

Web Service

Prowadzący

Michał Czyżykowski

Wprowadzenie

- Kim jestem?
- Kim jesteście Wy?
- Co robicie?
- Czego oczekujecie?

Data

- [Soap UI 5.5.0](#)
- <https://www.soapui.org/downloads/soapui.html>

[git clone https://github.com/czyzyk29/SoapUI_Workshop.git](https://github.com/czyzyk29/SoapUI_Workshop.git)

- <https://hub.docker.com/repository/docker/czyzyk/car>

Wprowadzenie

- Co to Usługi? I dlaczego są fajne?
- API, REST, SOAP?

Restauracja as service

- Stolik
 - Kelner
 - Kuchnia
-
- Żądania / Odpowiedzi (Requests / Responses)
 - Kody odpowiedzi (HTTP) 200, 404, 500...

Kino - service

- Klient rezerwuje/kupuje bilet w aplikacji web
- Użytkownik wykonuje akcje na GUI – usługi zaszyte zwracają dane o wolnych miejscach w sali na konkretna godzinę i film
- Po rezerwacji następuje zakup – klient przenoszony jest do okna płatności które są wystawionymi interfejsami bankowymi -> kolejne usługi
- Usługi mogą być re-używalne w innych miejscach i u innych partnerów

SOA - Service Oriented Architecture

- architektura zorientowana na serwisy
- może być używana niezależnie od innych
- usługi są dostępne w sieci
- wiedza o interface, a nie o implementacji usługi
- niezależne od języka i systemu operacyjnego
- możliwe połączenia wiele do wiele (brak spójności możliwy)

Usługi - WebService

- Client <-> Server
- Usługi są dostępne w sieci
- Niezależne od języka, systemu operacyjnego i sprzętu
- Dla aplikacji internetowych i systemów rozproszonych
- Usługi rejestrowane jako płatne
- dostępny przez protokoły sieciowe HTTP, SMTP, FTP
- implementacja przez WSDL / WADL - SOAP REST – dostawca nie musi znać GUI

Usługi - Webservice

- Google Maps
- Youtube
- FB



SOAP

- Simple Object Access Protocol
- Protokół do komunikacji z usługami internetowymi -> HTTP
- Wykorzystuje XML
- Przesyłanie standardów SOAP

SOAP – przykłady

- <http://www.dneonline.com/calculator.asmx?wsdl>
- <http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL>
- <http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL>

REST

- Representational State Transfer
- Luźny styl 😊
- Wszystko czego potrzebujesz żeby przesyła dane od A do B i dostać odpowiedź
- Jasne i proste
- Wykorzystuje XML, Json, Yaml
- Metody GET/POST/DELETE/PUT

REST

<https://gorest.co.in/>

<https://jsonplaceholder.typicode.com/>

<https://httpbin.org/>

<https://reqres.in/>

<https://petstore.swagger.io/#/> GET <http://petstore.swagger.io/v2/pet/id>

`docker pull czyzyk/car:newone`

`docker run -p 8080:80 czyzyk/car:newone`

<http://localhost:8080/>

REST vs SOAP

REST



SOAP



REST vs SOAP

- REST
 - Lekki, prosty bez security, otwarty, cache
- SOAP
 - Protokół, ciężki, udokumentowany, security, zamknięty, transferowany

Narzędzia

- Fiddler
- Soap UI – baza testów
- Jmeter
- Postman

Pro vs. Free

Zielone



Niebieskie



SoapUI 5.5.0

FileProjectSuiteCaseStepToolsDesktopHelp

EmptySOAPRESTImportSave AllForumTrialPreferencesProxyEndpoint Explorer

Search Forum

Workshop

Project 1

AWSECommerceServiceBinding

CalculatorSoap

Add

Divide

Request 1

Multiply

Subtract

Request 1

CalculatorSoap12

REST

WeatherSoap

WeatherSoap12

GetCityForecastByZIP

Request 1

GetCityWeatherByZIP

GetWeatherInformation

Request 1

http://thomas-bayer.com

https://localhost:5001

TestSuite 1

REST Project 1

http://localhost:8080

Car [/api/Car/4]

