. Berikut ini adalah tampilan jendela report secara keseluruhan.



5. Terdapat test case table yang akan menampilkan test case yang diuji beserta dengan status kelulusan, kita juga dapat memfilter tampilan dengan menggunakan checkbox di bagian atas jendela test case table.

Test Cases Table	
<input checked="" type="checkbox"/> Passed <input type="checkbox"/> Failed <input type="checkbox"/> Error <input type="checkbox"/> Incomplete <input type="checkbox"/> Skipped	
<input type="text"/> Search here...	
No.	Name
1	Second Test Case (18.795s)
2	Third Test Case (17.696s)

6. Di bagian bawah jendela report juga terdapat ringkasan pengujian yang terletak pada tab summary.

Summary		Execution Settings	Execution Environment
Test Suite ID	Test Suites/TestSuiteOne		
Host name	WINDOWS X - DESKTOP-JR2AETD	Local OS	Windows 10 64bit
Katalon version	8.5.0.208	Platform	Chrome 106.0.0.0
Start	2022-10-13 07:52:59	End	2022-10-13 07:53:36
Elapsed	37.152s		
Total TC	2		
Passed	2	Failed	0
Error	0	Skip	0
Incomplete	0		

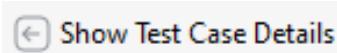
7. Kita juga dapat melihat seluruh setting saat eksekusi melalui tab Execution setting

Execution Settings	
Name	Value
autoApplyNeighborXpaths	false
ignorePageLoadTimeoutException	false
timeCapsuleEnabled	false
executionProfile	default
excludeKeywords	[verifyElementPresent, verifyElementNotPresent]
xpathPriority	[!left=xpath:attributes, right=true], [left=xpath:idRelative, right=true], [left=dom:name, right=true], [left=xpath:link, ...]
timeout	30.0
actionDelay	0.0
methodsPriorityOrder	[!left=XPATH, right=true], [left=BASIC, right=true], [left=CSS, right=true], [left=IMAGE, right=true]
proxy	{"proxyOption": "NO_PROXY", "proxyServerType": "HTTP", "username": "", "password": "", "proxyServerAddress": ""}
defaultFailureHandling	CONTINUE_ON_FAILURE
terminateDriverAfterTestCase	false
defaultPageLoadTimeout	30.0
report	{videoRecorderOptions:{enable=false, useBrowserRecorder=true, videoFormat=AVI, videoQuality=LOW, recordAllTests=false}}
enablePageLoadTimeout	true
terminateDriverAfterTestSuite	true
useActionDelayInSecond	SECONDS
testDataInfo	()
selfHealingEnabled	true
WebUI	()

8. Kita juga dapat melihat lingkungan pengujian melalui tab Execution Environtment

Execution Environment	
Name	Value
hostName	WINDOWS X - DESKTOP-JR2AETD
os	Windows 10 64bit
katalonVersion	8.5.0.208
browser	Chrome 106.0.0.0
hostAddress	192.168.1.90.1
sessionId	75ec0030e4e1df4c6017f904d89cbdb8
seleniumVersion	3.141.59
proxyInformation	ProxyInformation { proxyOption=NO_PROXY, proxyServerType=HTTP, username=, password=***** , proxyServerAddress= }
platform	Windows 10

9. Untuk melihat tampilan detail dari pengujian kita dapat menggunakan menu test case detail



10. Berikut adalah tampilan dari detail setiap pengujian, yang didalamnya terdapat tahapan pengujian, deskripsi, dan waktu pengujian.

Test Case's Log		
Test Log Information Integration		
<input checked="" type="checkbox"/> Info <input checked="" type="checkbox"/> Passed <input checked="" type="checkbox"/> Failed <input checked="" type="checkbox"/> Error <input checked="" type="checkbox"/> Incomplete <input checked="" type="checkbox"/> Warning <input checked="" type="checkbox"/> Not Run		
Search here...		
Item	Description	Elapsed
> <input checked="" type="checkbox"/> 1. openBrowser("")		8.137s
> <input checked="" type="checkbox"/> 2. navigateToUrl("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")		5.621s
> <input checked="" type="checkbox"/> 3. setText(findTestObject("test2/Page_OrangeHRM/input_Username_username"), "Admin")		1.762s
> <input checked="" type="checkbox"/> 4. setEncryptedText(findTestObject("test2/Page_OrangeHRM/input_Password_password"), "hUKwJTbofgPU9eVlw/CnDQ==")		0.677s
> <input checked="" type="checkbox"/> 5. click(findTestObject("test2/Page_OrangeHRM/button_Login"))		0.495s
> <input checked="" type="checkbox"/> 6. closeBrowser()		0.894s

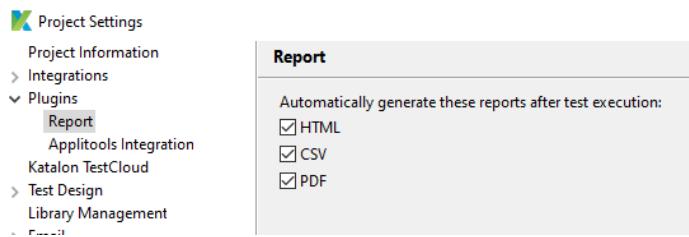
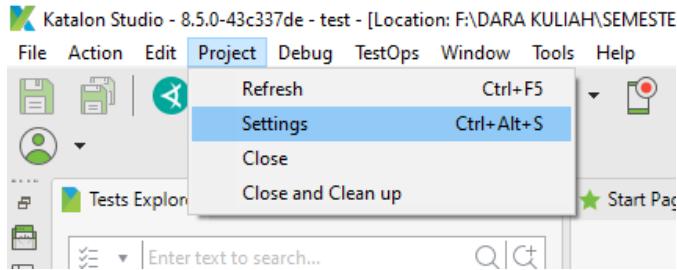
5.2 BASIC REPORTS PLUGINS

- Pada tahapan sebelumnya kita sudah banyak membuat test case, test suit, maupun test suit collection pada katalon. Salah satunya test case seperti yang terlihat pada gambar di bawah ini.

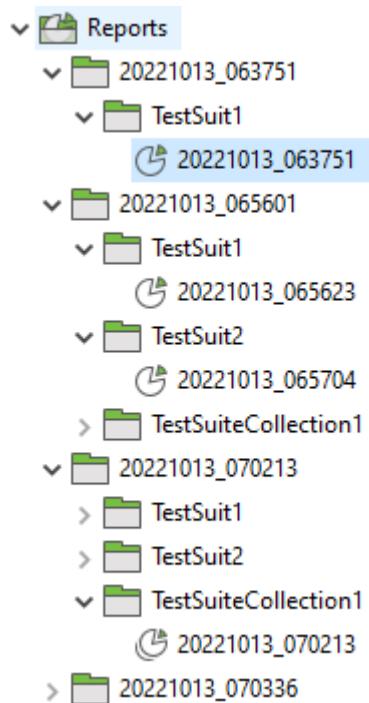
TestCase_1			
<input checked="" type="checkbox"/> Add <input checked="" type="checkbox"/> Recent keywords <input type="button" value="Delete"/> <input type="button" value="Move up"/> <input type="button" value="Move down"/> <input checked="" type="checkbox"/> Edit tags <input checked="" type="checkbox"/> Set default view			
Item	Object	Input	Output
-< 1 - Open Browser		""	
-< 2 - Navigate To Url		"https://opensource-demo.orangehrmlive.com/web/index.php/auth/login"	
-< 3 - Set Text	input_Username_username	"Admin"	
-< 4 - Set Encrypted Text	input_Password_password	"hUKwJTbofgPU9eVlw/CnDQ=="	
-< 5 - Click	button_Login		
-< 6 - Click	button_Search		
-< 7 - Click	a_Time		
-< 8 - Verify Element Present	a_Time	0	
-< 9 - Close Browser			

<input checked="" type="checkbox"/> Passed <input checked="" type="checkbox"/> Failed <input checked="" type="checkbox"/> Error <input checked="" type="checkbox"/> Incomplete <input checked="" type="checkbox"/> Skipped		Export
Search here...		
No.	Name	Resolution
1	Test1 (34.378s)	
2	TestCase_2 (24.477s)	

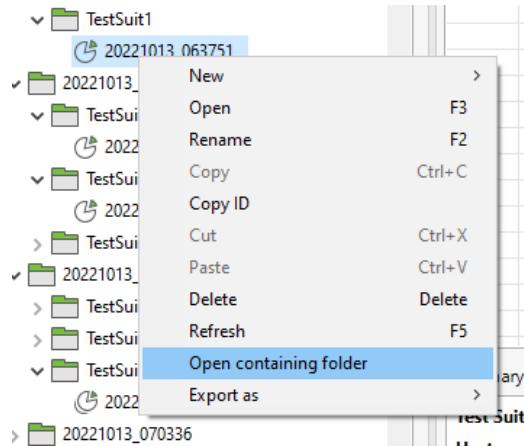
- Sebelum kita mulai mengeksport laporan hasil pengujian dalam format HTML, CSV, atay PDF, terlebih dahulu kita melakukan setting pada menu projek, dimana untuk semua plugins report ceklis untuk semua format pilihan yang ada.



3. Setelah itu pada menu reports, dapat kita lihat berbagai pengujian yang telah kita buat. Untuk kasus ini kita melakukan export terhadap pengujian test case yang sudah dilakukan penambahan pada test suit1.



4. Untuk melihat hasil report dalam format HTML, klik kanan pada report kemudian pilih **open containing folder**.



5. Selanjutnya user akan di arahkan ke halaman lokasi tempat penyimpanan folder.

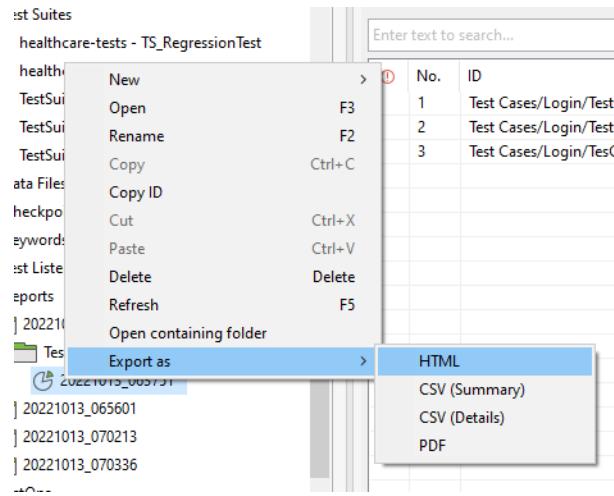
is PC > MASTER (F:) > DARA KULIAH > SEMESTER 7 > PL > TUGAS > PRAKTIKUM > Healthcare Sample > Reports

Name	Date modified	Type	Size
20221013_063751	10/13/2022 6:39 AM	File folder	

6. Ketika folder dibuka, user dapat melihat hasil report pengujian sudah ada dalam format HTML dan CSV, ini dapat terjadi karena sebelumnya kita telah melakukan setting terhadap report plugin untuk mengaktifkan ceklis terhadap berbagai pilihan format plugin yang tersedia.

20221013_063751.csv	10/13/2022 6:39 AM	Microsoft Office E...	3 KB
20221013_063751.html	10/13/2022 6:39 AM	Chrome HTML Do...	202 KB

8. Untuk melihat report hasil pengujian dalam format PDF, anda bisa klik kanan pada report, kemudian pilih **exports as**, lalu pilih format pdf.



Pilih tempat penyimpanan file, kemudian save. Berikut reprt hasil pengujian dalam format PDF1

TestSuit1

Execution Environment

Host name	ACER - DARA-MELISA-PC
Local OS	Windows 10 64bit
Katalon version	8.5.0.208
Browser	Chrome 106.0.0.0

Summary

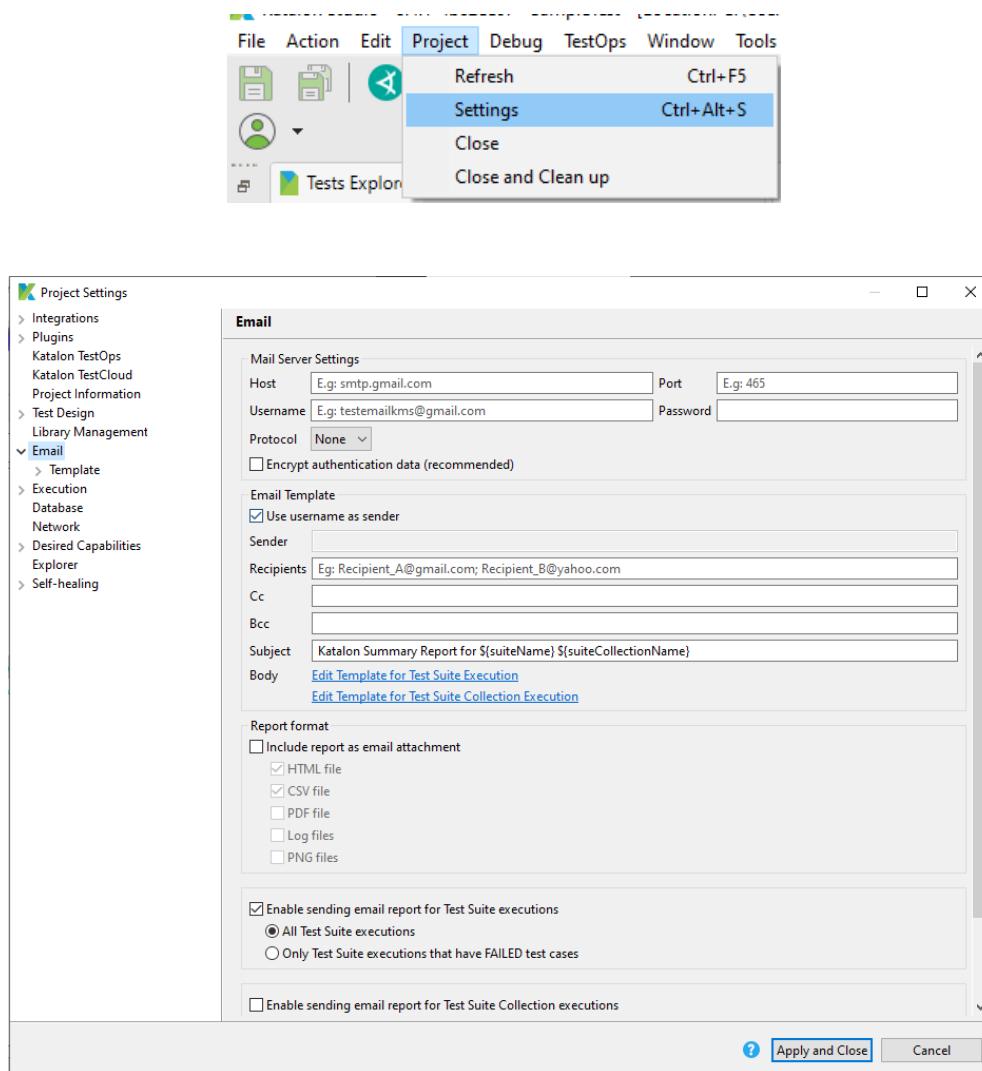
ID	Test Suites/TestSuit1
Description	
Total	2
Passed	2
Error	0
Skipped	0
Start	2022-10-13 06:38:26
Elapsed	1m - 1.449s
End	2022-10-13 06:39:28

#	ID	Description	Status
1	Test Cases/Test1		PASSED
2	Test Cases/TestCase_2		PASSED

5.3 HOW TO EMAIL RESULTS

Setelah melakukan pengujian perangkat lunak, terkadang seorang dari tim ingin hasil pengujian tersebut dikirim ke email secara otomatis agar menghemat waktu dalam kolaborasi sesama tim. Adapun langkah-langkah yang diperlukan untuk mengirim hasil pengujian ke email sebagai berikut :

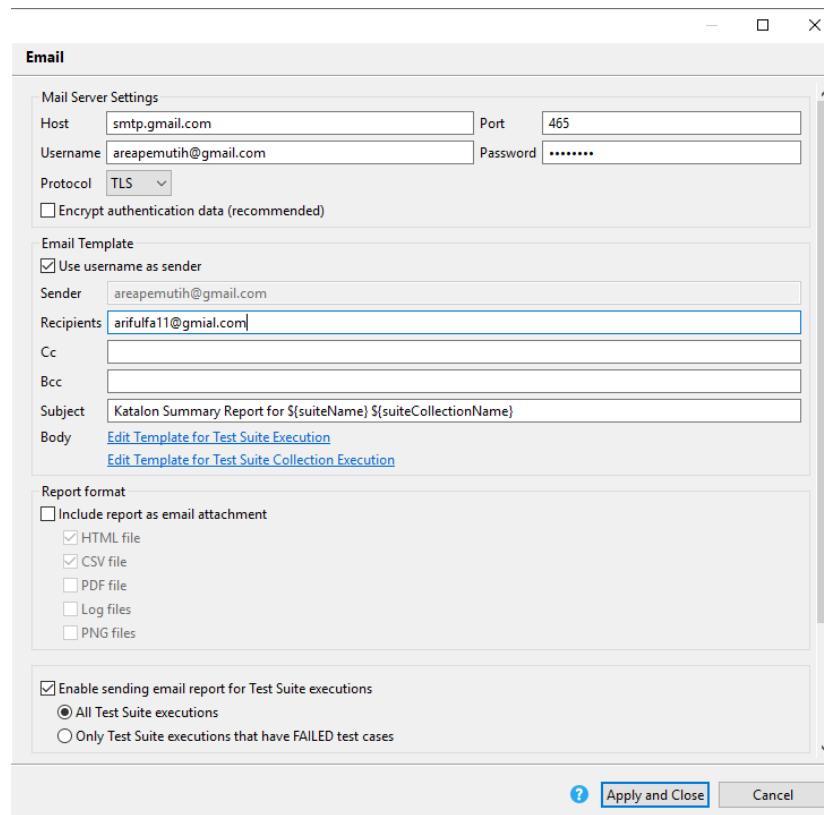
1. Bukan bagian Project>Settings>Email

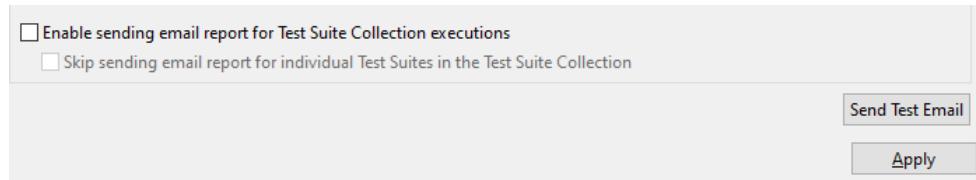


2. Sebelum mengisi kolom email, pahami dulu yang dimaksud dengan *Mail Server Settings*

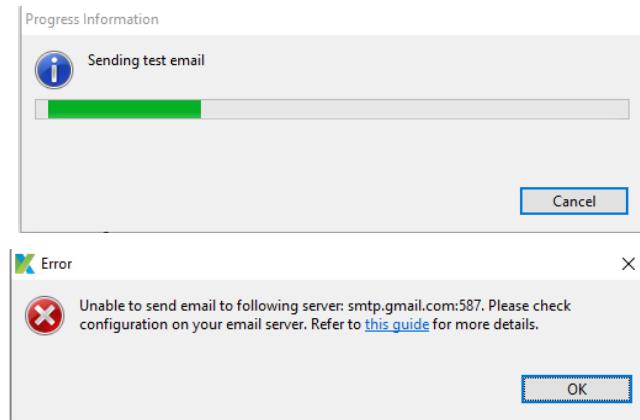
Email sever	Host	Port	Reference
Gmail	smtp.gmail.com	465 or 587	Check Gmail through other email platforms
Outlook	smtp.office365.com	587 or 25	How to set up a multifunction device or application to send email using Microsoft 365 or Office 365
Yahoo! Mail	smtp.mail.yahoo.com	465	POP access settings and instructions for Yahoo Mail

3. Setelah memahami isi dari *Mail Server Settings*. Selanjutnya isikan pada *email* seperti berikut ini, sesuaikan dengan email anda. Kemudian tekan *Send Test Email*, *email* akan dikirim kepada *email* yang terdapat pada kolom *Recipient*.

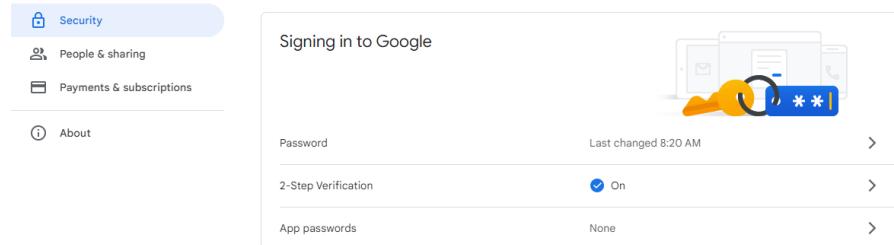




4. Tunggu proses berjalan, dan jika terjadi *error*, ikuti langkah selanjutnya



5. Adapun *error* terjadi dikarenakan kita menggunakan *protocol smtp* untuk mengirim *email*, dan *protocol* ini adalah sebuah *protocol* pihak ketiga. Oleh karena itu, untuk membuat *email* dapat terkirim dengan berhasil, lakukan langkah berikut untuk memberikan akses *email* kepada pihak ketiga dengan cara pilih akun gmail anda, kemudian pilih *Manage your Google Account*
6. Masuk ke bagian menu *Security*, kemudian scroll pada bagian *Signin in to Google*, pada bagian tersebut aktifkan *2-Step Verification* dengan cara menginput nomor *handphone* yang *valid*, setelah berhasil, maka akan terdapat menu *App passwords*, buka menu tersebut



7. Pada bagian *Select App*, pilih *Other (Custom name)*

The screenshot shows the 'App passwords' interface. At the top, a message says 'You don't have any app passwords.' Below it, a section titled 'Select the app and device you want to generate the app password for.' contains two dropdown menus: 'Select app' and 'Select device'. The 'Select app' dropdown has several options: Mail, Calendar, Contacts, YouTube, and 'Other (Custom name)', which is highlighted with a gray background. To the right of the dropdown is a 'GENERATE' button.

8. Isikan dengan Katalon Studio atau nama yang anda inginkan. Kemudian klik *Generate*

The screenshot shows the same 'App passwords' interface. The 'Select app' dropdown now has 'Katalon Studio' typed into it. The 'Select device' dropdown is empty. A blue 'GENERATE' button is visible on the right.

9. Kemudian anda akan diberikan *password* otomatis, kemudian klik *DONE*

10. Secara otomatis muncul daftar *App Password*

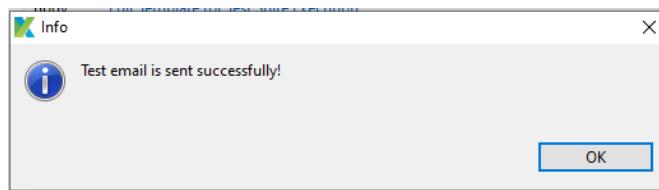
The screenshot shows the 'Your app passwords' page. It displays a table with one row, showing a generated password for 'Katalon Studio' created at '9:02 AM'. The table has columns for 'Name', 'Created', and 'Last used'. There is also a trash icon next to the entry. Below the table, there is a section to 'Select the app and device you want to generate the app password for.', with 'Select app' and 'Select device' dropdowns and a 'GENERATE' button.

Name	Created	Last used
Katalon Studio	9:02 AM	-

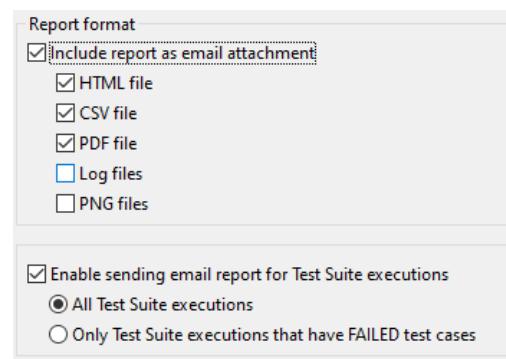
11. Copy-kan *password* sebelumnya, dan simpan pada bagian kolom *Password*, kemudian lakukan *Send Test Email* ulang.