Car 1

Request 1

Copy of Car 1

Request 1

Interface Properties

Property	Value
Name	CalculatorSoap
Description	
Definition URL	http://www.dneonline.com/calculator...
Binding	{http://tempuri.org/}CalculatorSoap
SOAP Version	SOAP 1.1
Cached	true
Style	<not loaded>
WS-A version	NONE
WS-A anonymous	optional

Request 1

MethodGET

Endpointhttps://localhost:5001

Resource/api/Car/4

Parameters

Request 1

Request 1

Request 1

Request 1

Request 1

http://www.dneonline.com/calculator.asmx

RawXML

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"><soapenv:Header><soapenv:Body><tem:Divide><tem:intA>33</tem:intA><tem:intB>3</tem:intB></tem:Divide></soapenv:Body></soapenv:Envelope>

RawXML

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><soap:Body><DivideResponse xmlns="http://tempuri.org/"><DivideResult>11</DivideResult></DivideResponse></soap:Body></soap:Envelope>

AuthHead

response time: 178ms (338 bytes)

response time: 365ms (576 bytes)

AuthHead

response time: 178ms (338 bytes)

response time: 365ms (576 bytes)

Properties

SoapUI loghttp logjetty logerror logwsrm logmemory log

GUI – co tu widać?

- Workshop - Project
 - Soap
 - REST
 - Properties
- Editor
 - Request
 - Response
- Log

Services / TestSuites – co tu kliknąć?

- Dodawanie usługi
- Dodawanie bazy testów

Struktura drzewiasta

- Projekt
 - Service
 - REST Endpoint
 - Resource
 - Metod -> Request
 - Soap Methods
 - TestSuite -> TS
 - TestCase -> TC
 - TestStep -> TS

Proste sprawdzenia – SOAP online

- <http://www.dneonline.com/calculator.asmx?wsdl>
- Projekt dla WS SOAP
- Test Suite - Calculator
- Test Case – Add
- Test Step – Add 2+2
- Run
- Simple assertion - Contains/Http Status

Proste sprawdzenia – REST Docker

Dostępne metody w docker

- `docker pull czyzyk/car:latest`
- `docker run -p 8080:80 czyzyk/car:latest`
- <http://localhost:8080/>
- Troubleshooting:
 - Switching to Windows/Linux containers

Proste sprawdzenia – REST - docker

Dostępne metody w docker

- GET: api/Car
 - Zwróć całą listę
- GET: api/Car/3
 - Zwróć id = 3
- POST: api/Car
 - ```
{"name":"Polonez"}
```
  - dodaj pojazd nazwa „Polonez”



# Proste sprawdzenia – REST - docker

Dostępne metody w docker

- PUT: api/Car/3
  - Aktualizacja id=3
- DELETE: api/Car/3
  - Usuwanie id=3

# Poste sprawdzenia REST - online

POST <https://gorest.co.in/public-api/users> HTTP/1.1

Content-Type: application/json

Authorization: Bearer z0bROAWgHDjTBrblAgC14nRRI4IxTDTSq5h0

```
{
 "first_name": "imie",
 "last_name": "nazwisko",
 "gender": "male",
 "email": "warsztat@test.com"
}
```

# Poste sprawdzenia REST - online

- Add Oauth 2.0
  - Add pass
- Follow redirects: false

# Assertions – asercje co tu sprawdzać?

- Walidowanie
  - Rezultat vs oczekiwany wynik
- Valid HTTP Status Code (Invalid)
- Contains (not Contains)

# Assertions

- Soap Fault (Not)
- Schema Compliance
- Response SLA
- JsonPatch
  - Count
  - Existence Match
  - Match
  - RegEx match

# Assertions – Xpath Soap

- Xpath Match

```
declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';
```

```
declare namespace ns1='http://tempuri.org/';
```

```
//ns1:MultiplyResult
```

- Or `//ns1:MultiplyResult[1]` -> more then 1 to choose this if first position
- Or `// ns1:MultiplyResult/ns1:ScoreResult` -> more nodes remember about additional ns1 if then more then 1 to choose

# Assertions – XPathQuery Soap

- XQuery Match

- Check all multiply data in same nodes
- Declare weather

```
declare namespace soap='http://schemas.xmlsoap.org/soap/envelope/';
```

```
declare namespace m='http://www.oorsprong.org/websamples.countryinfo';
```

```
<Result>
```

```
{
```

```
for $a in
```

```
//m:FullCountryInfoAllCountriesResponse/m:FullCountryInfoAllCountriesResult/m:tCountry
Info/m:sName/text()
```

```
return $a
```

```
}
```

```
</Result>
```

# Assertions

- Script – Groovie SOAP

```
def groovyUtils = new com.eviware.soapui.support.GroovyUtils(context)
def holder = groovyUtils.getXmlHolder(messageExchange.responseContent)
holder.namespaces["ns"] = 'http://tempuri.org/';
def resultAdd = holder.getNodeValue("//ns:MultiplyResult")
assert resultAdd == "4"
```



# Assertions - REST

- Script – Groovie REST

```
import groovy.json.JsonSlurper
def resp = messageExchange.response.responseContent
def js = new JsonSlurper().parseText(resp)
assert js.name[0] == "Porsche"
```

# Properties – parametry

- Global  
\${#Global#parameter}
- Project  
\${#Project#parameter}
- TestSuite  
\${#TestSuite#parameter}
- TestCase  
\${#TestCase#parameter}
- TestStep - properties  
\${Properties#test}

# Property Transfer – przekazywanie w SOAP

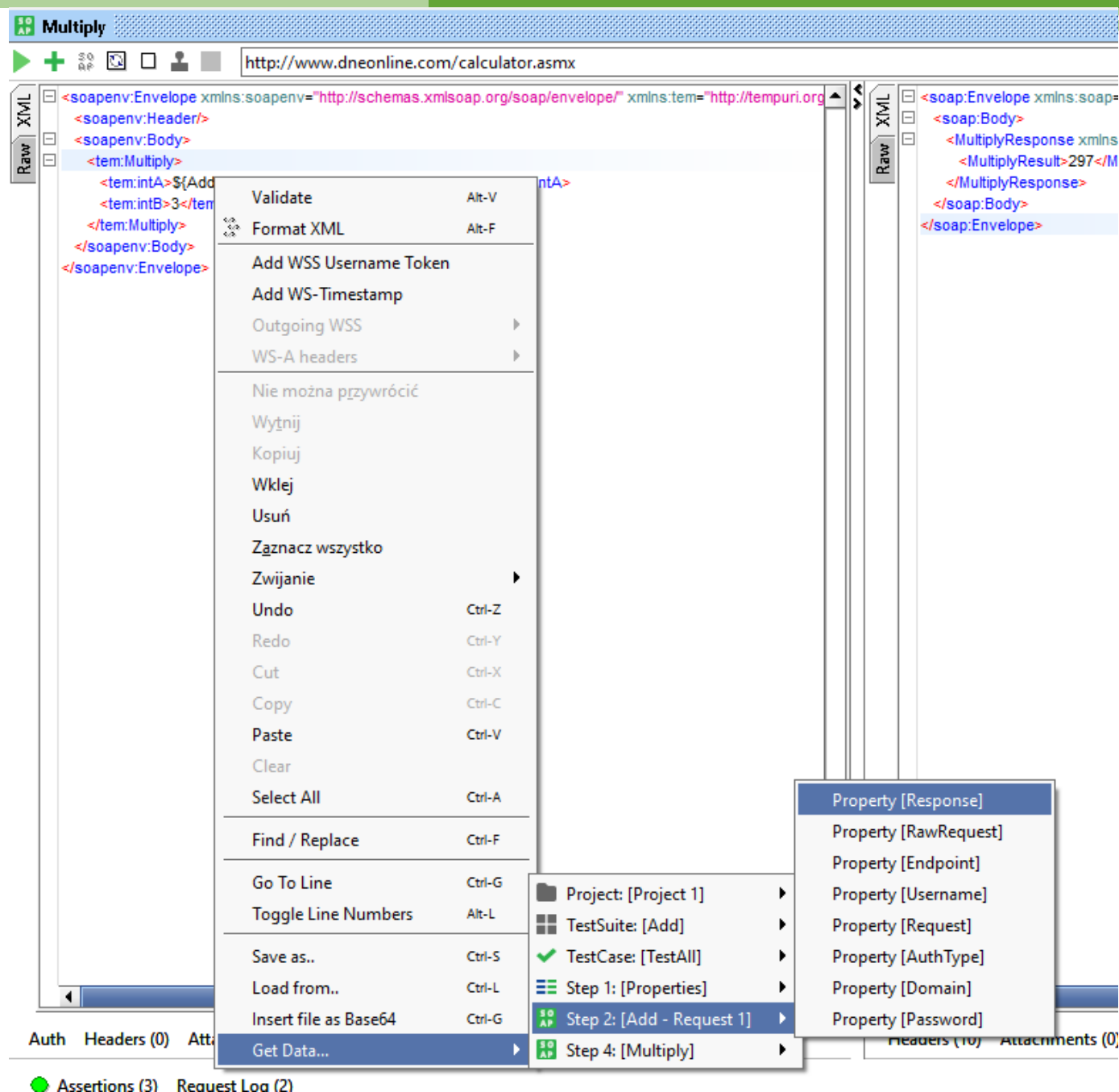
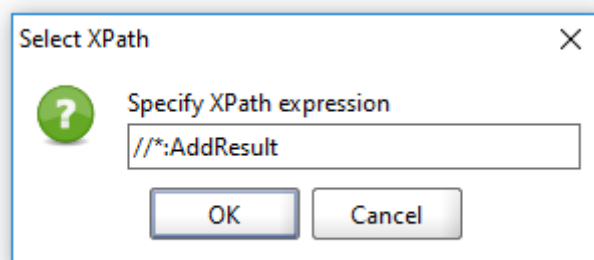
- Xpath -> helpers 😊
  - <https://xmlgrid.net/xpath.html> <-BEST
  - <http://xpather.com/>
  - <https://codebeautify.org/Xpath-Tester>
- Soap
  - Source From -> Target To
  - XPath
  - Property
  - ns – name spaces
  - XPath
    - `//*:AddResult`