Jika gagal menggunakan *port* dan *protocol* di atas, coba ganti ke *port* dan *protocol* seperti gambar di bawah

12. Maka *email* akan berhasil dikirimkan. Jika gagal, ganti *Port* yang lainnya



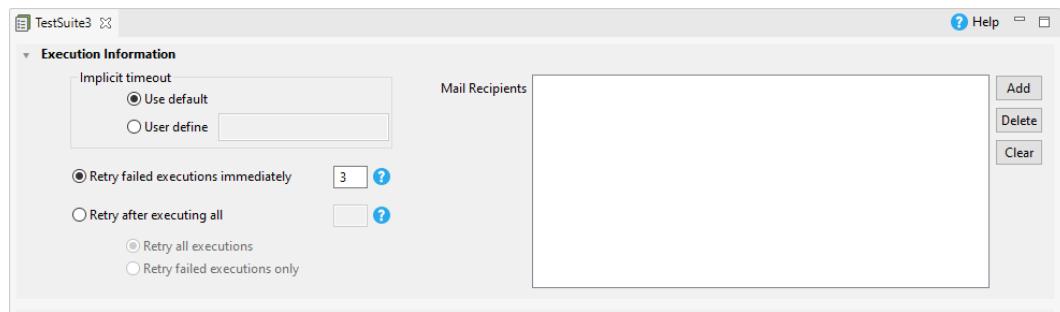
13. Jika tidak terdapat pada Kotak Masuk, kemungkinan akan dianggap sebagai *spam*
14. Langkah berikutnya, atur pada bagian *Report format* dengan berikut, dan klik *Apply* setelah melakukan konfigurasi



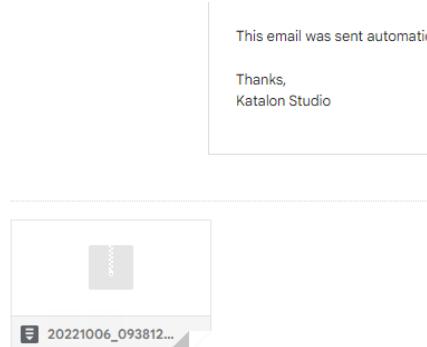
15. Adapun untuk mengubah *template* dari *email*, anda dapat melakukannya pada bagian Email>Template>Test Suite atau Test Suite Collection.

The image shows the Katalon Studio interface. On the left is a sidebar with a tree view of project settings. Under 'Email > Template', 'Test Suite' is selected. To the right is a preview window titled 'Test Suite'. It shows a header with the Katalon logo and 'Test S'. The body of the email starts with 'Dear Sir/Madam,' and a message: 'Your test suite has just finished its execution. He'. Below this is a table with two rows: 'Host Name' and '\${hostName}' in the same row, and 'Operating System' and '\${os}' in the next row.

16. Setelah melakukan konfigurasi *email*, selanjutnya buka file *Test Suite* anda, lakukan *expand* terhadap *Execution Information*. Pada jendela tersebut terdapat informasi mengenai *Mail Recipients*. Masukkan daftar nama *email* yang akan menerima pemberitahuan mengenai tes yang dilakukan



Jika sebelumnya anda mencentang bagian *include report as email attachment*, maka pada notifikasi *email* anda akan mendapat file berupa seperti .csv, .html, .pdf, dan lain sebagainya.

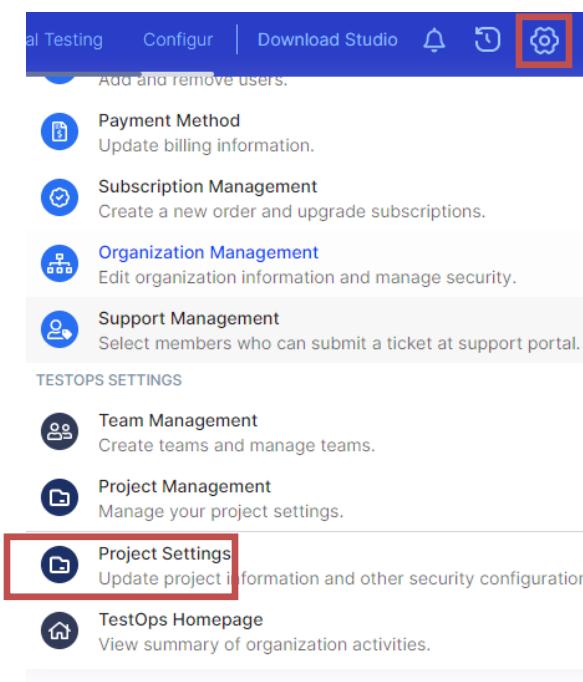


Setelah mempelajari modul ini, mahasiswa diharapkan sudah dapat melakukan pengujian beserta mengirim *email* hasil pengujian dengan berhasil.

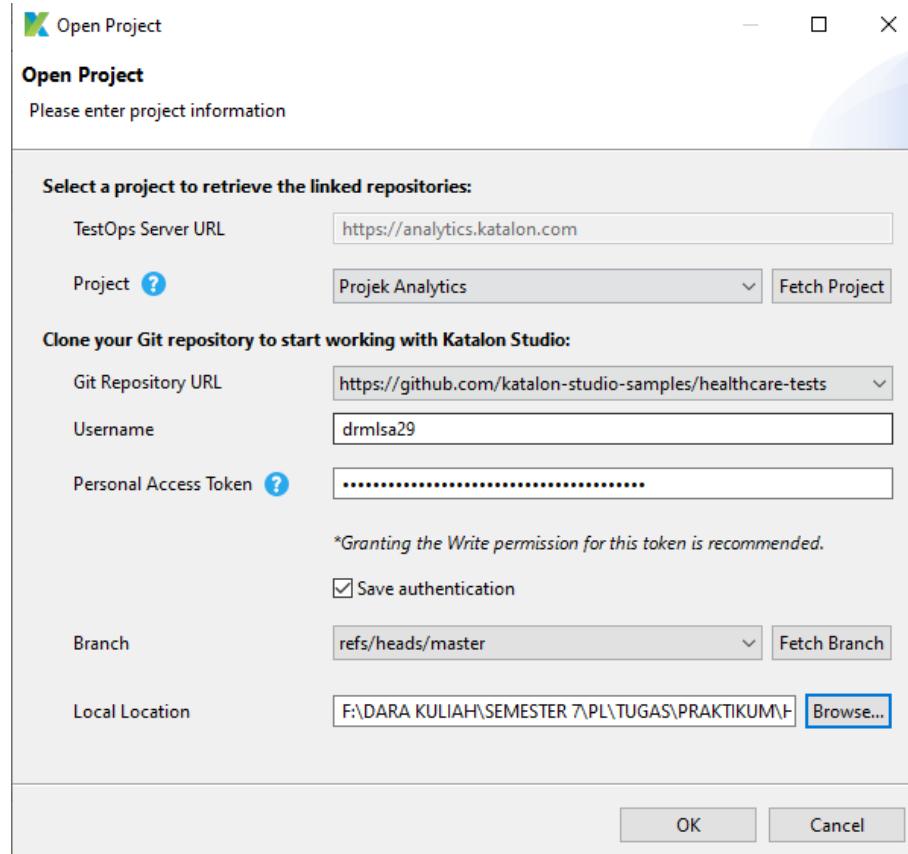
5.4 HOW TO USE KATALON ANALYTICS STEP BY STEP

Pada modul ini kita akan mempelajari bagaimana cara menggunakan katalon analytics. Katalon analytic merupakan suatu proses yang berfungsi untuk melihat, menyimpan, dan menganalisis hasil atau laporan dari pengujian yang dilakukan pada katalon. Berikut ini merupakan tahapan menggunakan katalon analytics.

1. Langkah pertama yang kita lakukan adalah, kita bisa membuat projek baru untuk katalon. Projek baru dapat dibuat dengan masuk ke web katalon, kemudian klik setting, lalu pilih new projek, dan projek baru pun telah dibuat.



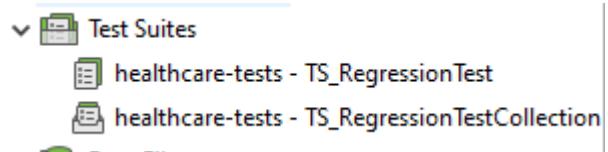
2. Selanjutnya, buka aplikasi katalon kemudian klik file lalu open projek. Masukkan username dan token dari akun github anda, lalu pada bagian projek pilih nama projek yang sudah dibuat pada web katalon analytics sebelumnya, pada kasus ini saya membuat projek dengan nama projek analytics. Pada proses ini kita sudah berhasil menghubungkan langsung test projek yang kita lakukan di aplikasi katalon ke web katalon analytics.



3. Jika kita lihat pada halaman test activities pada web katalon analytics, belum ada test activities apapun yang terekam karena memang belum ada pengujian yang dilakukan.

The screenshot shows the 'Projek Analytics' dashboard. The top navigation bar includes 'Dashboard', 'Planning', 'Test Management', 'Test Execution', 'Reports', 'Visual Testing', 'Configur', 'Download Studio', and a refresh icon. The main area has a 'Dashboard' header. Under 'Test Activities', there are two sections: 'Total Test Runs' (Last updated: Oct 15, 00:02) and 'Execution Result' (Last updated: Oct 15, 00:02). Both sections show 'There is no data to display.' Under 'Release Readiness', there is a section with 'There is no data to display. Please [create release](#) to view.' and a 'View all' button. A blue button at the top right says 'Enable Jira Integration'.

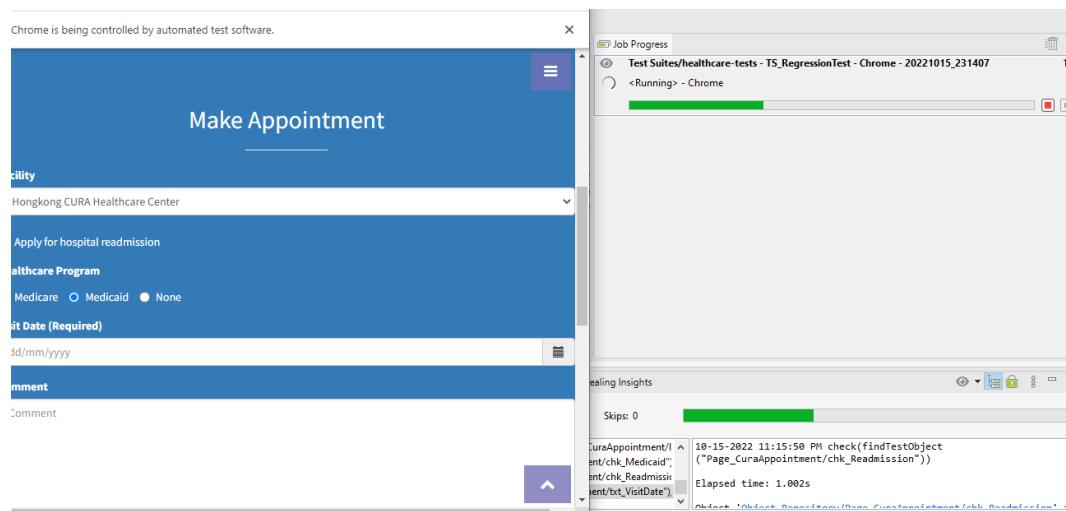
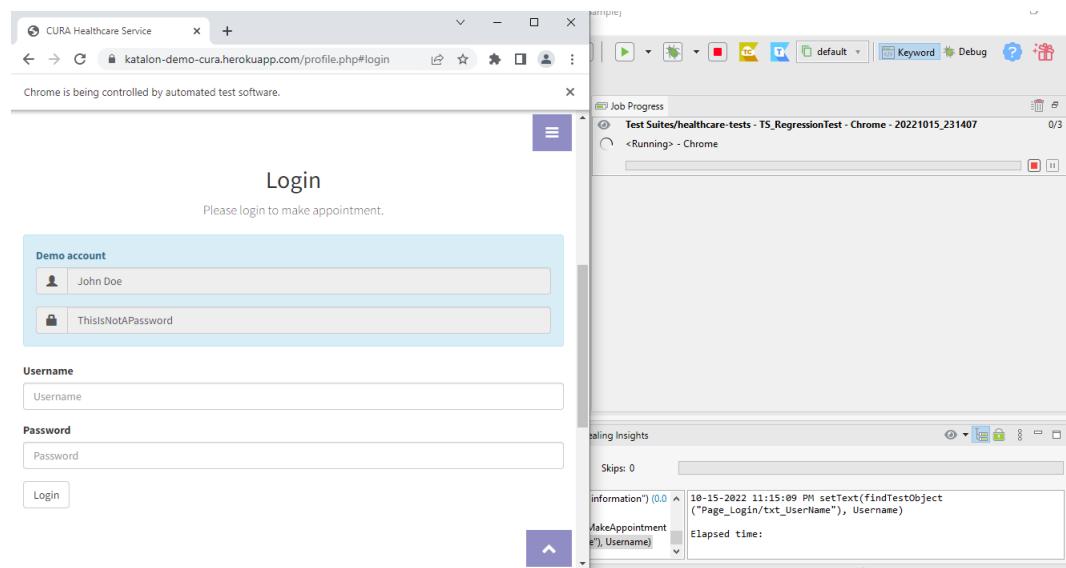
4. Selanjutnya kita akan menjalankan test suit sebagai bentuk percobaannya. Test suit ini merupakan test case yang sudah disediakan oleh katalon sebagai sampel pengujian. Untuk kasus ini saya memilih test case TR_RegressioonTest.



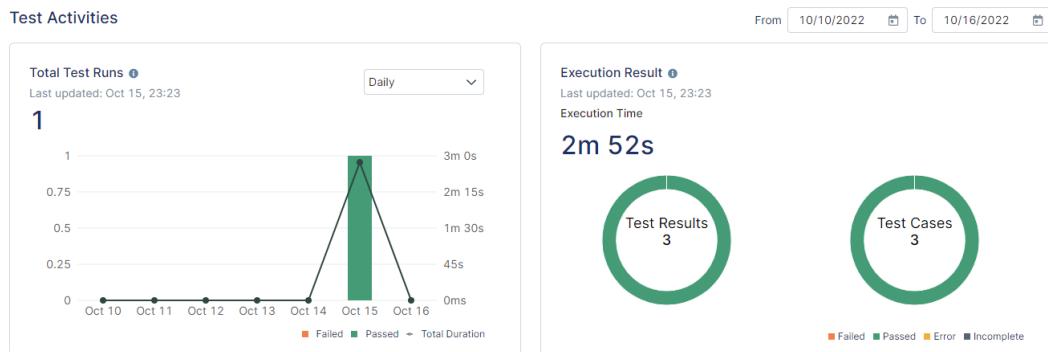
Seperti yang dapat dilihat pada gambar di bawah ini, pada test suit tersebut sudah ditambahkan beberapa test case.

		Add	Delete	Move Up	Move Down
Enter text to search...					
①	No.	ID			
	1	Test Cases/Main Test Cases/TC1_Verify Successful Login			
	2	Test Cases/Main Test Cases/TC2_Verify Successful Appointment			
	3	Test Cases/Main Test Cases/TC3_Visual Testing Example			

5. Jalankan test suit dengan browser pilihan anda, dan tunggu sampai proses eksekusi selesai. Berikut ini tampilan test suit yang sedang dieksekusi.



- Setelah proses eksekusi selesai, kita bisa melihat hasil report pengujian pada web analytics katalon. Terlihat pada test activities sudah ada 1 proses pengujian yang sudah dilakukan.



Dapat dilihat seperti pada gambar di atas, hasil detail report menunjukkan 1 passed, dan 0 error untuk pengujian yang dilakukan.

MODUL 9

API TESTING

A. Tujuan

1. Mengetahui apa itu API
2. Mengetahui bagaimana cara API Testing pada katalon
3. Mengetahui bagaimana cara mengirim nilai dari satu api ke api lainnya dengan xml.
4. Mengetahui bagaimana cara mengirim nilai dari satu api ke api lainnya dengan json.

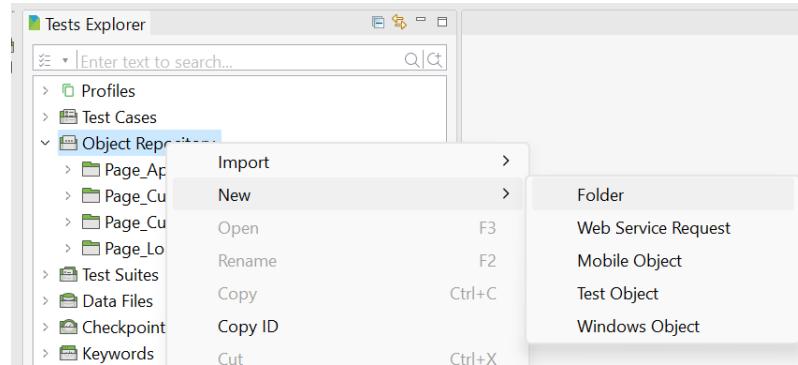
B. Dasar Teori

Api merupakan aplikasi yang memungkinkan para developer untuk mengintegrasikan bagian aplikasi dengan bagian aplikasi lainnya.

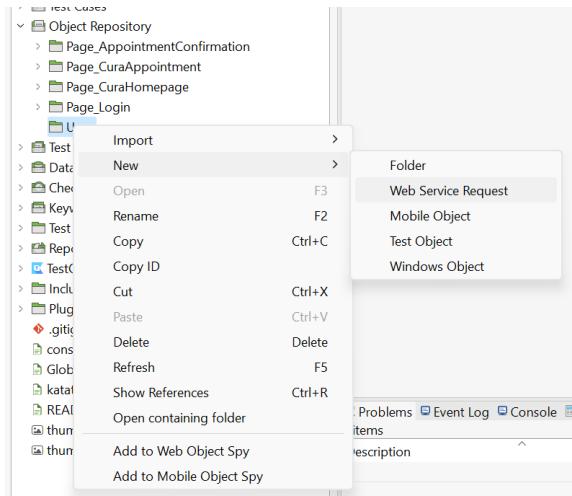
9.1. HOW TO DO API TESTING WITH KATALON

Katalon juga dapat digunakan sebagai aplikasi untuk melakukan pengujian terhadap API. Berikut langkah-langkah yang dapat dilakukan dalam menguji API pada katalon.

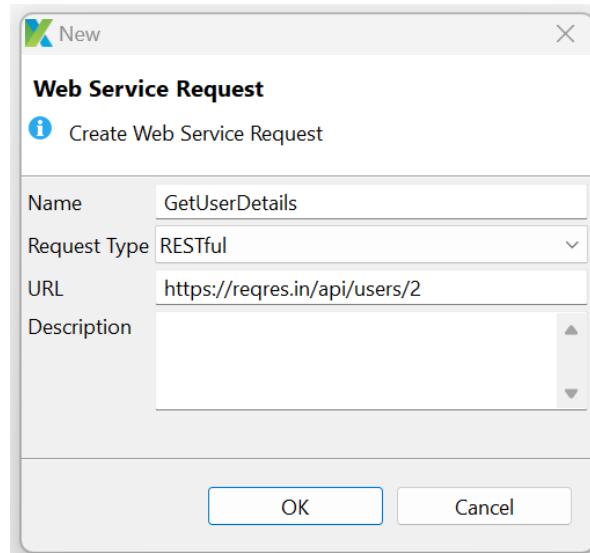
1. Buka project pada katalon, pada bagian Test Explorer klik kanan pada folder Object Repository untuk membuat folder baru.



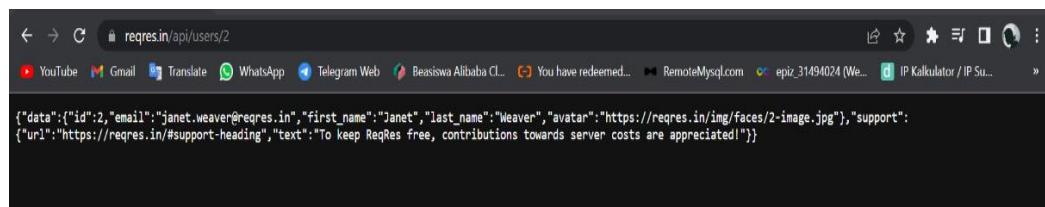
2. Setelah folder berhasil dibuat klik kanan, pilih web service Request



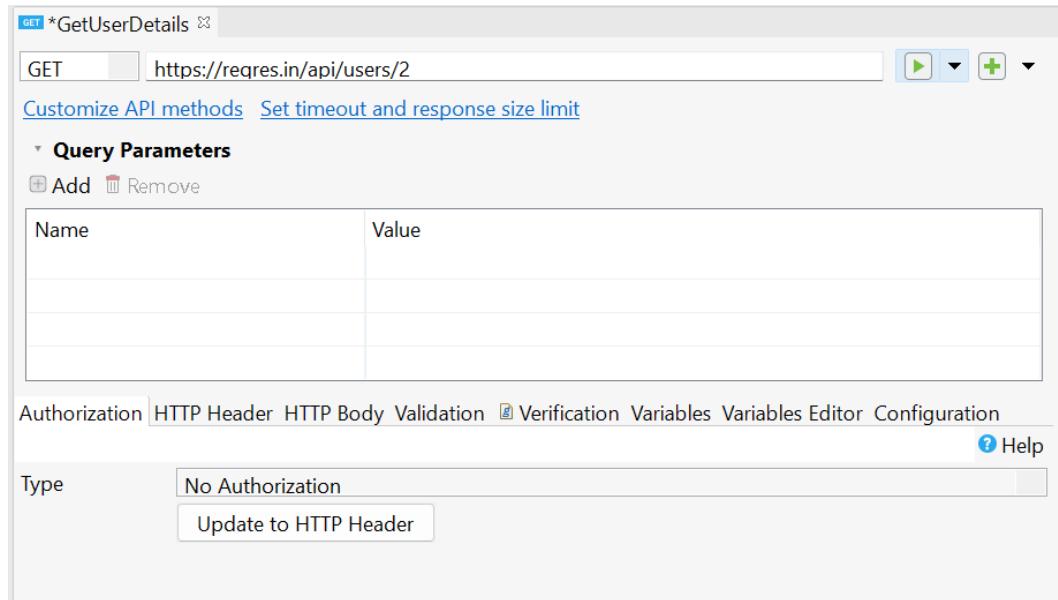
3. Kemudian isikan name untuk file yang akan dibuat, pilih RESTful sebagai request type, dan masukkan URL API, klik Ok.



4. URL API tersebut jika dibuka melalui web akan tampil seperti berikut



5. Berikut tampilan file yang dibuat, selanjutnya klik icon play.



The screenshot shows a REST client window with the following details:

- Method: GET
- Endpoint: https://reqres.in/api/users/2
- Query Parameters: An empty table with columns "Name" and "Value".
- Authorization: Type: No Authorization

6. Berikut tampilan hasilnya, yaitu GET user dengan id 2 berhasil dilakukan, ditandai dari Responsenya yaitu status 200 OK

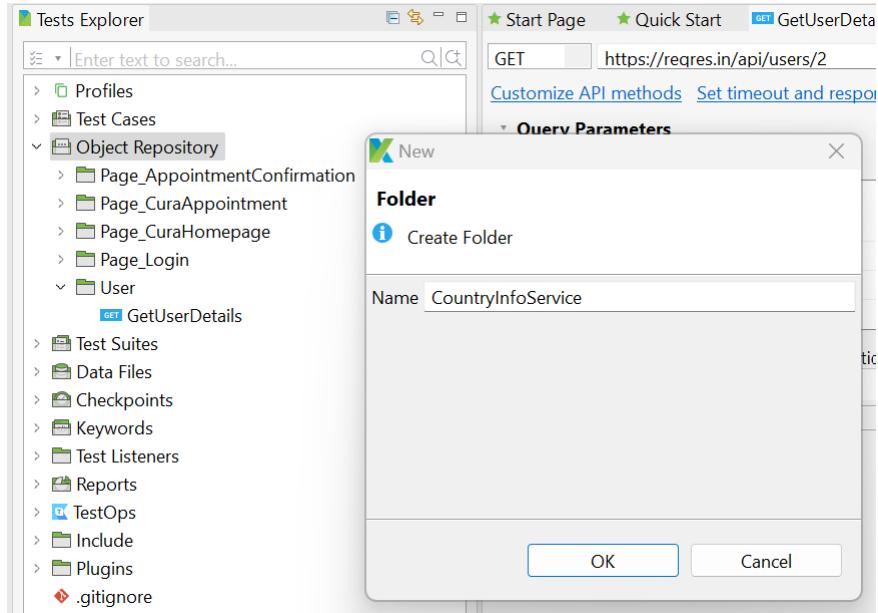


The screenshot shows a JSON response viewer with the following details:

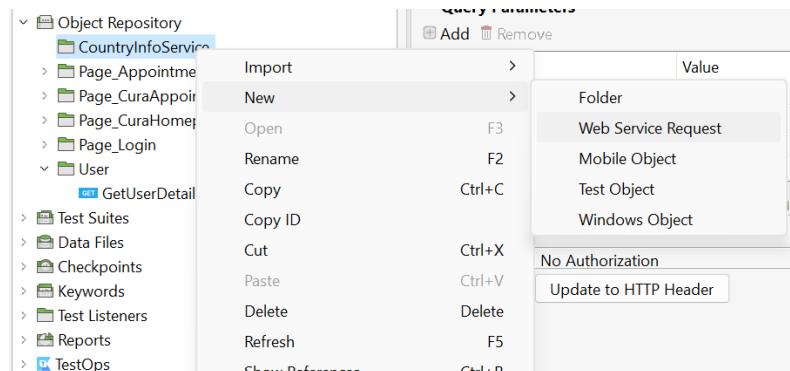
- Status: 200 OK
- Elapsed: 1131 ms
- Size: 644 bytes
- Body (selected tab):
 - pretty (radio button selected)
 - raw
 - preview
- JSON content:

```
1 {  
2   "data":{  
3     "id":2,  
4  
5     "email":"janet.weaver@reqres.  
6     .in",  
7     "first_name":"Janet",  
8     "last_name":"Weaver",  
9  
10    "avatar":"https://reqres.in/  
img/faces/2-image.jpg"  
11  },  
12  "support":{  
13    "url":"https://reqres.in/#su  
14  }
```
- Instructions at the bottom: "Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly."
- Format selection: JSON (radio button selected), XML, HTML, JavaScript, Wrap Lin

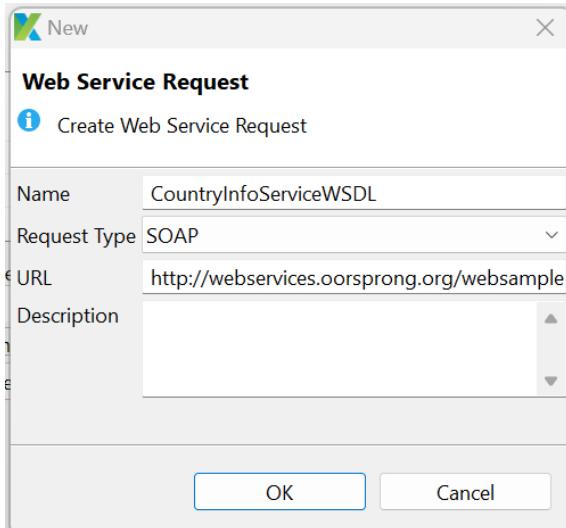
7. Selanjutnya buat folder baru dengan nama “CountryInfoService” pada Object Repository



8. Klik kanan pada folder “CountryInfoService”, pilih new dan klik Web Service Request



9. Kemudian isikan name sebagai berikut, pilih SOAP sebagai Request Type dan tempelkan url berikut.



10. Untuk mendapatkan URL diatas dapat di searching di google, buka website berikut.

Sekitar 261.000 hasil (0,48 detik)

<http://webservices.oorsprong.org/> > CountryInfoService ▾

Service Description

Returns the name of the capital city for the passed **country** code Returns the currency ISO code and name for the passed **country** ISO code Returns a link to a ...

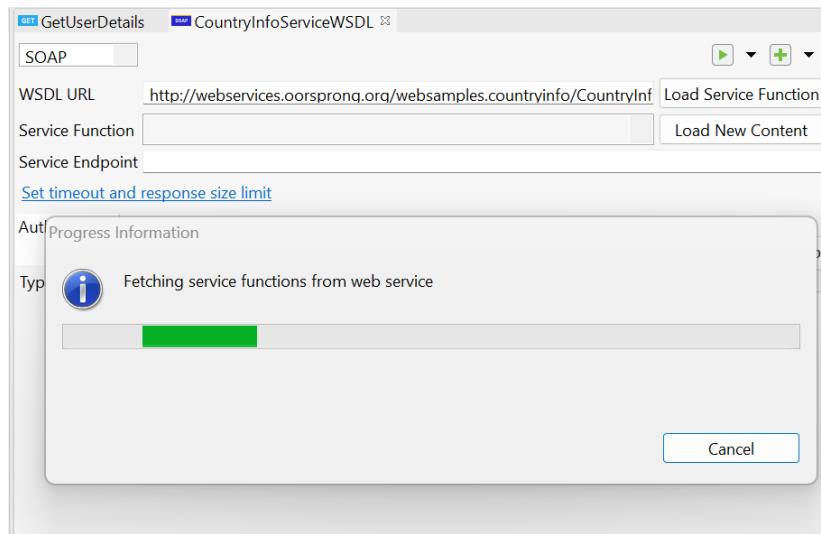
11. Berikut tampilan website, lalu copy link addressnya.

```

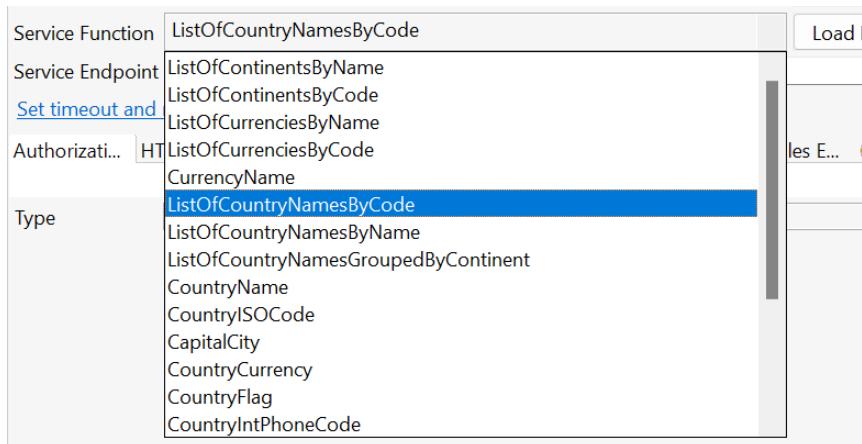
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:x="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:tns="http://www.oorsprong.org/websamples.countryinfo" name="CountryInfoService" targetNamespace="http://www.oorsprong.org/websamples.countryinfo">
  <types>
    <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.oorsprong.org/websamples.countryinfo">
      <xsd:complexType name="tContinent">
        <xsd:sequence>
          <xsd:element name="sCode" type="xs:string"/>
          <xsd:element name="sName" type="xs:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <xsd:complexType name="tCurrency">
    <xsd:sequence>
      <xsd:element name="sISOCode" type="xs:string"/>
      <xsd:element name="sName" type="xs:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="tCountryCodeAndName">
    <xsd:sequence>
      <xsd:element name="sISOCode" type="xs:string"/>
      <xsd:element name="sName" type="xs:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CountryCodeAndNameGroupedByContinent">
    <xsd:sequence>
      <xsd:element name="Continent" type="tns:tContinent"/>
      <xsd:element name="CountryCodeAndNames" type="tns:ArrayOfCountryCodesAndName"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="tCountryInfo">
    <xsd:sequence>
      <xsd:element name="sISOCode" type="xs:string"/>
      <xsd:element name="sName" type="xs:string"/>
      <xsd:element name="sCapitalCity" type="xs:string"/>
      <xsd:element name="sPhonecode" type="xs:string"/>
      <xsd:element name="sContinentCode" type="xs:string"/>
      <xsd:element name="sCurrencyISOCode" type="xs:string"/>
    </xsd:sequence>
  </xsd:complexType>

```

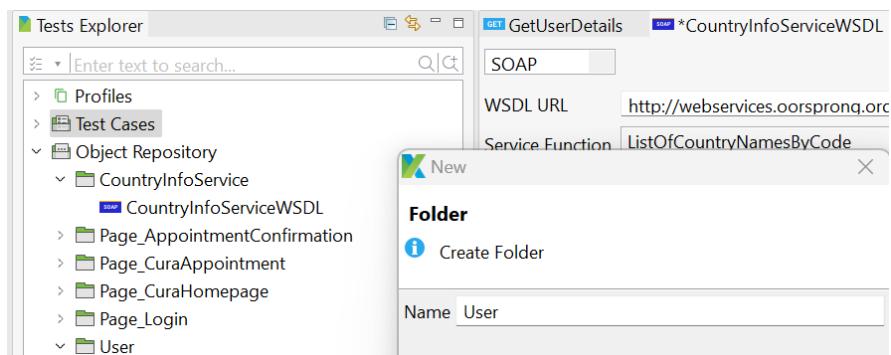
12. Berikut Tampilan File yang telah kita buat, klik “Load Service Function”.



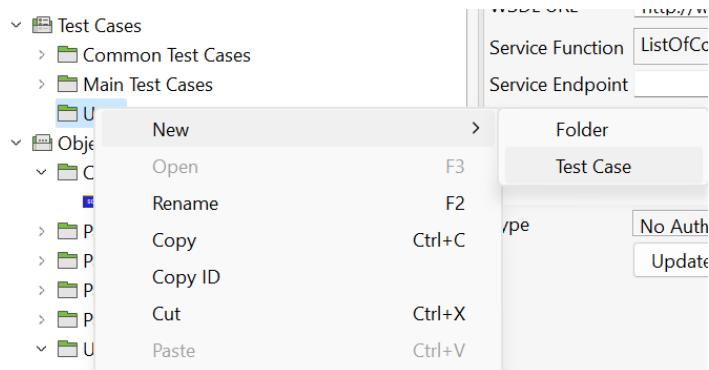
13. Pada Service Function pilih “ListOfCountryNamesByCode”



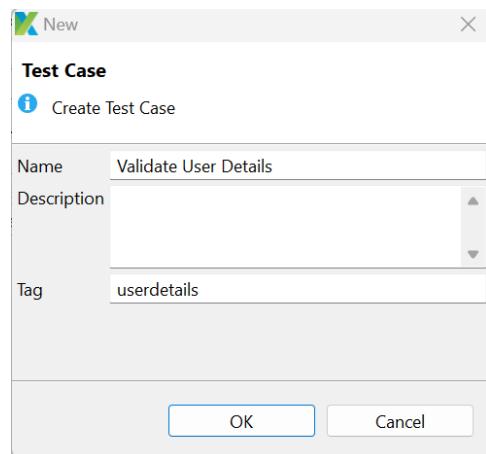
14. Langkah selanjutnya pada Test Explorer, buat folder “User” didalam Test Case



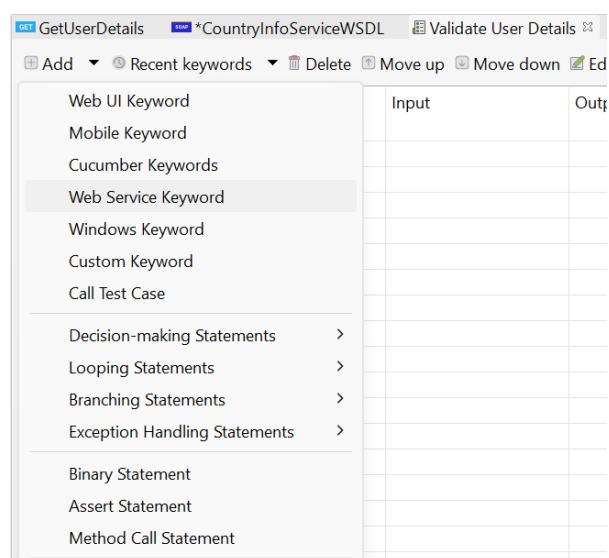
15. Pada folder “User” klik kanan pilih New dan klik Test Case



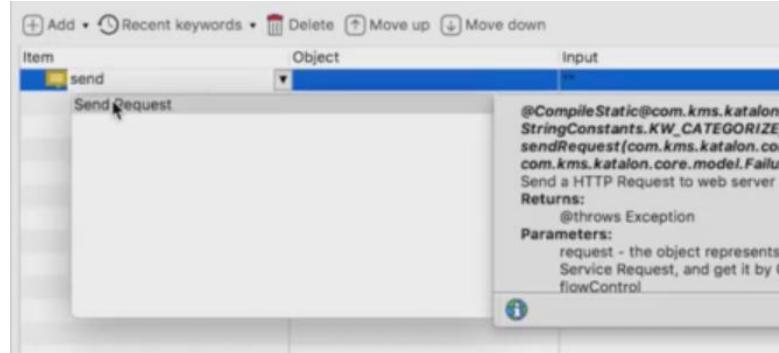
16. Isikan form seperti berikut, klik Ok



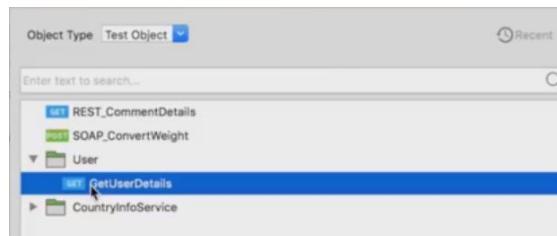
17. Selanjutnya klik dropdown Add pilih Web Service Keyword



18. Gantikan Namanya menjadi “Send Request”



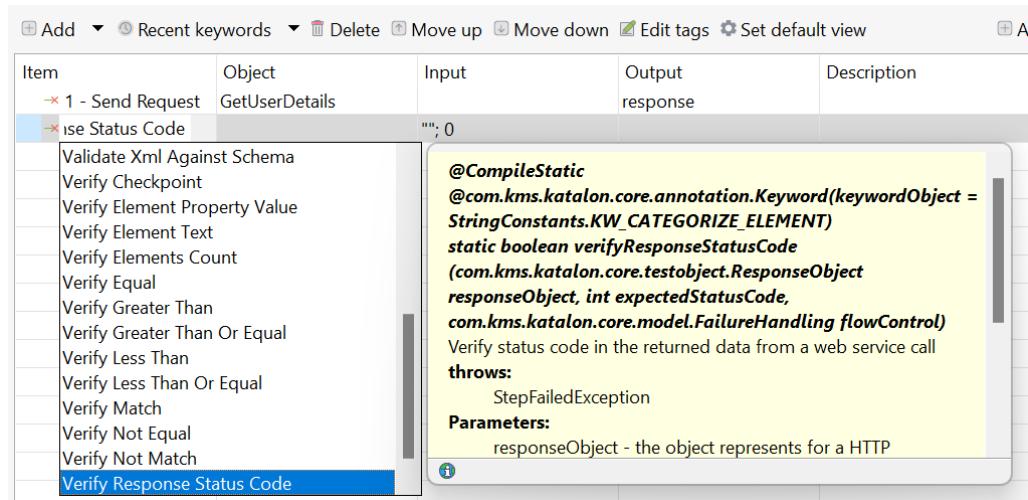
19. Pada Object yang berisi null, klik lalu pilih “GetUserDetails”



20. Pada Output isikan “response”

Item	Object	Input	Output	Description
* 1 - Send Request	GetUserDetails		response	

21. Tambahkan “Verify Response Status Code”



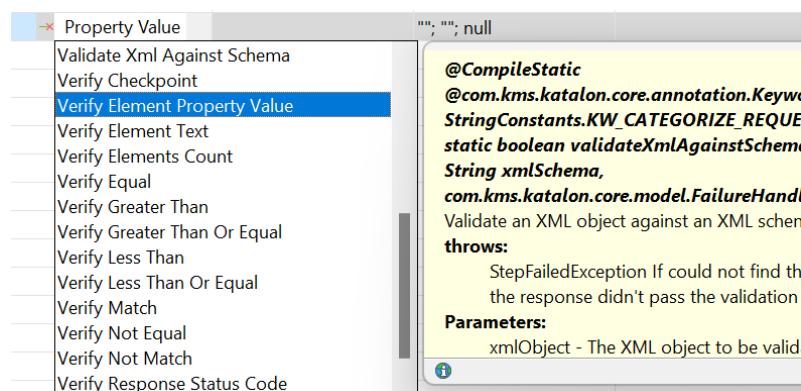
22. Ubah input dari “Verify Response Status Code” seperti berikut

The screenshot shows the 'Input' dialog box with two rows of parameters:

No.	Param Name	Param Type	Value Type	Value
1	responseObject	ResponseObject	Variable	response
2	expectedStatusCode	Integer	Number	200

Buttons at the bottom: OK and Cancel.

23. Kemudian tambahkan “Verify Element Property Value”



24. Ubah input dari “Verify Element Property Value” seperti berikut

The screenshot shows the 'Input' dialog box with three rows of parameters:

No.	Param Na...	Param Ty...	Value Type	Value
1	response	ResponseO	Variable	response
2	locator	String	String	"data.first_name"
3	value	Object	String	"Janet"

Buttons at the bottom: OK and Cancel.

25. Sehingga menjadi seperti berikut, selanjutnya klik icon play

The screenshot shows the Katalon Studio interface with the 'Test Cases/User/Validate User Details' tab selected. The test step 'Validate User Details' is expanded, showing three sub-steps:

Item	Object	Input	Output	Description
-> 1 - Send Request	GetUserDetails			response
-> 2 - Verify Response Status Code		response: 200		
-> 3 - Verify Element Property Value		response: "data.first_name": "Janet"		

26. Berikut hasilnya, dapat dilihat lewat Log Viewer, hasilnya tidak ada errornya

The screenshot shows the Katalon Studio Log Viewer. The test run 'Test Cases/User/Validate User Details' was successful, with 1 pass and 0 errors. The log details show the steps and their execution times:

Step	Time	Description
1 - response = sendRequest(findTestObject("User/GetUserDetails"))	0.780s	HAR: C:\Users\LENOVO\AppData\Local\Temp\Katalon\Test Cases\User\Validate User Details\20221017
2 - verifyResponseStatus(response, 200)	0.048s	
3 - verifyElementPropertyValue(response, "data.first_name", "Janet")	0.267s	

27. Namu Ketika kita ganti inputan pada “Verify Element Property Value” seperti berikut

The screenshot shows the 'Input' dialog for the 'Verify Element Property Value' step. The parameters are updated as follows:

No.	Param Na...	Param Ty...	Value Type	Value
1	response	ResponseO	Variable	response
2	locator	String	String	"data.first_name"
3	value	Object	String	"Daffa"

28. Maka hasilnya akan error, seperti berikut

The screenshot shows the Katalon Studio Log Viewer. The test run failed with 1 failure and 0 errors. The error details indicate a 'StepFailedException' due to an expected element property value mismatch:

Step	Time	Message
1 - response = sendRequest(findTestObject("User/GetUserDetails"))	0.531s	For trouble shooting, please visit: https://docs.katalon.com/katalon-studio/docs/troubleshooting.html
2 - verifyResponseStatus(response, 200)	0.008s	
3 - verifyElementPropertyValue(response, "data.first_name", "Daffa")	0.089s	Test Cases/User/Validate User Details FAILED. Reason: com.kms.katalon.core.exception.StepFailedException: Expected element property value 'Daffa' is not equal with actual property value 'Janet' at com.kms.katalon.core.keyword.internal.KeywordMain.stepFailed (KeywordMain.groovy:58) at com.kms.katalon.core.webservice.keyword.builtin.VerifyElementPropertyValueKeyword.\$_verifyElementPropertyValue_closure1.doCall (VerifyElementPropertyValueKeyword.groovy:56)

29. Penyebab Error dikarenakan pada Object Repository terdapat folder User yang didalamnya memiliki file GetUserDetails yang berisi sebagai berikut. Oleh

karena itu inputan pada “Verify Element Property Value” hanya menerima “Janet” sebagai string selain itu akan error.

Response [HAR](#)

Status: **200 OK** Elapsed: **1505 ms** Size: **640 bytes**

[Body](#) [Header](#) [Verification Log](#) [Validation Log](#)

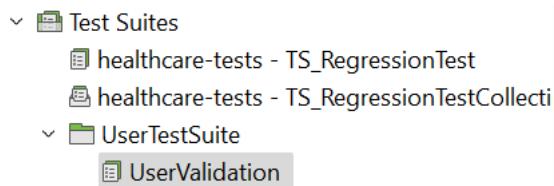
pretty raw preview

```
1  {
2   "data": {
3     "id": 2,
4
5       "email": "janet.weaver@reqres.in",
6       "first_name": "Janet",
7       "last_name": "Weaver",
8
9     "avatar": "https://reqres.in/img/faces/2-image.jpg"
10    },
11   "support": {
12     "url": "https://reqres.in/#support-link"
13   }
14 }
```

Select JSON or XML response data and press
Ctrl/Command + K to add verification scripts directly.

JSON XML HTML JavaScript Wrap Lin

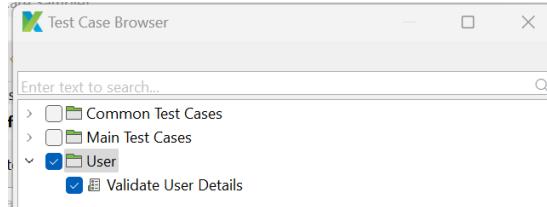
30. Buatkan test suite didalam folder UserTestSuite berikan nam “UserValidation”



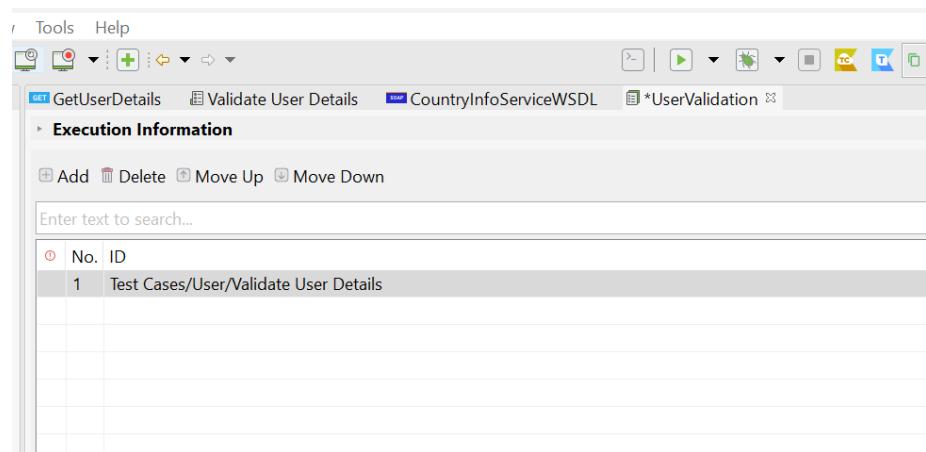
31. Berikut tampilannya, klik add

Test Suite: UserValidation		
No.	Description	Run
1		

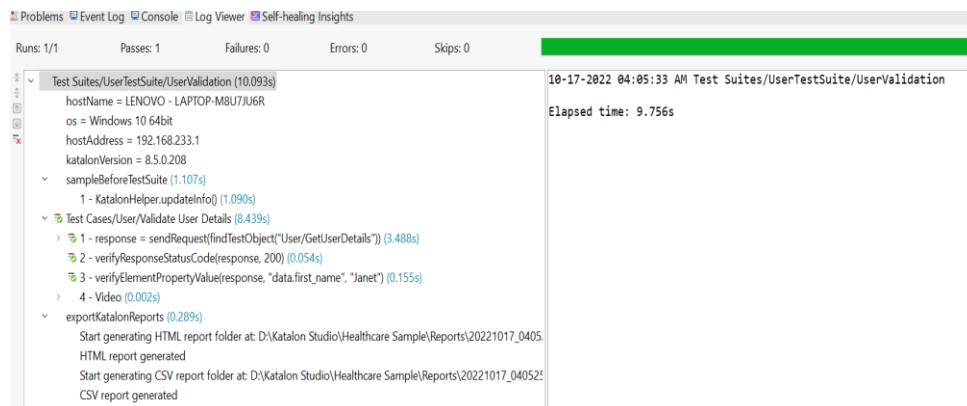
32. Centang Validate User Details



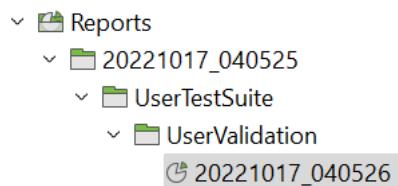
33. Berikut setelah di add, klik icon play



34. Berikut hasil dari Log Viewer



35. Selanjutnya kita buka Reports untuk melihat hasil pengujian



36. Berikut hasil report dari pengujian file json yang telah dilakukan

Test Cases Table

Passed Failed Error Incomplete Skipped Export report Katalon TestOps

Search here...

No.	Name	Resolution
1	Validate User Details (8.439s)	

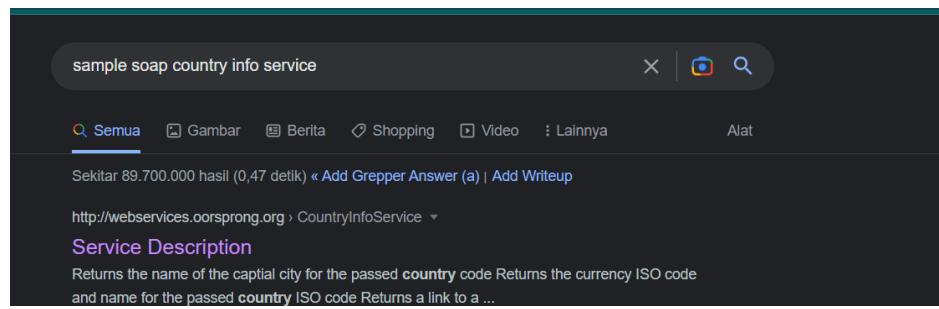
Summary Execution Settings Execution Environment

Test Suite ID	Test Suites/UserTestSuite/UserValidation
Host name	LENOVO - LAPTOP-M8U7JU6R
Katalon version	8.5.0.208
Start	2022-10-17 04:05:33
Elapsed	9.756s
Total TC	1
Passed	1
Error	0
Incomplete	0
Failed	0
Skip	0

9.2 API CHAINING | SOAP-XML | HOW TO SEND VALUE FROM ONE API TO OTHER

Katalon juga dapat melakukan chaining API. Chaining API adalah suatu proses untuk mengekstrak nilai dari respon suatu API dan memberikan nilai ke permintaan API berikutnya. Pada tahapan ini kita akan melakukan chaining API pada API SOAP yang memiliki respon XML. Berikut tahapannya.

1. Cari satu contoh SOAP API, seperti gambar di bawah ini.



sample soap country info service

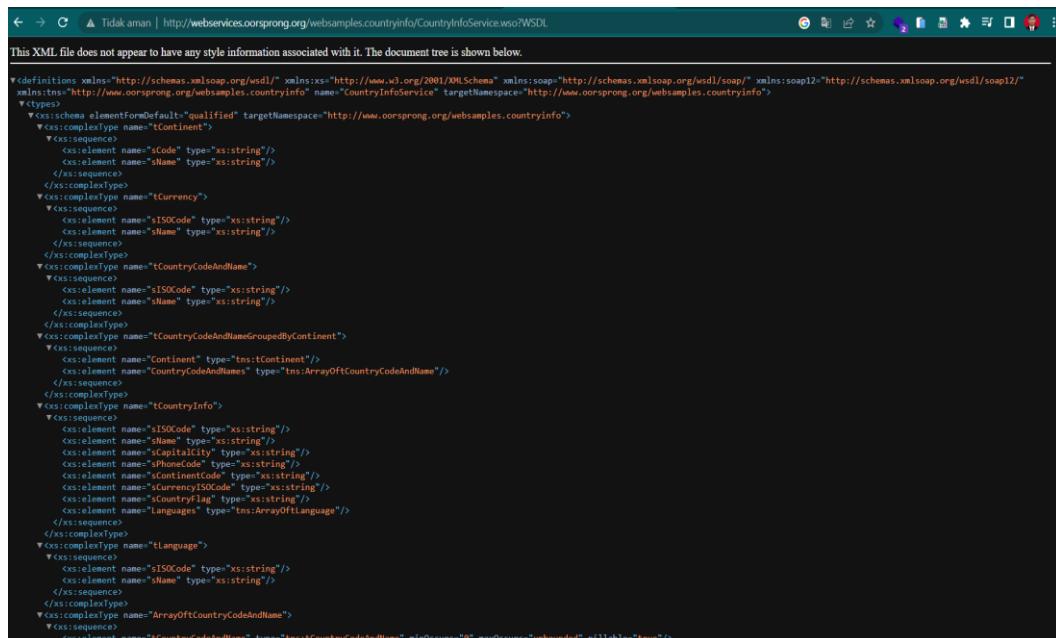
Sekitar 89.700.000 hasil (0,47 detik) « Add Grepper Answer (a) | Add Writeup

<http://webservices.orsprong.org> › CountryInfoService

Service Description

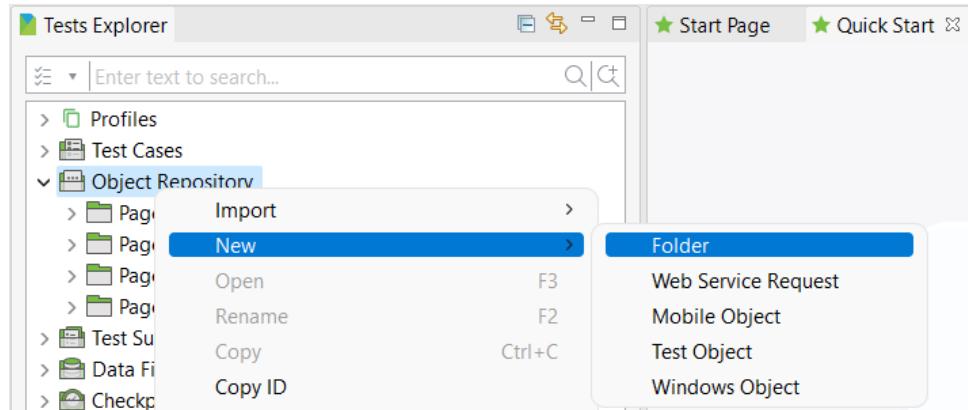
Returns the name of the capital city for the passed **country** code Returns the currency ISO code and name for the passed **country** ISO code Returns a link to a ...

2. Berikut merupakan contoh script soap api dalam format xml.

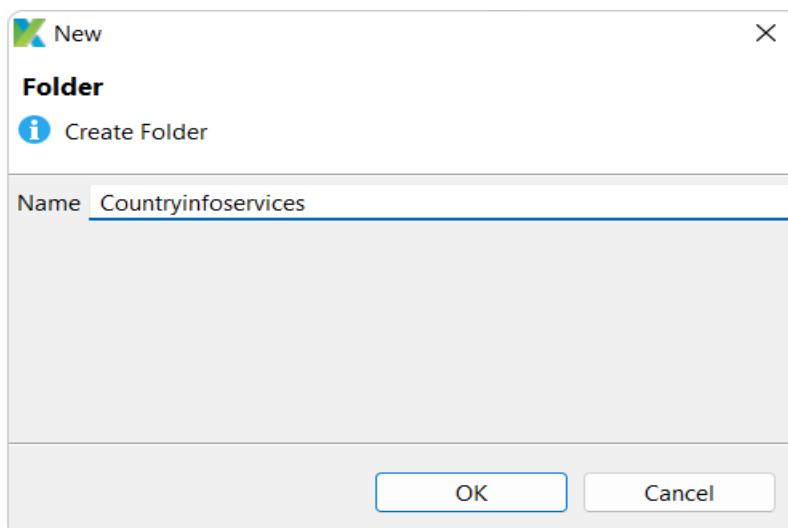


```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:tns="http://www.orsprong.org/websamples.countryinfo" name="CountryInfoService" targetNamespace="http://www.orsprong.org/websamples.countryinfo">
    <wsdl:types>
        <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.orsprong.org/websamples.countryinfo">
            <xsd:sequence>
                <xsd:element name="tContinent">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Code" type="xs:string"/>
                            <xsd:element name="Name" type="xs:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="CountryCodeAndName">
        <wsdl:part name="CountryCode" type="tns:tContinent"/>
        <wsdl:part name="Name" type="xs:string"/>
    </wsdl:message>
    <wsdl:message name="CountryCodeAndNameGroupedByContinent">
        <wsdl:part name="Continent" type="tns:tContinent"/>
        <wsdl:part name="CountryCodeAndName" type="tns:ArrayOfCountryCodeAndName"/>
    </wsdl:message>
    <wsdl:portType name="CountryInfoPortType">
        <wsdl:operation name="GetContinent">
            <wsdl:input message="tns:CountryCodeAndName"/>
            <wsdl:output message="tns:CountryCodeAndNameGroupedByContinent"/>
        </wsdl:operation>
        <wsdl:operation name="GetCountryInfo">
            <wsdl:input message="tns:CountryCodeAndNameGroupedByContinent"/>
            <wsdl:output message="tns:ArrayOfCountryInfo"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="CountryInfoSoapBinding" type="tns:CountryInfoPortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetContinent">
            <soap:operation soapAction="http://www.orsprong.org/websamples.countryinfo/GetContinent"/>
            <wsdl:input>
                <soap:body/>
            </wsdl:input>
            <wsdl:output>
                <soap:body/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="GetCountryInfo">
            <soap:operation soapAction="http://www.orsprong.org/websamples.countryinfo/GetCountryInfo"/>
            <wsdl:input>
                <soap:body/>
            </wsdl:input>
            <wsdl:output>
                <soap:body/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CountryInfoService">
        <wsdl:port name="CountryInfoPort" binding="tns:CountryInfoSoapBinding">
            <soap:address location="http://webservices.orsprong.org/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

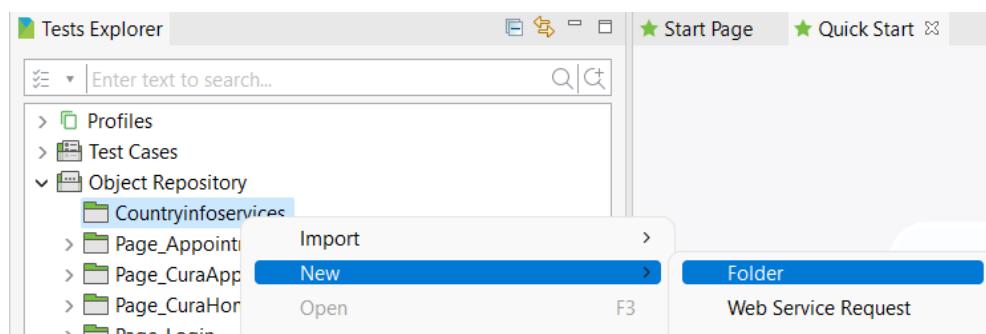
3. Selanjutnya buat 1 folder baru.



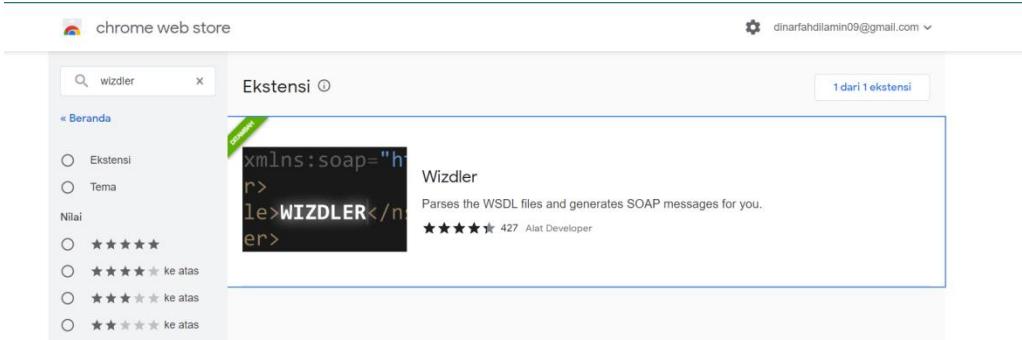
4. Kemudian, beri nama Countrinfoservices, lalu klik ok.



5. Pada folder Countrinfoservice, buat 1 web service request.



6. Karena kita akan mengambil beberapa permintaan dari script xml tadi, maka tambahkan extensi wizdler pada chrome. Extensi ini berguna untuk melewati dokumen vestal seperti forma, sehingga permintaan yang di request tadi dapat diterima.



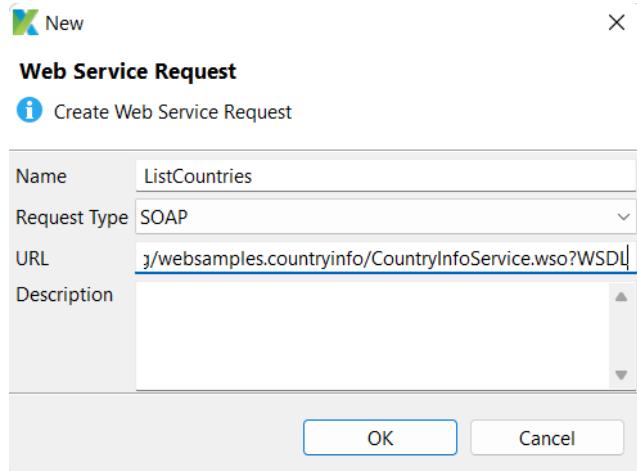
7. Jika extensi sudah ditambahkan, maka akan muncul ikon seperti gambar dibawah ini.



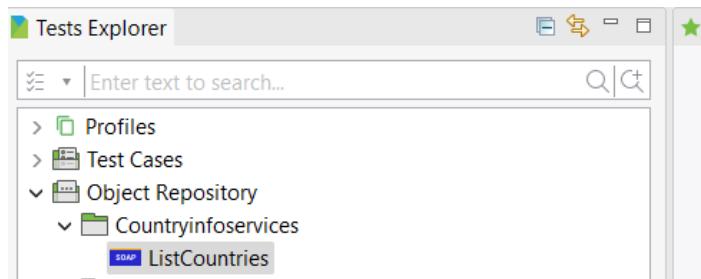
8. Berikut ini merupakan list request yang tersedia.

9. Selanjutnya kita akan mengambil request dari ListOfCountryNamesByName, berikut tampilannya.

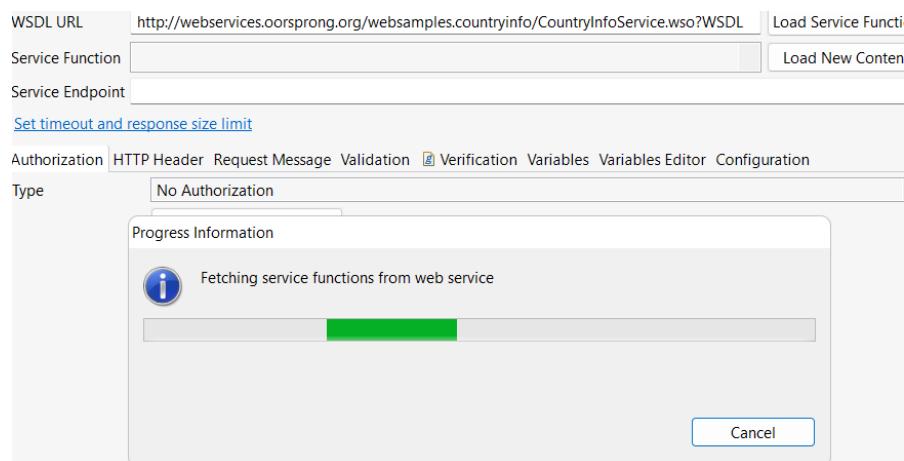
10. Buat web service request dengan nama ListCountries, dengan pilihan requestType adalah SOAP, dan masukan URL dari sample SOAP API tadi.



11. Dapat kita lihat web service request sudah terbentuk.



12. Masukkan service function, sebelumnya klik load service function terlebih dahulu untuk melakukan pencarian terhadap service function yang tersedia.



13. Pilih ListOfCountryNamesByName untuk service functionnya.

The screenshot shows a configuration interface with the following settings:

- WSDL URL: <http://webservices.orsprong.org/websamples.countryinfo/CountryInfoService.wsdl>
- Service Function: ListOfCountryNamesByName
- Service Endpoint:
 - ListOfContinentsByName
 - ListOfContinentsByCode
 - ListOfCurrenciesByName
 - ListOfCountryNamesByName** (highlighted in blue)
 - ListOfCurrenciesByCode
 - CurrencyName
 - ListOfCountryNamesByCode
 - ListOfCountryNamesByName** (highlighted in blue)
 - ListOfCountryNamesGroupedByContinent
 - CountryName
 - CountryISOCode
- Authorization: HTTP
- Type: ListOfCountryNamesByName

14. Kemudian pada request message copi script yang ada pada url ListOfCountryNamesByName tadi.

The screenshot shows a configuration interface with the following settings:

- Request Message tab is selected.
- Script content:

```
1 <Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
2   <Body>
3     <ListOfCountryNamesByName
4       xmlns="http://www.orsprong.org/websamples.countryinfo"/>
5   </Body>
</Envelope>
```

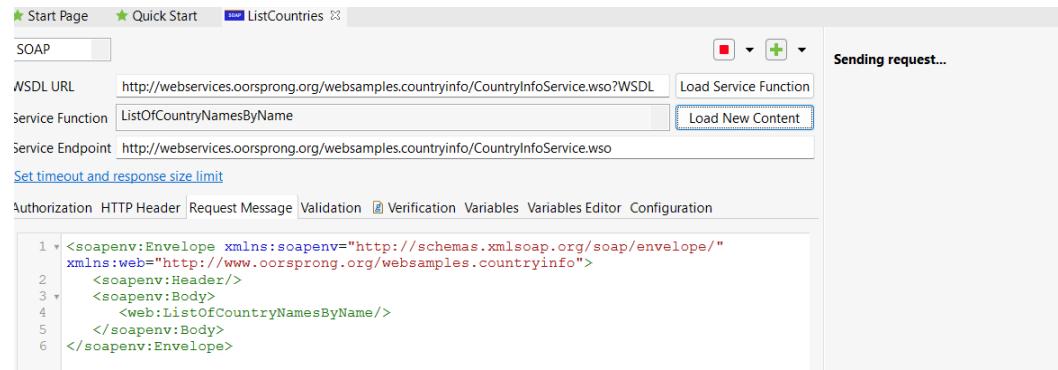
15. Kemudian save script.

The screenshot shows a configuration interface with the following settings:

- Service Endpoint: (empty)
- Set timeout and response size limit: (empty)
- Request Message tab is selected.
- Script content:

```
1 <Envelope xmlns="http://www.w3.org/2003/05/soap-envelope">
2   <Body>
3     <ListOfCountryNamesByName
4       xmlns="http://www.orsprong.org/websamples.countryinfo"/>
5   </Body>
</Envelope>
```
- A warning dialog box is displayed, asking "Do you want to save the changes?" with "OK" and "Cancel" buttons.

16. Selanjutnya jalankan script untuk melakukan request/permintaan.

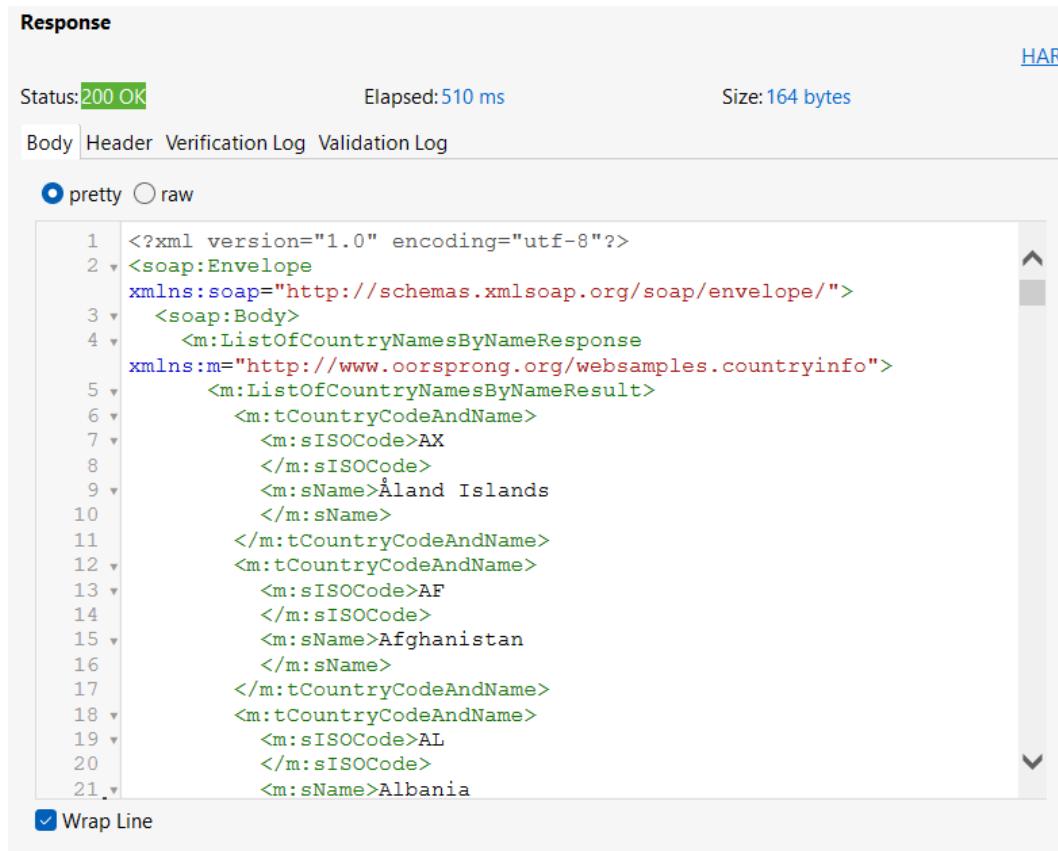


The screenshot shows the SoapUI interface with the following details:

- Top navigation bar: Start Page, Quick Start, ListCountries.
- SOAP tab is selected.
- WSSDL URL: <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL>
- Service Function: ListOfCountryNamesByName
- Service Endpoint: <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso>
- Request message content (Message 1):

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://www.oorsprong.org/websamples.countryinfo">
<soapenv:Header/>
<soapenv:Body>
<web:ListOfCountryNamesByName/>
</soapenv:Body>
</soapenv:Envelope>
```
- Status bar: Sending request...

17. Berikut ini hasil dari request yang diberikan, dimana berisi list nama-nama dari berbagai negara.

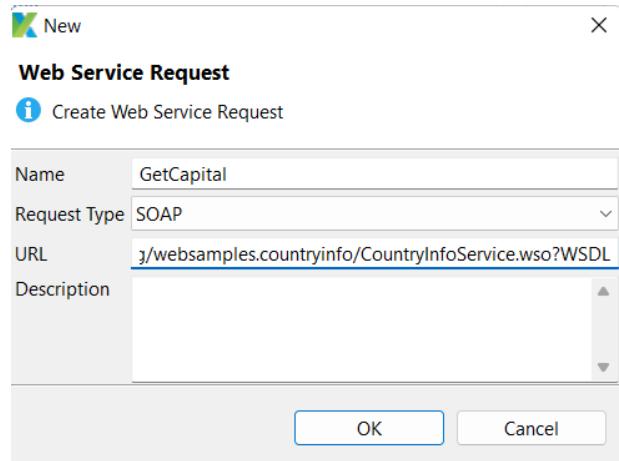


The screenshot shows the SoapUI interface with the following details:

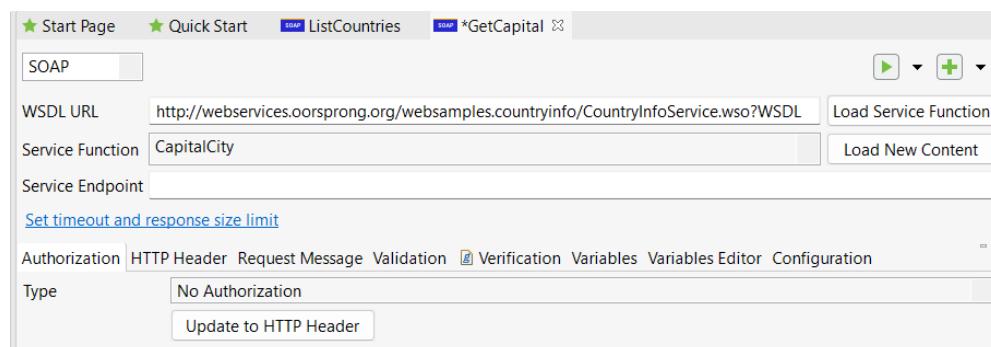
- Response tab is selected.
- Status: 200 OK
- Elapsed: 510 ms
- Size: 164 bytes
- Body tab is selected.
- Response content (Message 1):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <m:ListOfCountryNamesByNameResponse
            xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
            <m:ListOfCountryNamesByNameResult>
                <m:tCountryCodeAndName>
                    <m:sISOCode>AX
                    </m:sISOCode>
                    <m:sName>Åland Islands
                    </m:sName>
                </m:tCountryCodeAndName>
                <m:tCountryCodeAndName>
                    <m:sISOCode>AF
                    </m:sISOCode>
                    <m:sName>Afghanistan
                    </m:sName>
                </m:tCountryCodeAndName>
                <m:tCountryCodeAndName>
                    <m:sISOCode>AL
                    </m:sISOCode>
                    <m:sName>Albania
                    </m:sName>
                </m:tCountryCodeAndName>
            </m:ListOfCountryNamesByNameResult>
        </m:ListOfCountryNamesByNameResponse>
    </soap:Body>
</soap:Envelope>
```
- Wrap Line checkbox is checked.

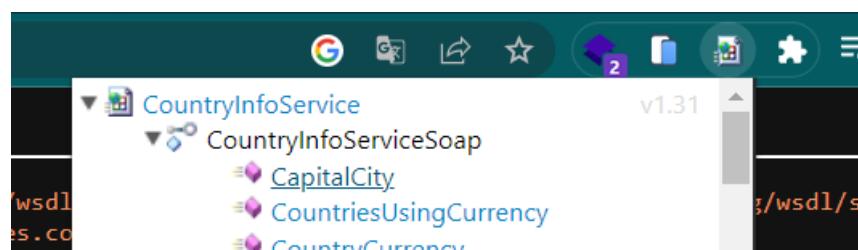
18. Buat web service request lainnya.



19. Pilih server functio nya CapitalCity.



20. Pada extansi wizdler, pilih CapitalCity.



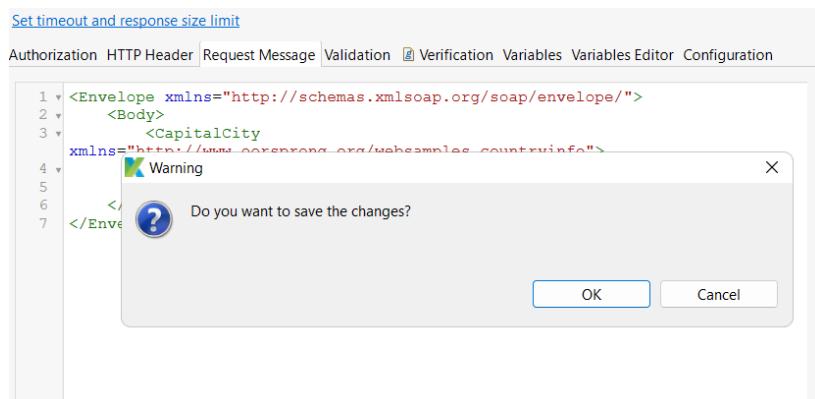
21. Berikut script dari CapitalCity.



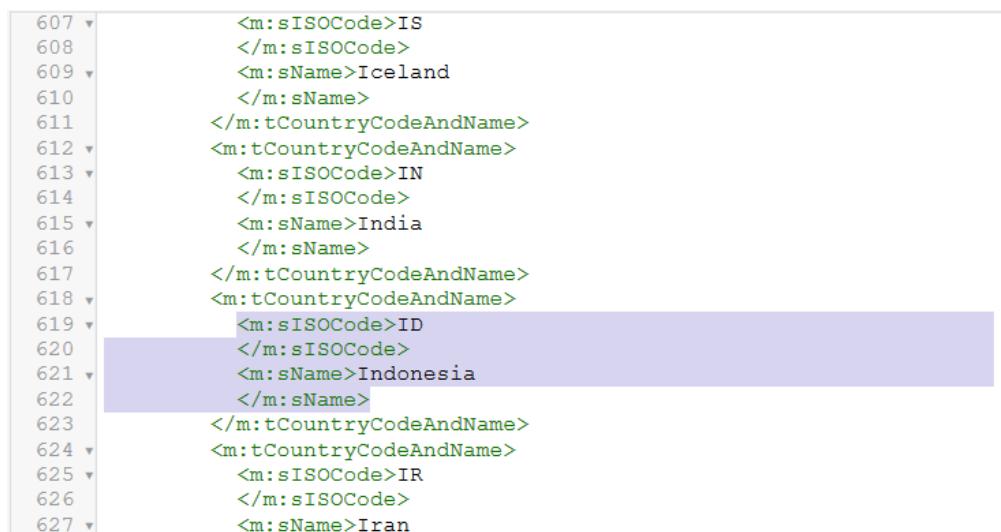
The screenshot shows a browser window with the address bar containing "Wizdler | chrome-extension://oebpmncolmhiapingjaagmapififiakb/editor.html#wsd". Below the address bar, it says "POST" and the URL "http://webservices.orsprong.org/websamples.countryinfo/CountryInfoService.wsdl". The main content area displays the following XML code:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <CapitalCity xmlns="http://www.orsprong.org/websamples.countryinfo">
      <sCountryISOCode>[string]</sCountryISOCode>
    </CapitalCity>
  </Body>
</Envelope>
```

22. Save script.



23. Berikut ini hasil dari request yang diberikan, dengan menampilkan nama-nama ibukota dari tiap negara.



The screenshot shows the Wizdler editor interface displaying the results of the SOAP request. The results are presented as a list of lines, each starting with a line number (e.g., 607, 608, etc.) and followed by XML code. A specific line for Indonesia is highlighted with a purple background:

607	<m:sISOCode>IS
608	</m:sISOCode>
609	<m:sName>Iceland
610	</m:sName>
611	</m:tCountryCodeAndName>
612	<m:tCountryCodeAndName>
613	<m:sISOCode>IN
614	</m:sISOCode>
615	<m:sName>India
616	</m:sName>
617	</m:tCountryCodeAndName>
618	<m:tCountryCodeAndName>
619	<m:sISOCode>ID
620	</m:sISOCode>
621	<m:sName>Indonesia
622	</m:sName>
623	</m:tCountryCodeAndName>
624	<m:tCountryCodeAndName>
625	<m:sISOCode>IR
626	</m:sISOCode>
627	<m:sName>Iran

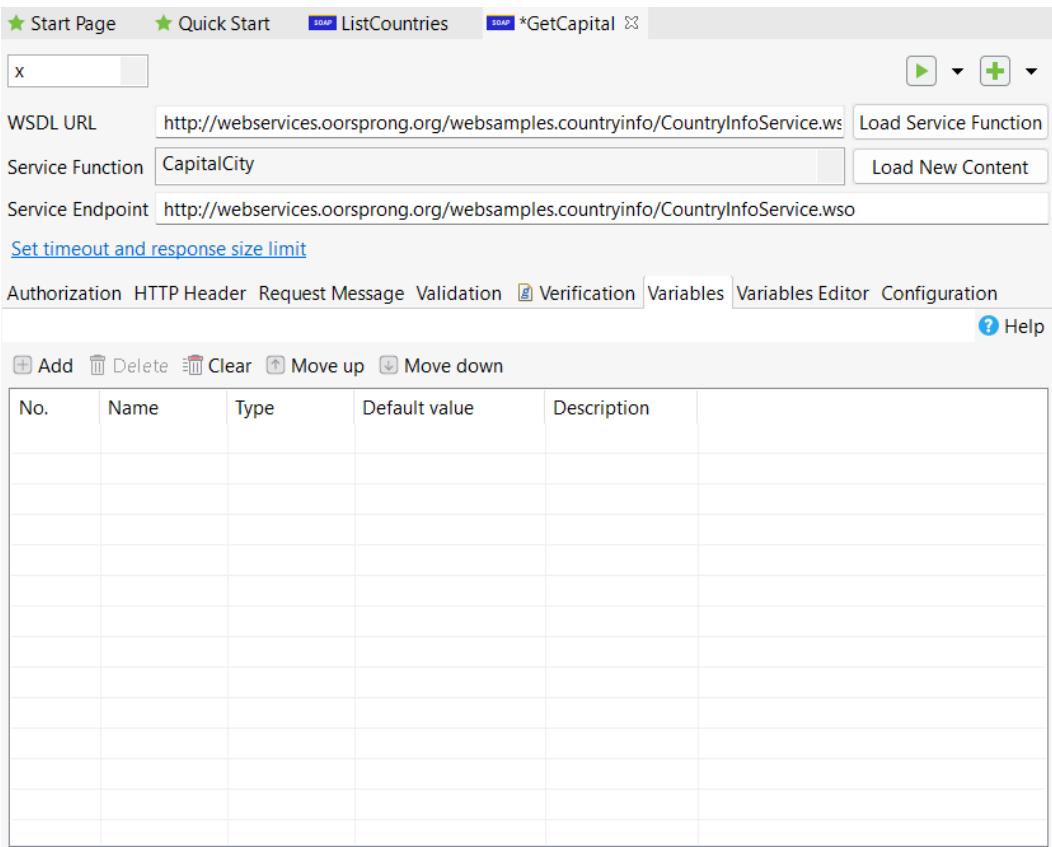
24. Kita juga dapat mengambil request berdasarkan variabel tertentu, seperti yang terlihat pada gambar di bawah ini terjadi perubahan script pada syntaks pemanggilan ISO code.



The screenshot shows a software interface for editing SOAP messages. The top menu bar includes 'Set timeout and response size limit', 'Authorization', 'HTTP Header', 'Request Message', 'Validation' (which is selected), 'Verification', 'Variables', 'Variables Editor', and 'Configuration'. The main area displays the following XML code:

```
1 <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
2   <Body>
3     <CapitalCity
4       xmlns="http://www.oorsprong.org/websamples.countryinfo">
5         <sCountryISOCode>${CountryISOCode}</sCountryISOCode>
6       </CapitalCity>
7     </Body>
</Envelope>
```

25. Kemudian pilih menu variabel dan klik add.



The screenshot shows a software interface for managing service configurations. The top menu bar includes 'Start Page', 'Quick Start', 'ListCountries', 'SOAP *GetCapital', and several other tabs. Below the menu, there are input fields for 'WSDL URL' (set to 'http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wsdl'), 'Service Function' (set to 'CapitalCity'), and 'Service Endpoint' (set to 'http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso'). There is also a link 'Load Service Function' and a button 'Load New Content'. A link 'Set timeout and response size limit' is visible. The bottom section is titled 'Variables' and contains a table with columns 'No.', 'Name', 'Type', 'Default value', and 'Description'. Below the table are buttons for '+ Add', 'Delete', 'Clear', 'Move up', and 'Move down'.

26. Selanjutnya, pilih name yaitu CountryISOCODE, pilih type sebagai Global Variable.

A screenshot of a software interface showing a table with five columns: No., Name, Type, Default value, and Description. The table has one row selected, which contains the value 'CountryISO...' in the Name column, 'Global Vari...' in the Type column, and 'GlobalVariable.null' in the Default value column. The Description column is empty. Below the table are standard toolbar buttons for Add, Delete, Clear, Move up, and Move down.

No.	Name	Type	Default value	Description
1	CountryISO...	Global Vari...	GlobalVariable.null	

27. Buka profiles, kemudian pilih default, isi value dengan ISO CODE yang ingin anda tampilkan, untuk kasus ini saya isi value dengan inisial ID.

A screenshot of a software interface showing the 'Tests Explorer' window and a 'New Variable' dialog box. The 'Tests Explorer' window displays a tree structure with 'Profiles' expanded, showing 'default' selected. Under 'default', 'Test Cases' and 'Object Repository' are listed. 'Object Repository' is expanded, showing 'Countryinfoservices' with 'GetCapital' and 'ListCountries' under it. 'Page_AppointmentConfirmation' and 'Page_CuraAppointment' are also listed. The 'New Variable' dialog box is open at the bottom, titled 'New Variable'. It contains a table with four columns: Name, Value Type, Value, and Description. A single row is present, with 'CountryISO...' in the Name column, 'String' in the Value Type column, and '"IN"' in the Value column. The Description column is empty. At the bottom of the dialog box are 'OK' and 'Cancel' buttons.

Name	Value Type	Value	Description
CountryISO...	String	"IN"	

SOAP Editor		
Start Page Quick Start ListCountries *GetCapital *default		
Add Edit Delete Clear Move Up Move Down		
Name	Value	Description
CountryISOCode	'IN'	

28. Berikut hasil request yang diberikan dari modifikasi sesuai dengan variabel ISO code dengan inisial ID.

Response [HAR](#)

Status: **200 OK** Elapsed: 390 ms Size: 164 bytes

[Body](#) [Header](#) [Verification Log](#) [Validation Log](#)

pretty raw

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <soap:Envelope
3    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
4    <soap:Body>
5      <m:CapitalCityResponse
6        xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
7          <m:CapitalCityResult>Jakarta
8          </m:CapitalCityResult>
9        </m:CapitalCityResponse>
10       </soap:Body>
11     </soap:Envelope>

```

29. Selanjutnya kita akan membuat pengujian test case/kasus uji, terhadap kedua web service request. Anda dapat klik icon tambah, kemudian pilih add to new test case, kemudian beri dengan nama TestOne, lalu klik ok.

[Add Web Service Keyword](#) [Recent keywords](#) [Delete](#) [Move up](#) [Move down](#) [View Execution History](#)

Item	Object	Input	Output	Description
-x 1 - Send Request	ListCountries		response1	

30. Selanjutnya kita akan menyimpan output dari API ListCountries pada variabel yang disebut response1.

[Output](#)

```
response1
```

31. Kemudian pergi ke script, dan anda dapat melihat script dari variabel response1 yang telah dibuat.

```

import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.coretestdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.cucumber.keyword.CucumberBuiltinKeywords as CucumberKW
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.coretestdata.TestData as TestData
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable

response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))

```

32. Sekarang kita akan melakukan fetch pada respon, dan mengambil nilai tertentu dari respon tersebut.

The screenshot shows the Katalon Studio interface with the following details:

- Tests Explorer:** Shows a tree structure of test cases and objects. A specific test case named "TestOne" under "Object Repository" is selected.
- SOAP Request:** The request URL is `http://webservices.orsprong.org/websamples.country/`. The service function is set to `ListCountryNamesByName`.
- Response:**
 - Status: 200 OK
 - Elapsed: 464 ms
 - Size: 35 KB
 - Body (pretty): Displays the XML response content.
- Response Body Content:**

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<?xml version="1.0" encoding="UTF-8"?>
<soap:Body>
<?xml version="1.0" encoding="UTF-8"?>
<ListCountryNamesByNameResponse xmlns:m="http://www.orsprong.org/websamples.countryinfo">
<?xml version="1.0" encoding="UTF-8"?>
<m:ListCountryNamesByNameResult>
<?xml version="1.0" encoding="UTF-8"?>
<m:tCountryCodeAndName>
<n:sISOCode>AX
<n:sName>Åland Islands
</n:sName>
</m:tCountryCodeAndName>
<m:tCountryCodeAndName>
<n:sISOCode>AF
<n:sName>Afghanistan
</n:sName>
</m:tCountryCodeAndName>
<m:tCountryCodeAndName>
<n:sISOCode>AL
<n:sName>Albania
</n:sName>
</m:tCountryCodeAndName>
<m:tCountryCodeAndName>
<n:sISOCode>DZ
</n:sName>
</m:tCountryCodeAndName>

```

33. Pada jendela verifikasi kita dapat melihat list cuplikan yang tersedia, dan dapat digunakan untuk cuplikaan verifikasi.

The screenshot shows the SoapUI interface with the 'Variables' tab selected. Below it is a 'SNIPPETS' section containing several Groovy snippets for testing responses:

- Get current response
- Get current request
- Get a global variable
- Get a variable
- Response body: Contains string
- Response body: Convert to JSON Object
- Response body: Is equal to a String
- Response body: JSON value check
- Response headers: Content-Type header
- Status code: Code is 200
- Status code: Successfully request
- Response body: Array contains

34. Untuk melihat pemeriksaan atau verifikasi nilai XML kita dapat melakukannya dengan mengarahkan kursor ke syntaks Iso Code, kemudian tekan **ctrl + case** pada keyboard, sehingga kita dapat melihat pada jendela show snipped , telah ada verifikasi khusus yang sudah dibuat.

The screenshot shows the SoapUI interface with the 'Variables' tab selected. A specific line of code, `WS.verifyElementText(response, "countryName", "Aland Islands")`, is highlighted. To its right, the XML response is displayed in a 'pretty' formatted view. The 'Aland Islands' entry is highlighted with a blue box, demonstrating the use of ISO codes for verification.

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m>ListOfCountryNamesByNameResponse
      xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
      <m:ListOfCountryNamesByNameResult>
        <m:tCountryCodeAndName>
          <m:sISOCode>AX</m:sISOCode>
          <m:sName>Åland Islands</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>AF</m:sISOCode>
          <m:sName>Afghanistan</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>AL</m:sISOCode>
          <m:sName>Albania</m:sName>
        </m:tCountryCodeAndName>
        <m:tCountryCodeAndName>
          <m:sISOCode>DZ</m:sISOCode>
          <m:sName>Algeria</m:sName>
        </m:tCountryCodeAndName>
      </m:ListOfCountryNamesByNameResult>
    </m>ListOfCountryNamesByNameResponse>
  </soap:Body>
</soap:Envelope>

```

35. Selanjutnya kita akan menuliskan kode untuk menyimpan nilai respon yang ada pada variabel response1 tadi, dimana nilainya kita tampung pada variabel xml1.

```
*import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
*response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
String xml1 = response1.responseBodyContent
```

36. Kemudian parsing nilai respon tersebut ke dalam bentuk xml, dengan kode berikut, dimana hasil akan disimpan pada variabel `dataValue`.

```
*import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
*response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
String xml1 = response1.responseBodyContent
def dataValue= new XmlSlurper().parseText(xml1)
```

37. Untuk mendapatkan nilai atau simpul tertentu, hasil paring yang ada pada variabel `dataValue` tadi, kita simpulkan berdasarkan list country name by name, lalu simpan hasil pada variabel `countryCode`, seperti yang terlihat pada kode berikut.

```
*import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
*response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
String xml1 = response1.responseBodyContent
def dataValue= new XmlSlurper().parseText(xml1)
def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountryCodeAndName[2]
```

38. Selanjutnya hasil pada variabel `countryCode` tadi kita simpan pada global variabel, dengan perintah seperti berikut.

```
*import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
*response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountries'))
String xml1 = response1.responseBodyContent
def dataValue= new XmlSlurper().parseText(xml1)
def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountryCodeAndName[2]
GlobalVariable.countryISOCode = countryCode
```

39. Setelah itu pergi ke jenda test case, dan panggil API GetCapital yang telah dibuat sebelumnya.

The screenshot shows the Katalon Studio interface. On the left, there's a tree view with 'Profiles', 'Test Cases' (containing 'TestOne'), and 'Object Repository' (containing 'CountryInfoService' with 'ListCountries' and 'GetCapital'). The main area is a table with columns: Item, Object, Input, and Output. The 'Item' column lists steps 1 through 6. Step 6, 'Send Request', has 'Object' set to 'GetCapital'. In the 'Input' column, the code `CountryInfoService/GetCapital` is visible. The 'Output' column contains the variable `response1`.

40. Lalu save dan jalankan, dan berikut hasil yang diberikan.

This screenshot shows the 'Job Progress' tab in Katalon Studio. It displays the status 'Test Cases/TestOne - 20181101_151408' and shows the progress bar for the test case execution.

41. Buka jendela console, dan memperlihatkan hasil verifikasi yang telah kita lakukan.

```

[terminated> TempTestCase1541065448450 [Katalon] /Applications/Katalon Studio 6.app/Contents/Eclipse/jre/Contents/Home/jre/bin/java (01-Nov-2018, 3:14:09 PM)
11-01-2018 03:14:13 PM - [START] - Start Test Case : Test Cases/TestOne
11-01-2018 03:14:13 PM - [INFO] - Evaluating variables for test case
11-01-2018 03:14:14 PM - [START] - Start action : Statement - response1 = com.kms.kal.core.response.ResponseObject@541065448450
11-01-2018 03:14:14 PM - [INFO] - Finding Test Object with id 'Object Repository/GetCapital'
11-01-2018 03:14:14 PM - [INFO] - Checking request object
11-01-2018 03:14:16 PM - [INFO] - Send request successfully
11-01-2018 03:14:16 PM - [END] - End action : Statement - response1 = com.kms.kal.core.response.ResponseObject@541065448450
11-01-2018 03:14:16 PM - [START] - Start action : Statement - xml1 = responseBodyContent@541065448450
11-01-2018 03:14:17 PM - [END] - End action : Statement - xml1 = responseBodyContent@541065448450
11-01-2018 03:14:17 PM - [START] - Start action : Statement - dataValue = new groovy.lang.XmlSlurper@541065448450
11-01-2018 03:14:17 PM - [END] - End action : Statement - dataValue = new groovy.lang.XmlSlurper@541065448450
11-01-2018 03:14:17 PM - [START] - Start action : Statement - countryCode = sISOCode@541065448450
11-01-2018 03:14:17 PM - [END] - End action : Statement - countryCode = sISOCode@541065448450
11-01-2018 03:14:17 PM - [START] - Start action : Statement - countryISOCode = countryISOCode@541065448450
11-01-2018 03:14:17 PM - [END] - End action : Statement - countryISOCode = countryISOCode@541065448450
11-01-2018 03:14:17 PM - [START] - Start action : sendRequest
11-01-2018 03:14:17 PM - [INFO] - Finding Test Object with id 'Object Repository/GetCapital'
11-01-2018 03:14:17 PM - [INFO] - Checking request object
11-01-2018 03:14:18 PM - [INFO] - Send request successfully
11-01-2018 03:14:18 PM - [END] - End action : sendRequest
11-01-2018 03:14:18 PM - [PASSED] - Test Cases/TestOne
11-01-2018 03:14:18 PM - [END] - End Test Case : Test Cases/TestOne

```

42. Untuk melihat country code yang di ekstrak, dapat tambahkan code berikut.

```

import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable

response1 = WS.sendRequest(findTestObject('CountryInfoService/ListCountryInfo'))

String xml1 = response1.responseBodyContent

def dataValue = new XmlSlurper().parseText(xml1)

def countryCode = dataValue.ListOfCountryNamesByNameResult.tCountry[0].countryISOCode

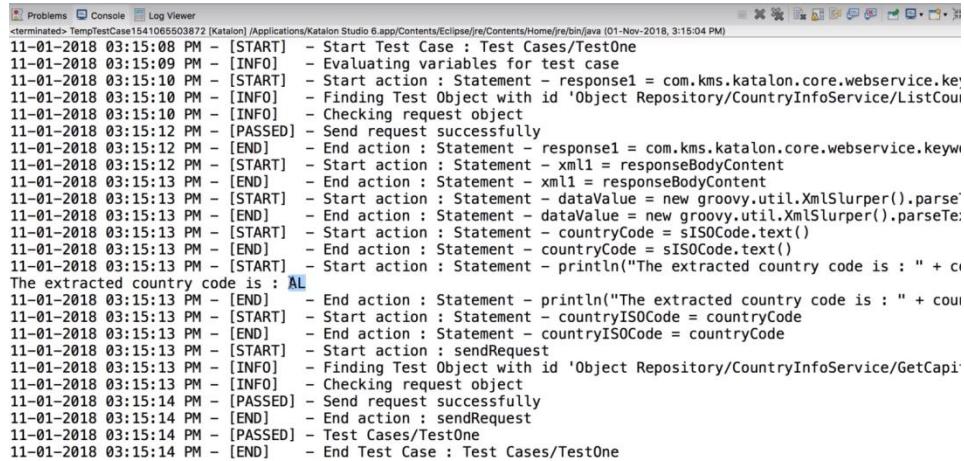
println("The extracted country code is : "+countryCode)

GlobalVariable.countryISOCode = countryCode

WS.sendRequest(findTestObject('CountryInfoService/GetCapital'))

```

43. Dapat dilihat berikut hasil dari country code yang di ekstrak, yaitu country code dengan kode negara AL.



The screenshot shows the Katalon Studio Log Viewer window with the following log entries:

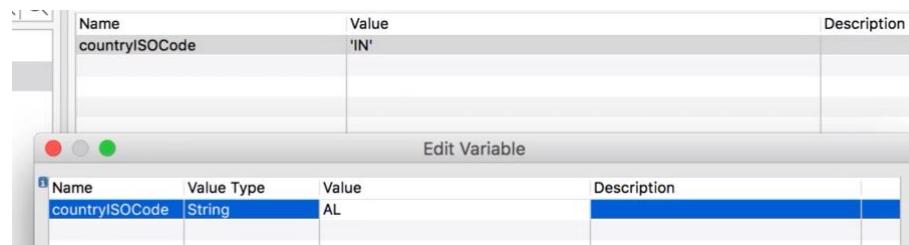
```

11-01-2018 03:15:08 PM - [START] - Start Test Case : Test Cases/TestOne
11-01-2018 03:15:09 PM - [INFO] - Evaluating variables for test case
11-01-2018 03:15:10 PM - [START] - Start action : Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords$SendRequestAction@540d1f
11-01-2018 03:15:10 PM - [INFO] - Finding Test Object with id 'Object Repository/CountryInfoService>ListCountryInfo'
11-01-2018 03:15:10 PM - [INFO] - Checking request object
11-01-2018 03:15:12 PM - [PASSED] - Send request successfully
11-01-2018 03:15:12 PM - [END] - End action : Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords$SendRequestAction@540d1f
11-01-2018 03:15:12 PM - [START] - Start action : Statement - xml1 = responseBodyContent
11-01-2018 03:15:12 PM - [END] - End action : Statement - xml1 = responseBodyContent
11-01-2018 03:15:13 PM - [START] - Start action : Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1)
11-01-2018 03:15:13 PM - [END] - End action : Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1)
11-01-2018 03:15:13 PM - [START] - Start action : Statement - countryCode = sISOCode.text()
11-01-2018 03:15:13 PM - [END] - End action : Statement - countryCode = sISOCode.text()
11-01-2018 03:15:13 PM - [START] - Start action : Statement - println("The extracted country code is : "+ countryCode)
11-01-2018 03:15:13 PM - [END] - End action : Statement - println("The extracted country code is : "+ countryCode)
11-01-2018 03:15:13 PM - [START] - Start action : Statement - countryISOCode = countryCode
11-01-2018 03:15:13 PM - [END] - End action : Statement - countryISOCode = countryCode
11-01-2018 03:15:13 PM - [START] - Start action : sendRequest
11-01-2018 03:15:13 PM - [INFO] - Finding Test Object with id 'Object Repository/CountryInfoService/GetCapital'
11-01-2018 03:15:13 PM - [INFO] - Checking request object
11-01-2018 03:15:14 PM - [PASSED] - Send request successfully
11-01-2018 03:15:14 PM - [END] - End action : sendRequest
11-01-2018 03:15:14 PM - [PASSED] - Test Cases/TestOne
11-01-2018 03:15:14 PM - [END] - End Test Case : Test Cases/TestOne

```

The log shows the execution of a test case named "TestOne". It starts by evaluating variables, then sends a request to the "ListCountryInfo" endpoint, parses the response using an XmlSlurper, and extracts the country ISO code ("AL"). It then prints this value to the console and sets it as a global variable ("countryISOCode"). Finally, it sends a request to the "GetCapital" endpoint and ends the test case.

44. Untuk memvalidasi country code tersebut, pergi ke halaman default dan ubah value dari country code menjadi AL.



45. Lalu save dan jalankan, hasil memperlihatkan country coude dengan inisial AL adalah negara Tirana.

```

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <GetCapital xmlns="http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wsdl">
      <sCountryISOCode>${countryISOCode}</sCountryISOCode>
    </GetCapital>
  </Body>
</Envelope>

```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:CapitalCityResponse xmlns:m="http://www.oorsprong.org/websamples.countryinfo">
      <m:CapitalCityResult>Tirana</m:CapitalCityResult>
    </m:CapitalCityResponse>
  </soap:Body>
</soap:Envelope>

```

46. Kembali ke halaman verifikasi, arahkan kursor ke “Tirana” lalu tekan ctrl + case pada keyboard, dan dapat dilihat code verifikasi telah terbentu.

```

import static org.assertj.core.api.Assertions
import com.kms.katalon.core.testobject.RequestObject
import com.kms.katalon.core.testobject.ResponseObject
import com.kms.katalon.core.webservice.keyword.KatalonWebServiceKeywords
import com.kms.katalon.core.webservice.verify.Verify

import groovy.json.JsonSlurper
import internal.GlobalVariable as GlobalVariable

```

```

RequestObject request = WSResponseManager.getRequest()
ResponseObject response = WSResponseManager.getResponse()

WS.verifyElementText(response, 'CapitalCityRe')

```

Test scripts are written in Groovy and only run after response is received.

SNIPPETS

- Get current response
- Get current request
- Get a global variable
- Get a variable
- Response body: Contains string
- Response body: Convert to JSON Object
- Response body: Is equal to a String
- Response body: JSON value check
- Response headers: Content-Type header
- Status code: Code is 200
- Status code: Successfully request
- Response body: Array contains

47. Selanjutnya masuk ke halaman test case, lalu update item dengan “send request and verify”. Hal ini dilakukan untuk memverifikasi permintaan/request dari API yang dilakukan.

Item	Object	Input	Output
-x 1 - Send Request	ListCountries		response1
DOI 2 - Binary Statement		xml1 = response1.responseBodyContent	
DOI 3 - Binary Statement		dataValue = new XmlSlurper().parseText(xml1)	
DOI 4 - Binary Statement		countryCode = dataValue.ListOfCountry[0].countryCode	
f_x 5 - Method Call Statement		println("The extracted country code is " + countryCode)	
DOI 6 - Binary Statement		GlobalVariable.countryISOCode = countryCode	
-x Send Request And Verify	GetCapital		

48. Kemudian save script, dan jalankan tet case.

The screenshot shows the Katalon Studio interface with the 'Job Progress' tab selected. It displays the status of three test cases under the 'Test Cases/TestOne' category. The first test case, '20181101_151649', is currently running. The second test case, '20181101_151503', has completed successfully (indicated by a green checkmark). The third test case, '20181101_151408', has also completed successfully. The 'Test Cases/TestOne' section in the main pane shows the script steps: 1 - Send Request, 2 - Binary Statement, 3 - Binary Statement, 4 - Binary Statement, 5 - Method Call Statement, 6 - Binary Statement, and 7 - Send Request And Verify.

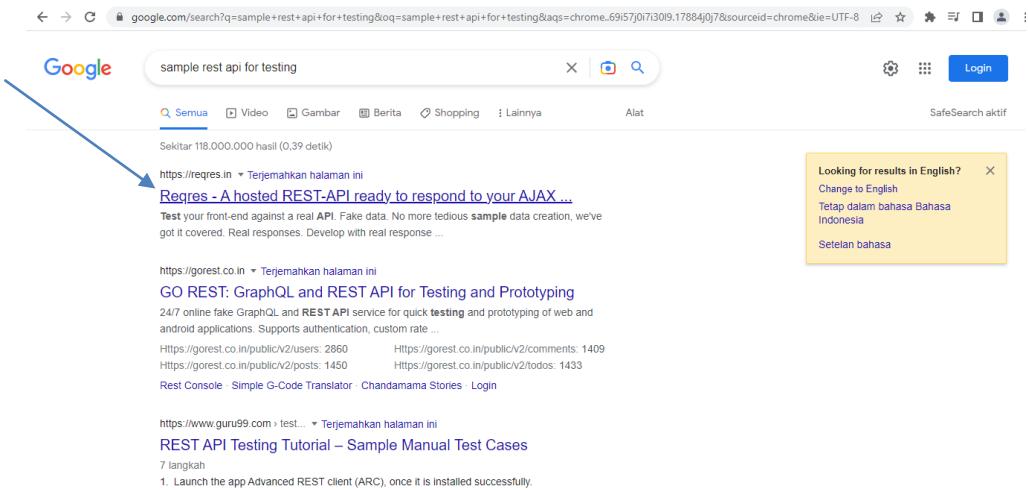
49. Setelah proses running selesai dan berhasil, pergi ke halaman log viewer dan kita dapat melihat verifikasi yang sudah berhasil dan berfungsi dengan baik.

The screenshot shows the Katalon Studio Log Viewer. On the left, a tree view lists the test steps: 'Evaluating variables for test case', 'Statement - response1 = com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords.sendRequest...', 'Statement - xml1 = responseBodyContent', 'Statement - dataValue = new groovy.util.XmlSlurper().parseText(xml1)', 'Statement - countryCode = sISOCode.text()', 'Statement - println("The extracted country code is :" + countryCode)', 'Statement - countryISOCode = countryCode', and 'sendRequestAndVerify'. On the right, a detailed log entry for the '7 - sendRequestAndVerify' step is shown. The log entry includes the step name, start time (11-01-2018 03:16:59 PM), elapsed time (0.080s), and message ('Verify element text successfully').

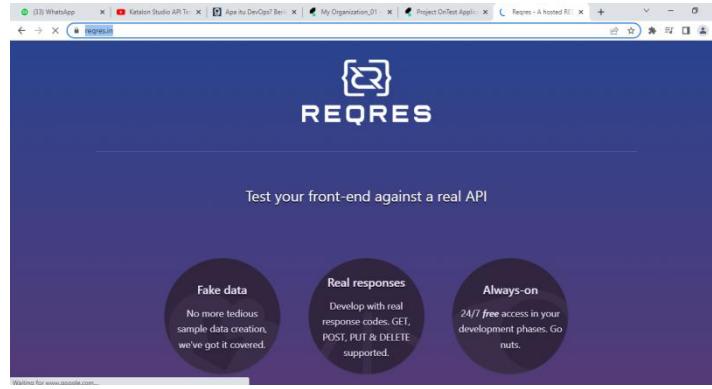
9.3. API CHAINING | REST - JSON | HOW TO SEND VALUE FROM ONE API TO OTHER

Setelah rangkaian pengujian atau kumpulan rangkaian pengujian, sesi terakhir kita telah melihat cara mengek nilai kita dari respon XML kita dan kemudian memberikan nilai itu dalam permintaan API, selanjutnya kita akan menggunakan API Rest dan melihat bagaimana melakukannya dengan respons Json. Berikut tahapan-tahapannya.

1. Setelah membuka katalon studio. Selanjutnya kita akan mencari sampel sisa untuk pengujian dan disini kita memiliki sampel eqres di web



2. Situs web ini memiliki api istirahat yang baik untuk pengujian dan disini kita dapat melihat ini memiliki api palsu sehingga kita dapat menggunakan untuk pengujian dan yang paling pasti berlaku kita akan membuat url ini tersedia sehingga kita dapat merujuk mereka menjadi



3. Disini bisa kita lihat jika kita turun kita akan memiliki banyak aplikasi dan kita telah mendapatkan posting yang di hapus batch dan seterusnya kita akan menggunakan daftar ini, pengguna mendapatkan API, jadi kita akan menyalin url ini dan meletakkan nya di tab baru.

A screenshot of a web browser displaying the Reqres API testing interface. The page shows a "Request" table and a "Response" panel. The Request table lists various API endpoints with their methods and descriptions: LIST USERS (GET), SINGLE USER (GET), SINGLE USER NOT FOUND (GET), LIST <RESOURCE> (GET), SINGLE <RESOURCE> (GET), SINGLE <RESOURCE> NOT FOUND (GET), CREATE (POST), and UPDATE (PUT). The Response panel shows a JSON response for the endpoint "/api/users?page=2" with status code 200. The JSON data includes information about users, such as id, email, first_name, last_name, and avatar. The response is as follows:

```
{  
  "page": 2,  
  "per_page": 6,  
  "total": 12,  
  "total_pages": 2,  
  "data": [  
    {  
      "id": 7,  
      "email": "michael.lawson@reqres.in",  
      "first_name": "Michael",  
      "last_name": "Lawson",  
      "avatar": "https://reqres.in/img  
    },  
    {  
      "id": 8,  
      "email": "lindsay.ferguson@reqres.in",  
      "first_name": "Lindsay",  
      "last_name": "Ferguson",  
      "avatar": "https://reqres.in/img  
    },  
    {  
      "id": 9,  
      "email": "tobias.funke@reqres.in",  
      "first_name": "Tobias",  
      "last_name": "Funke",  
      "avatar": "https://reqres.in/img  
    },  
    {  
      "id": 10,  
      "email": "byron.fields@reqres.in",  
      "first_name": "Byron",  
      "last_name": "Fields",  
      "avatar": "https://reqres.in/img  
    },  
    {  
      "id": 11,  
      "email": "george.kingsley@reqres.in",  
      "first_name": "George",  
      "last_name": "Kingsley",  
      "avatar": "https://reqres.in/img  
    },  
    {  
      "id": 12,  
      "email": "david.wong@reqres.in",  
      "first_name": "David",  
      "last_name": "Wong",  
      "avatar": "https://reqres.in/img  
    }  
  ]  
}
```

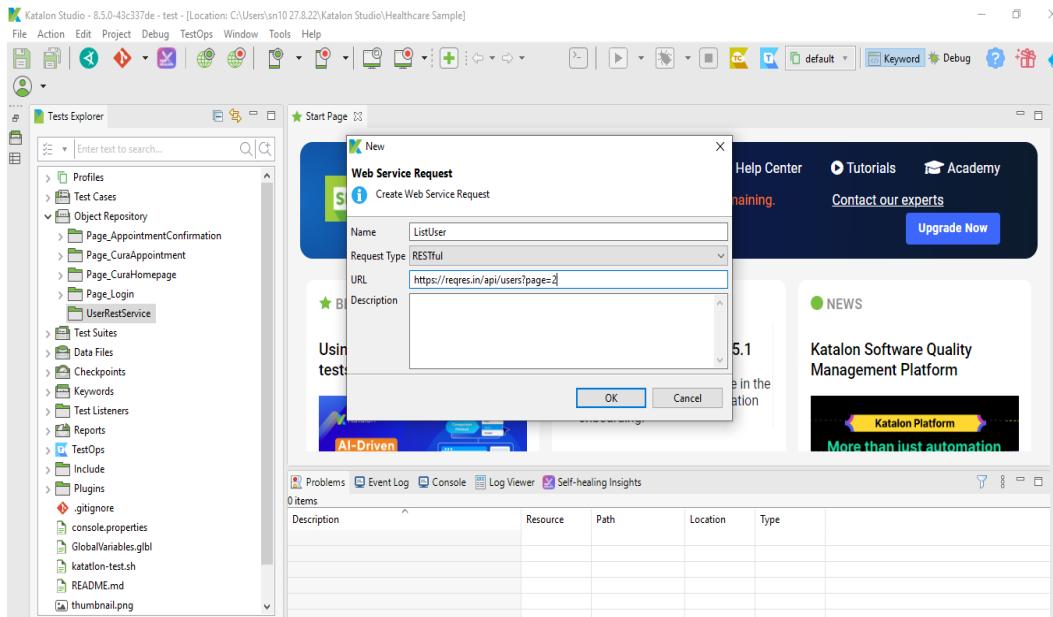
4. Menyalin teks url reqres.in dan menyalin sisa titik akhir dari sini dan menambahkan nya lagi, dan ini adalah titik terakhir API jika kita menjalankan nya kita dapat melihat respon. Jadi saya akan menyalin url ini

```

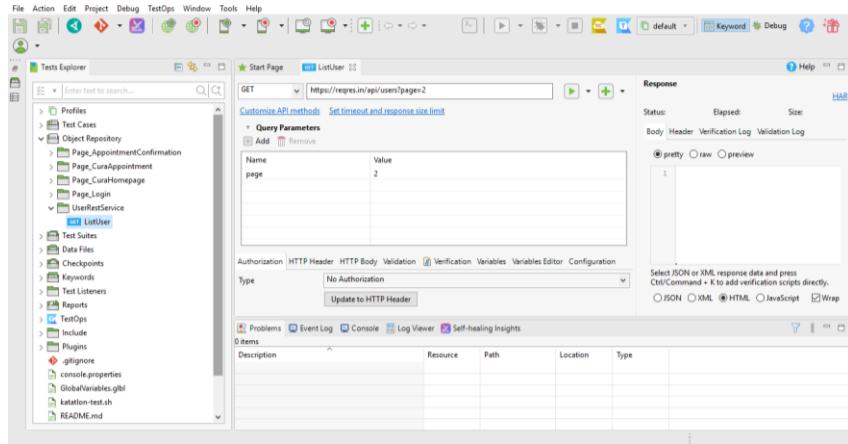
{
  "page": 2,
  "per_page": 6,
  "total": 12,
  "total_pages": 2,
  "data": [
    {
      "id": 7,
      "email": "michael.lawson@reqres.in",
      "first_name": "Michael",
      "last_name": "Lawson",
      "avatar": "https://reqres.in/img/faces/7-image.jpg"
    },
    {
      "id": 8,
      "email": "lindsay.ferguson@reqres.in",
      "first_name": "Lindsay",
      "last_name": "Ferguson",
      "avatar": "https://reqres.in/img/faces/8-image.jpg"
    },
    {
      "id": 9,
      "email": "tobias.funke@reqres.in",
      "first_name": "Tobias",
      "last_name": "Funke",
      "avatar": "https://reqres.in/img/faces/9-image.jpg"
    },
    {
      "id": 10,
      "email": "byron.fields@reqres.in",
      "first_name": "Byron",
      "last_name": "Fields",
      "avatar": "https://reqres.in/img/faces/10-image.jpg"
    },
    {
      "id": 11,
      "email": "george.edwards@reqres.in",
      "first_name": "George",
      "last_name": "Edwards",
      "avatar": "https://reqres.in/img/faces/11-image.jpg"
    },
    {
      "id": 12,
      "email": "rachel.howell@reqres.in",
      "first_name": "Rachel",
      "last_name": "Howell",
      "avatar": "https://reqres.in/img/faces/12-image.jpg"
    }
  ],
  "support": {
    "url": "https://reqres.in/#support-heading",
    "text": "To keep ReqRes free, contributions towards server costs are appreciated!"
  }
}

```

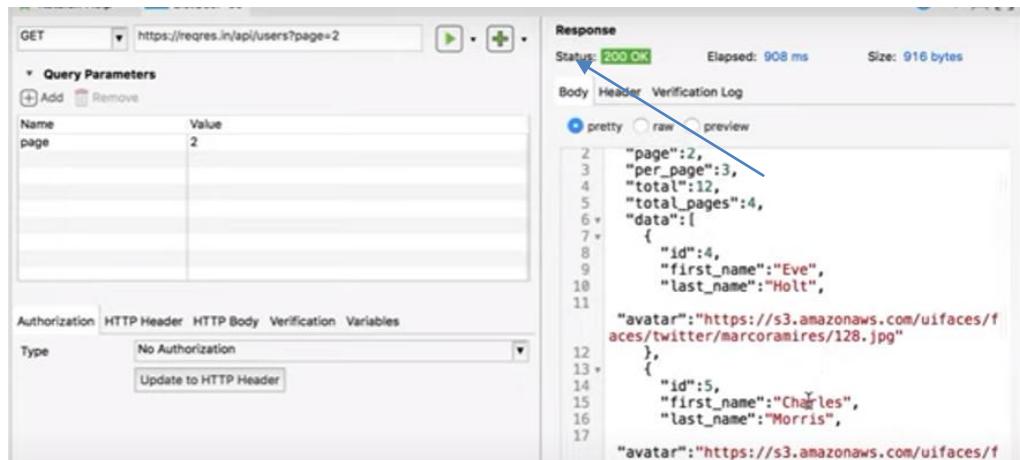
- Setelah menyalin url, kemudian buka katalon studio dan buat folder baru pada object repository dengan nama UserRestService dan memasukkan link url ke web service request



- Dan ini hasil setelah kita masukkan link url dan tidak ada otorsasi yang diperlukan untuk contoh API khusus ini, jika kita memerlukan otorisasi kita dapat mengedit di verifikasi dan dalam verifikasi kita dapat menambahkan beberapa cuplikan dan variable



7. Dan kita jalankan apakah sesuai dan kita akan melihat apa yang harus dilakukan kemudian kita memeriksa apakah kita mendapatkan respons yang valid. Dan hasil nya seperti pada gambar dibawah dengan status 200 ok. Dan kita juga mendapatkan daftar pengguna dengan API ini dan pengujian ini berfungsi dengan baik.



MODUL 12

CUSTOM KEYWORS

A. Tujuan

1. Mengetahui apa itu custom keywords pada katalon studio.
2. Mengetahui bagaimana cara membuat dan cara merujuk costum keywords pada katalon studio.

B. Dasar Teori

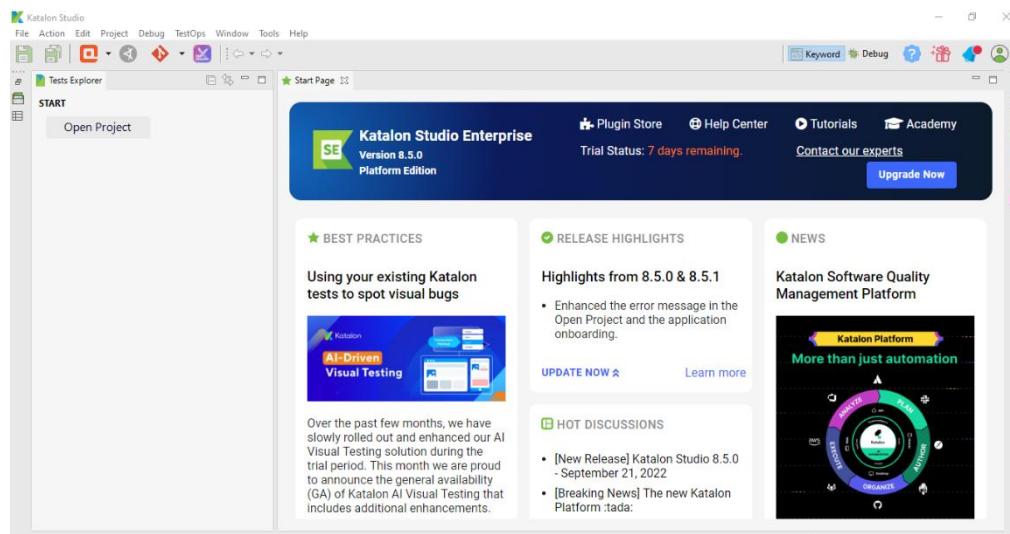
Custom keywords merupakan kata kunci khusus yang dapat anda gunakan untuk memperluas kemampuan Katalon Studio. Kata kunci ini berguna saat kita ingin menerapkan kasus pengujian, sama seperti kata kunci bawaan lainnya.

12.1. HOW TO CREATE CUSTOM KEYWORDS

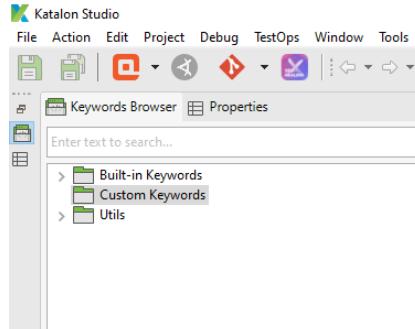
Bagaimana cara membuat custom keywords pada katalon studio? Selain kata kunci bawaan, pada katalon kita juga dapat menentukan kata kunci khusus, berikut ini langkah-langkah dalam pembuatan costum keywords pada katalon studio.

12.1.1. Bagaimana cara membuat kata kunci khusus

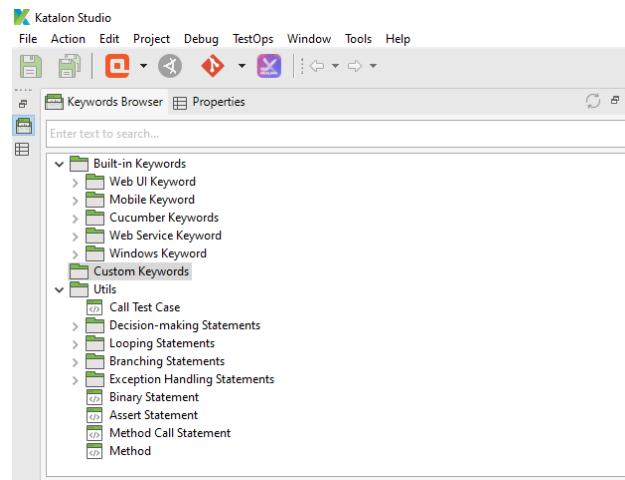
1. Untuk membuat kata kunci khusus pertama bukan aplikasi katalon studio.



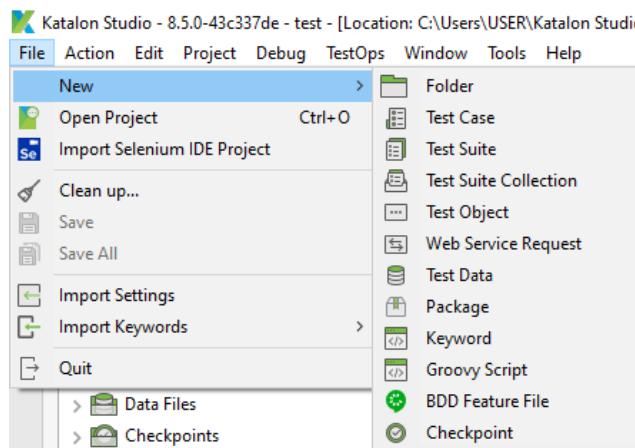
2. Kemudian pilih costume keywords.



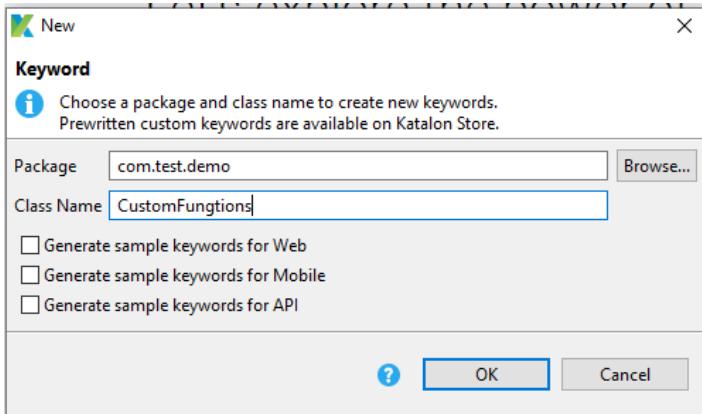
3. Berikut adalah fitur-fitur yang terdapat pada costume keyword dan semua kata kunci yang terdapat pada keyword dibawah ini semua bias dijalankan.



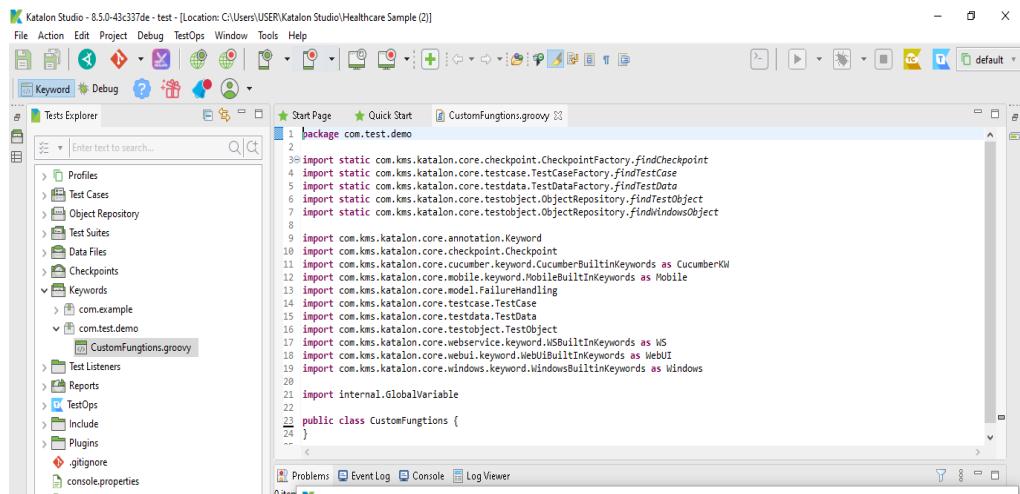
4. Langkah pertama membuat kata kunci khusus adalah klik file, lalu pilih keyword.



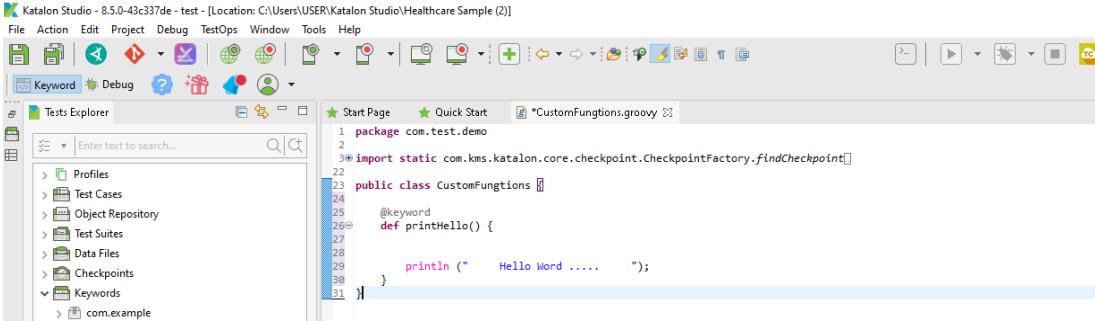
5. Kemudian inputkan package dengan nama *com.test.demo* lalu pada class name inputkan dengan nama *CustomFungsions*.



6. Setelah mengklik OK maka akan muncul tampilan seperti dibawah ini lalu Kata kunci baru dibuat di bawah paket yang ditentukan.

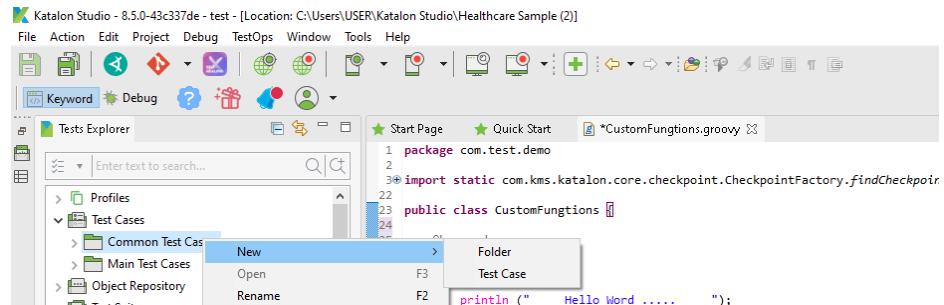


7. Masukkan kode berikut di kelas yang telah ditentukan untuk menentukan kata kunci khusus:

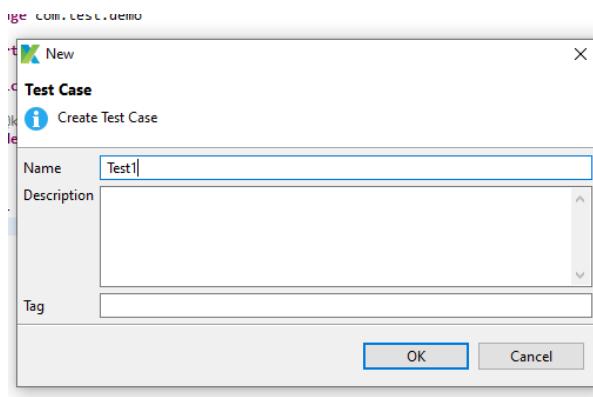


```
package com.test.demo
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
public class CustomFuntions {
    @keyword
    def printHello() {
        println ("Hello Word ....");
    }
}
```

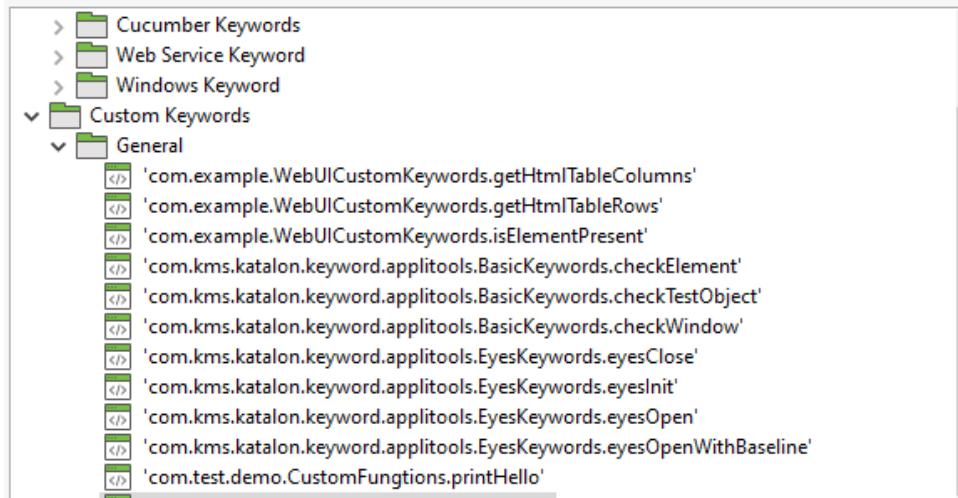
8. Kemudian buka kasus uji dalam tampilan Manual dan pilih Kata Kunci Khusus dari toolbar perintah`, pilih “NEW KLIK TEST CASE”.



9. Inputkan name case sebagai Test1, lalu klik ok.



10. Selanjutnya Langkah pengujian baru ditambahkan. Pilih salah satu kata kunci khusus.



11. Setelah ditambahkan maka akan muncul tampilan seperti gambar di bawah.

Item	Object	Input	Output	Description
-> 1 - com.test.demo.CustomFur				

12. Selanjutnya klik running dan tunggu beberapa saat jika telah selesai klik Console maka muncul inputan Hello word

```

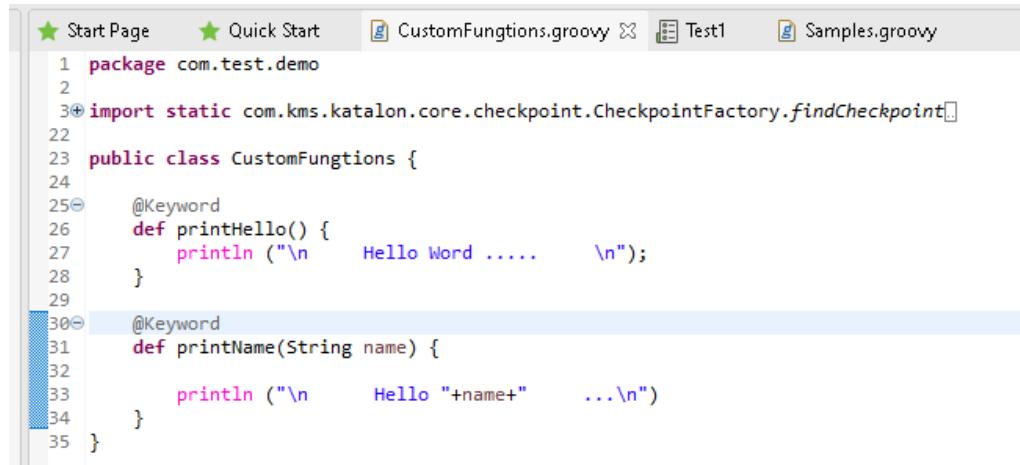
Manual </> Script </> Variables </> Variables (Script mode) Data Binding Integration Properties

Problems EventLog Console Log Viewer
<terminated> TempTestCase1665839381023[Katalon] F:\SEMESTER 7\TOERI KPL PAK ARHAM\Katalon_Studio_PE_Windows_64-8.5.0\jre\bin\javaw.exe (Oct 15, 2022 6:09:42 AM - 6:09:53 AM)
2022-10-15 06:09:51.946 INFO c.k.katalon.core.main.TestCaseExecutor - -----
2022-10-15 06:09:51.946 INFO c.k.katalon.core.main.TestCaseExecutor - START Test Cases/Common Test Cases/Test1
2022-10-15 06:09:53.251 DEBUG testcase.TestCase1 - 1: com.test.demo.CustomFungtions.printHello()
Hello Word .....
2022-10-15 06:09:53.284 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printHello is PASSED
2022-10-15 06:09:53.315 INFO c.k.katalon.core.main.TestCaseExecutor - END Test Cases/Common Test Cases/Test1

```

12.1.2. Bagaimana cara merujuk kata kunci khusus

13. Pertama buat terlebih dahulu kata kunci lain, seperti gambar di bawah ini.

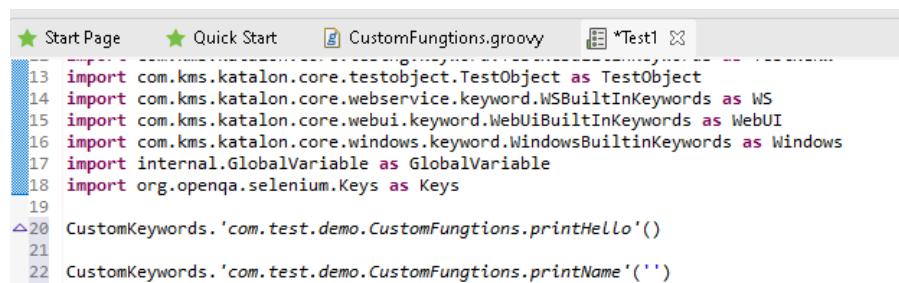


```
1 package com.test.demo
2
3 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
4
5 public class CustomFuntions {
6
7     @Keyword
8     def printHello() {
9         println ("\n      Hello Word ..... \n");
10    }
11
12    @Keyword
13    def printName(String name) {
14        println ("\n      Hello "+name+" ..... \n")
15    }
16}
```

14. Untuk menginputkan file dibawahnya langkah yang dilakukan sama seperti langkah sebelumnya.

Item	Object	Input	Output	Description
-> 1 - com.test.demo.CustomFuntions				
-> 2 - com.test.demo.CustomFuntions		'''		

15. Dan saat menu skrip dibuka, maka kita dapat melihat tampilan seperti gambar ini.

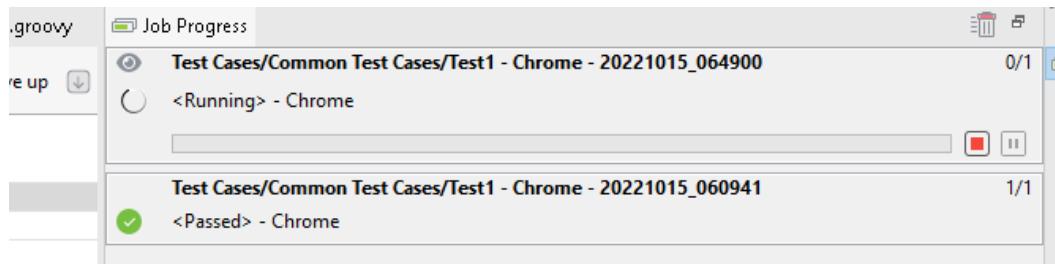


```
13 import com.kms.katalon.core.testobject.TestObject as TestObject
14 import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
15 import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
16 import com.kms.katalon.core.windows.keyword.WindowsBuiltInKeywords as Windows
17 import internal.GlobalVariable as GlobalVariable
18 import org.openqa.selenium.Keys as Keys
19
20 CustomKeywords.'com.test.demo.CustomFuntions.printHello'()
21
22 CustomKeywords.'com.test.demo.CustomFuntions.printName'('')
```

16. Kemudian kembali lagi ke halaman Test1 klik tanda petik “” lalu inputkan kata kunci contohnya “Raihan” lalu klik ok.

No.	Param Name	Param Type	Value Type	Value
1	name	String	String	""Raihan""

17. Setelah itu klik running, dan tunggu hingga proses running selesai.



18. Jika proses running telah selesai, klik Consule dan akan muncul kata kunci tambahan.

```

Problems Event Log Console Log Viewer
<terminated> TempTestCase1665841740893 [Katalon] F:\SEMESTER 7\TOERI KPL PAK ARHAMI\Katalon_Studio_PE_Windows_64-8.5.0\jre\bin\javaw.exe (Oct 15, 2022 6:49:01 AM - 6:49:20
2022-10-15 06:49:18.854 INFO c.k.katalon.core.main.TestCaseExecutor - -----
2022-10-15 06:49:18.863 INFO c.k.katalon.core.main.TestCaseExecutor - START Test Cases/Common Test Cases/Test1
2022-10-15 06:49:20.180 DEBUG testcase.Test1 - 1: com.test.demo.CustomFungtions.printHello()

Hello Word .....

2022-10-15 06:49:20.230 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printHello is PASSED
2022-10-15 06:49:20.231 DEBUG testcase.Test1 - 2: com.test.demo.CustomFungtions.printName("Raihan")

Hello "Raihan" ...

2022-10-15 06:49:20.234 INFO k.k.c.m.CustomKeywordDelegatingMetaClass - com.test.demo.CustomFungtions.printName is PASSED
2022-10-15 06:49:20.271 INFO c.k.katalon.core.main.TestCaseExecutor - END Test Cases/Common Test Cases/Test1

```

MODUL 17

WINDOWS DEKSTOP APPLICATION TESTING

A. Tujuan

1. Mempelajari tentang pengujian terhadap dekstop.
2. Mengetahui bagaimana cara melakukan pengujian terhadap dekstop dengan menggunakan katalon studio.

B. Dasar Teori

Aplikasi desktop adalah program yang berjalan secara independen di sistem operasi desktop. Tidak seperti aplikasi web, aplikasi desktop memerlukan sumber daya perangkat keras yang cukup untuk berfungsi.

Pengujian aplikasi desktop adalah praktik pengujian perangkat lunak yang memeriksa fungsionalitas, keamanan, kegunaan, dan stabilitas aplikasi setelah diterapkan. Dalam pengujian aplikasi desktop, kita perlu memperhatikan pemasangan serta pengujian penghapusan instalasi untuk sepenuhnya memenuhi persyaratan pengujian aplikasi. Apa yang bisa diuji pada dekstop dengan menggunakan katalon studio? Katalon studio dapat melakukan beberapa pengujian diantaranya :

- Melakukan pengaturan dan konfigurasi yang mudah – baik secara lokal maupun jarak jauh
- Mendeteksi dan memata-matai objek Windows
- Merekam tindakan Windows
- Menguji lokasi elemen cerdas
- Menguji kata kunci bawaan dan khusus Windows Perawatan minimal

Berikut ini merupakan langkah-langkah untuk melakukan pengujian dekstop dengan menggunakan Katalon Studio.

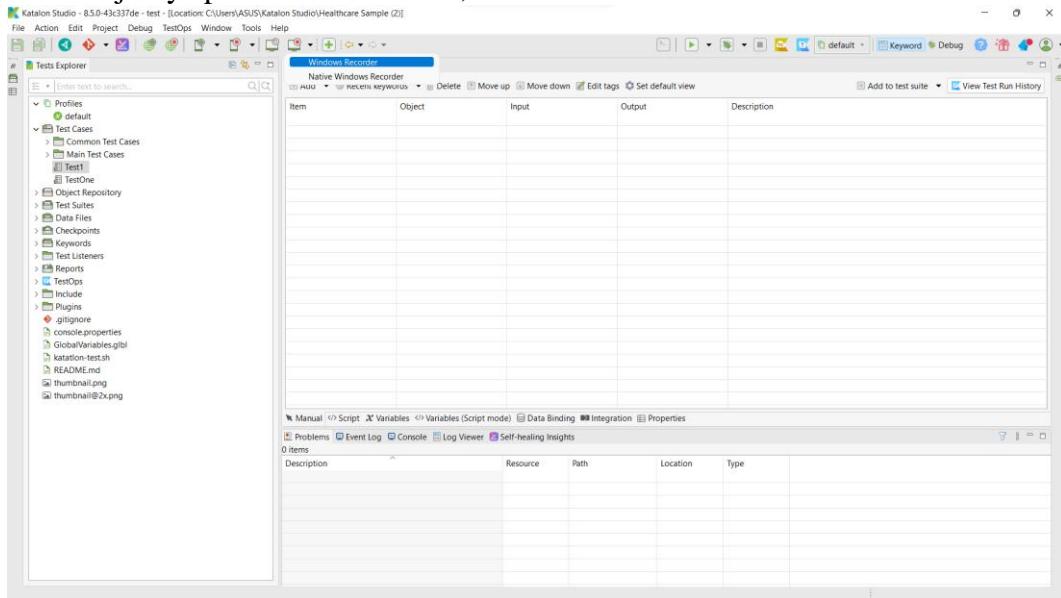
1. Sebelum melakukan pengujian terhadap dekstop pada katalon studio, pastikan katalon studio yang digunakan sudah dalam versi terbaru. Jika anda masih menggunakan katalon studio versi lama, anda dapat mendownload versi terbaru katalon studio pada web resmi katalon studio.com. Hal ini dilakukan untuk memastikan bahwa menu atau tools pengujian terhadap dekstop telah tersedia.

The screenshot shows the 'Katalon docs' website at <https://docs.katalon.com/docs/general-information/release-notes/katalon-studio/katalon-studio-release-notes-version-8-x#version-852>. The main content area displays the 'Version 8.5.2' section under 'Enhancements', which includes a bullet point about changing the activation flow of the Katalon Runtime Engine. Below it is the 'Version 8.5.1' section, also under 'Enhancements'. To the right, a vertical sidebar lists the contents of previous versions: 'Version 8.5.1 Enhancements', 'Version 8.5.0 New features', 'Version 8.5.0 Enhancements', 'Changes for Katalon Studio - Platform Edition', 'Known limitations', 'Version 8.4.1 Enhancements', 'Version 8.4.0 New features', 'Version 8.4.0 Enhancements', 'Fixes', and 'Version 8.3.5 New features'.

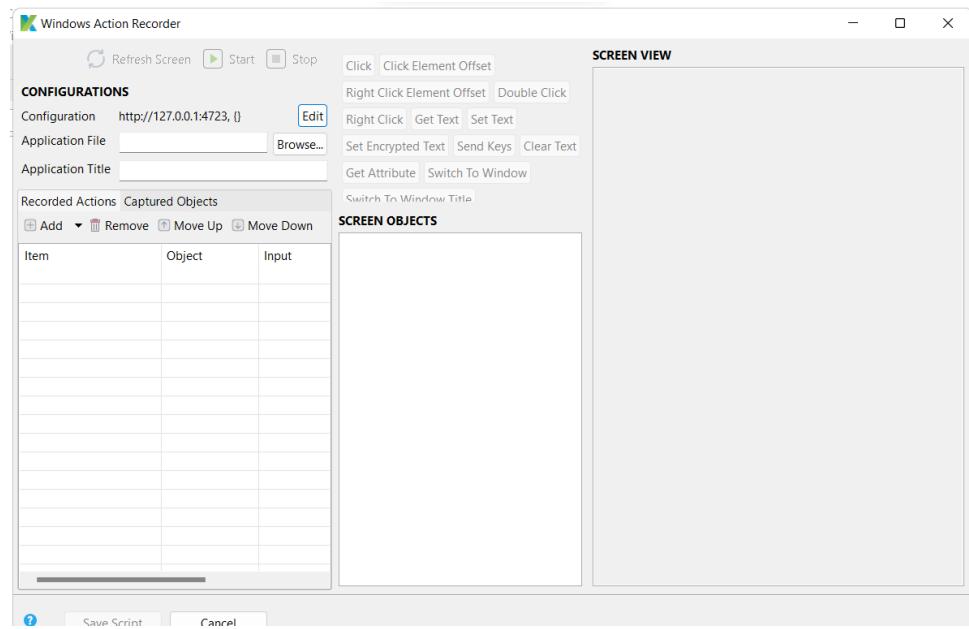
2. Setelah katalon studio berhasil di download, buka katalon studio kemudian untuk memastikan apakah katalon studio yang dipakai bisa melakukan pengujian terhadap dekstop, pada menu utama buka projek, lalu pilih Desired Capabilities, pada menu tersebut bisa dilihat bahwa pilihan untuk windows telah tersedia.

The screenshot shows the Katalon Studio interface with the title bar 'Katalon Studio - 8.5.0-43c337de - test - [Location: C:\Users\ASUS\Katalon Studio\Healthcare Sample (2)]'. The left side features a 'Tests Explorer' window with a tree view of project files. On the right, a 'Project Settings' dialog box is open, specifically the 'Windows' tab under the 'Desired Capabilities' section. The dialog box contains a table with three columns: 'Name', 'Type', and 'Value'. At the bottom of the dialog box are buttons for 'Import', 'Export', 'Restore Defaults', 'Apply', 'Apply and Close', and 'Cancel'.

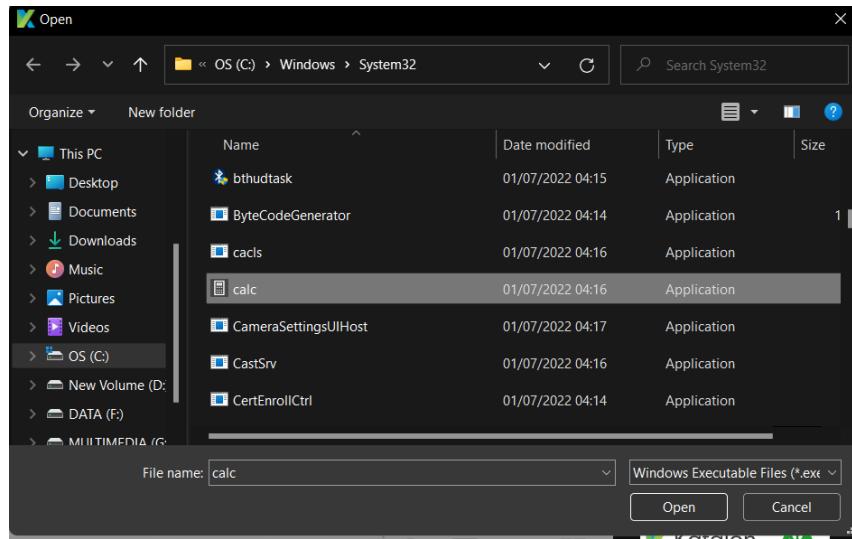
3. Selanjutnya pada menu test case, buat satu test case baru.



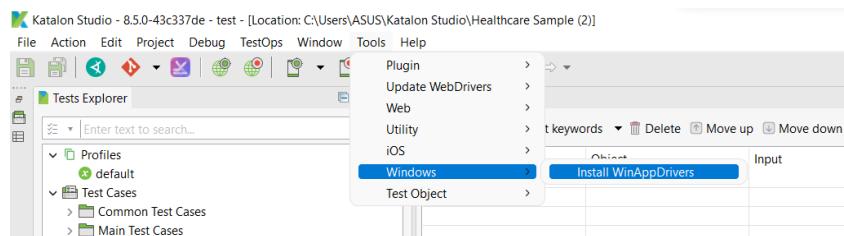
4. Klik tombol windows recorder. Setelah tombol di klik, maka user akan disuguhkan dengan tampilan jendela yang berisi rekaman tindakan windows.



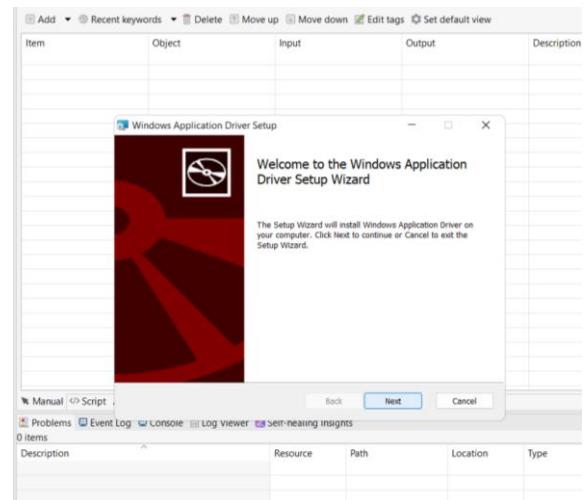
5. Pada application file, pilih file yang ingin diuji, dan biasanya file yang diuji berformat .exe. Pada kasus ini saya menggunakan file calc.exe untuk pengujiannya. File calc.exe bisa didapat pada folder AdvancedInstallers yang berada di dalam folder System32 pada localdisc C. Setelah file dipilih, lalu klik ok.



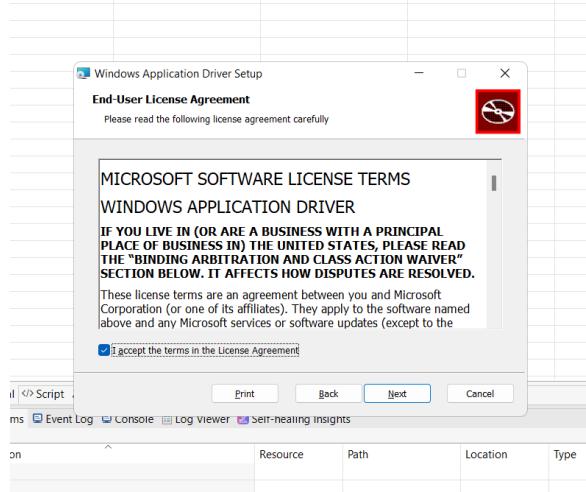
6. Sebelum melakukan proses recording, ada beberapa hal yang harus dilakukan. Terlebih dahulu kita harus menginstall winAppDriver, winAppDriver ini digunakan sebagai libarary agar bisa berinteraksi dengan aplikasi windows. Proses penginstallan dapat dilakukan dengan tools pada menu utama, pilih windows, lalu klik Install WinAppDrivers.



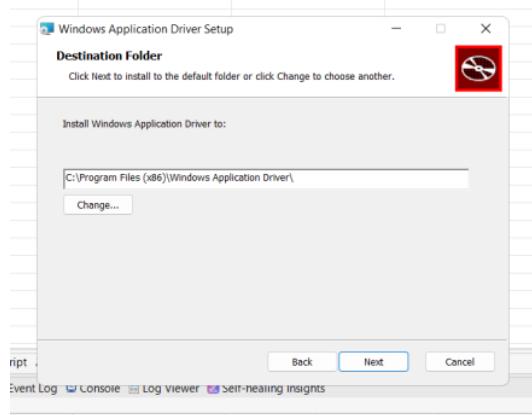
7. Pada instalasi wizard klik next saja.



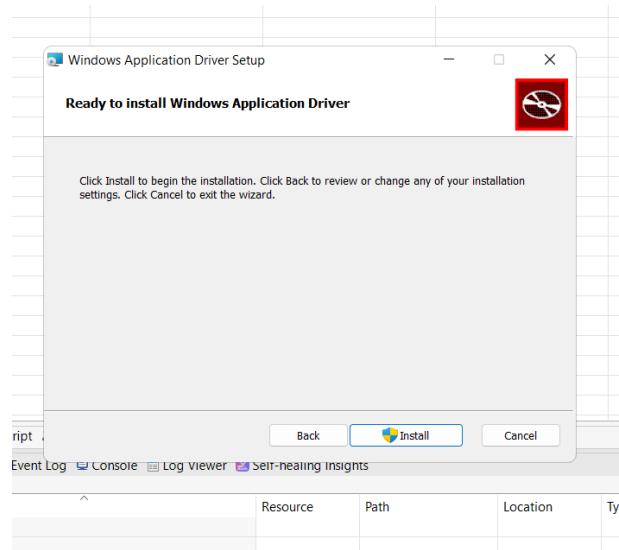
8. Pilih I accept untuk license agreement, kemudian next.



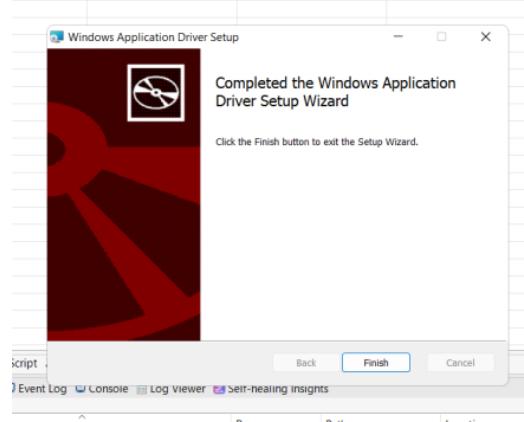
9. Pilih lokasi penyimpanan file install, lalu klik next.



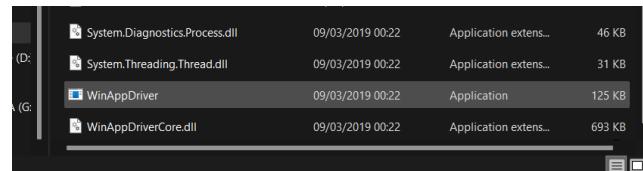
10. Kemudian klik install.



11. Tunggu sampai proses installasi selesai, lalu klik finish.



12. Berikut merupakan tampilan WinAppDriver yang sudah di install, dan tersimpan pada directory komputer. Langkah selanjutnya adalah melakukan start terhadap WinAppDriver.



13. Start WinAppDriver bisa dilakukan dengan menggunakan CMD. Copy file path WinAppDriver, lalu paste ke CMD, kemudian klik enter.

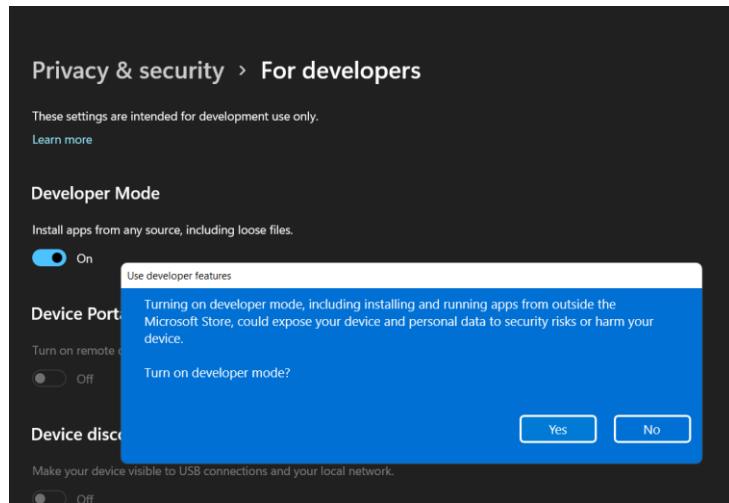
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Developer mode is not enabled. Enable it through Settings and restart Windows Application Driver
Failed to initialize: 0x80004005

C:\Program Files (x86)\Windows Application Driver>
```

Hasil di atas menunjukkan bahwa WinAppDriver belum berhasil di start, anda dapat mengatasinya dengan mengaktifkan terlebih dahulu developer mode yang ada pada windows anda.

14. Anda dapat mengaktifkan developer mode pada developer setting, seperti yang terlihat pada gambar di bawah ini.

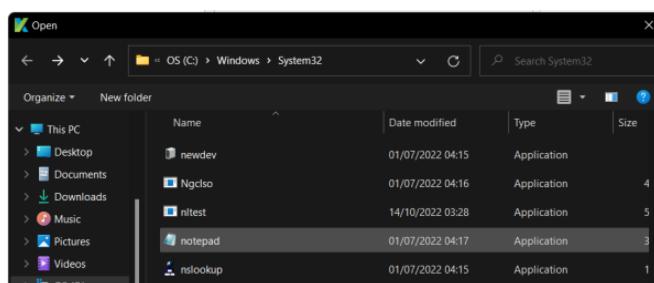


- Setelah developer mode berhasil di aktifkan, kembali ke halaman cmd kemudian lakukan kembali proses start pada WinAppDriver. Pada kasus ini WinAppDriver sudah berhasil di start, dengan hasil yang ditampilkan seperti gambar di bawah ini.

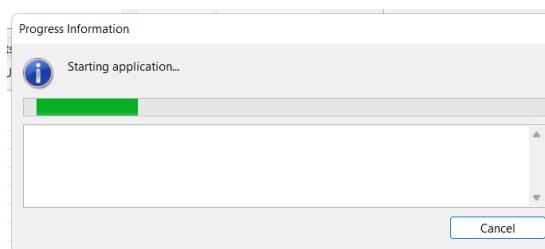
```
C:\Windows\System32\cmd.exe - WinAppDriver
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Windows Application Driver listening for requests at: http://127.0.0.1:4723/
Press ENTER to exit.
```

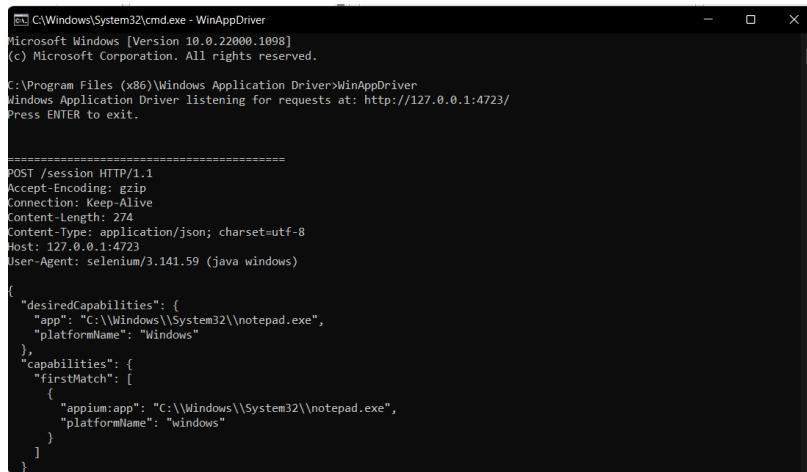
- Selanjutnya pada katalon studio pilih application dengan notepad.exe, lalu klik open.



- Setelah itu klik tombol start, dan tunggu sampai proses start selesai.



18. Kembali ke halaman cmd, kemudian klik enter. Setelah tombol enter di klik, maka user akan disuguhkan dengan tampilan jendela notepad yang ada pada sistem kita.



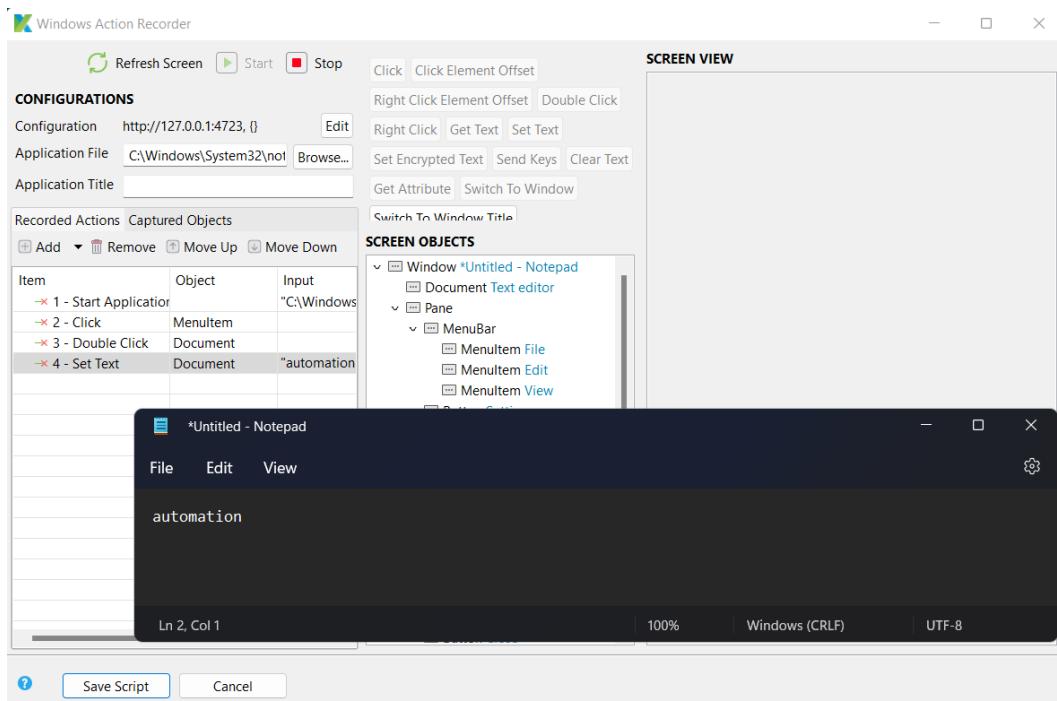
```
C:\Windows\System32\cmd.exe - WinAppDriver
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Windows Application Driver>WinAppDriver
Windows Application Driver listening for requests at: http://127.0.0.1:4723/
Press ENTER to exit.

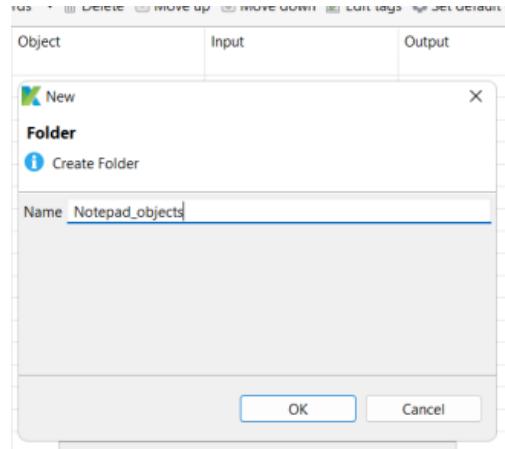
=====
POST /session HTTP/1.1
Accept-Encoding: gzip
Connection: Keep-Alive
Content-Length: 274
Content-type: application/json; charset=utf-8
Host: 127.0.0.1:4723
User-Agent: selenium/3.141.59 (java windows)

{
  "desiredCapabilities": {
    "app": "C:\Windows\System32\notepad.exe",
    "platformName": "Windows"
  },
  "capabilities": {
    "firstMatch": [
      {
        "appium:app": "C:\Windows\System32\notepad.exe",
        "platformName": "windows"
      }
    ]
  }
}
```

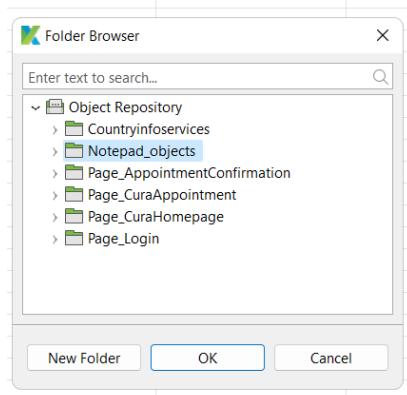
19. Selanjutnya, anda sudah dapat melakukan pengujian terhadap dekstop. Pada menu possible action, user bebas untuk melakukan proses yang ada didalamnya. Untuk kasus ini user melakukan proses click, double click, dan proses untuk menambahkan text dengan kata “automation”. Jika proses pengujian selesai dilakukan, anda dapat klik tombol stop.



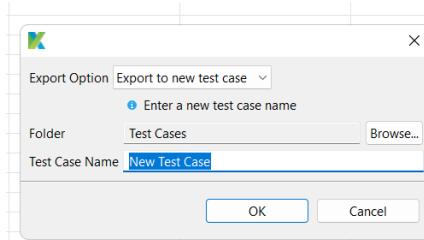
20. Kemudian, pada objek repository buat satu folder baru dengan nama Notepad_objects, lalu klik ok.



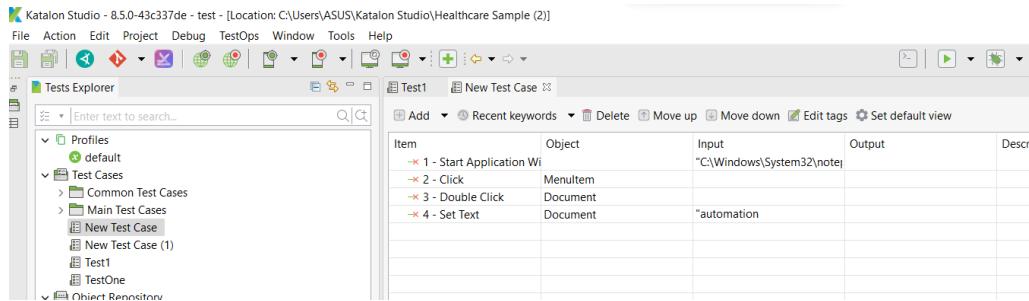
21. Folder Notepad_objects berhasil dibuat.



22. Pada folder Notepad_object dapat dilihat objek dekstop yang telah dibuat, kemudian, buat satu test case baru, lalu klik ok.



23. Pada test case tersebut dapat dilihat kasus uji yang telah direkam dan telah dilakukan sebelumnya. Seperti yang terlihat pada gambar di bawah ini.

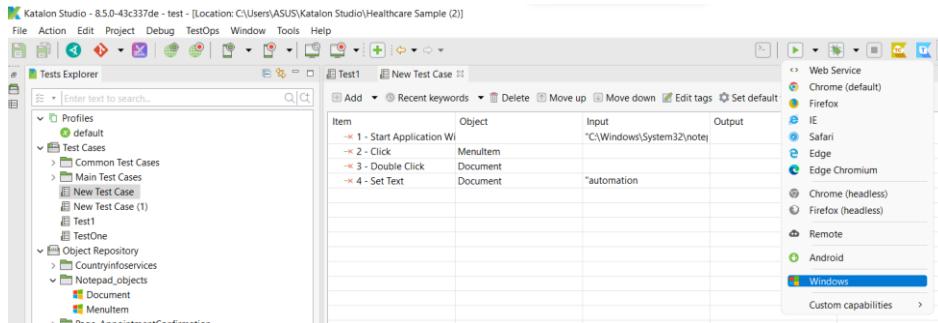


24. Kemudian jika kita lihat pada jendela script juga tertera proses-proses apa saja yang telah dilakukan saat melakukan pengujian.

```

import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.coretestdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import static com.kms.katalon.core.testobject.ObjectRepository.findWindowsObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.cucumber.keyword.CucumberBuiltInKeywords as CucumberKW
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.core.testdata.TestData as TestData
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WBSBuiltInKeywords as WS
import com.kms.katalon.core.windows.keyword.WindowsBuiltInKeywords as Windows
import internal.GlobalVariable as GlobalVariable
import org.openqa.selenium.Keys as Keys
Windows.startApplicationWithTitle('C:\Windows\System32\notepad.exe', '')
Windows.click(findWindowsObject('Object Repository/Notepad_objects/MenuItem'))
Windows.doubleClick(findWindowsObject('Object Repository/Notepad_objects/Document'))
Windows.setText(findWindowsObject('Object Repository/Notepad_objects/Document'), 'automation\r\n')
    
```

25. Selanjutnya running test case untuk melihat proses eksekusinya.



26. Berikut merupakan hasil running yang telah berhasil dilakukan, dan tidak terjadi error.