# Property Transfer – przekazywanie w REST

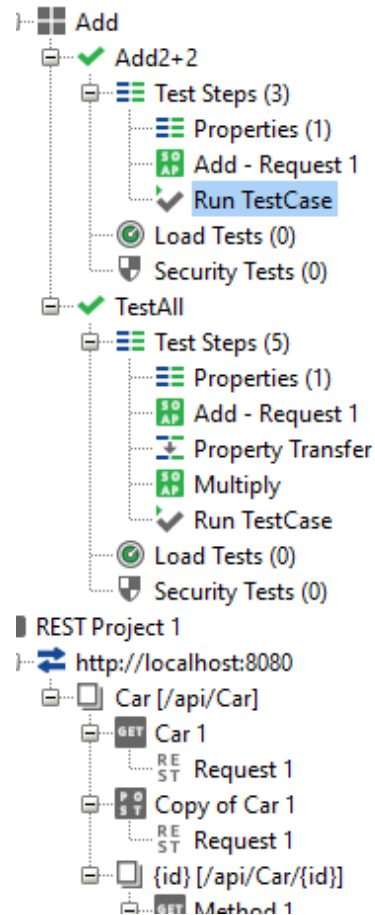
- JSONPath Finder – chrome addon helps 😊
- REST
  - Source From -> Target To
  - JsonPath
  - Property
  - [1].id

# Get Data

- If you know Xpath expression



# Run TestCase – uruchamianie innego TC



The 'Run TestCase Options' dialog box is shown. It has a title bar with a question mark icon and a close button. The main title is 'Run TestCase Options' and the subtitle is 'Set options for the Run TestCase Step below'. The options are as follows:

- Target TestCase: TestAll (dropdown)
- Target TestSuite: Add (dropdown)
- Return Properties: Select all Unselect all (radio buttons)
- Run Mode: ☒ Create isolated copy for each run (Thread-Safe) ☐ Run primary TestCase (fail if already running) ☐ Run primary TestCase (wait for running to finish, Thread-Safe)
- Copy LoadTest Properties: ☐ Copies LoadTest related properties to target context
- Copy HTTP Session: ☐ Copies HTTP Session to and from the target TestCase
- Ignore Empty Properties: ☒ Does not set empty TestCase property values

At the bottom, there is a list of properties with checkboxes. The 'number' property is checked. At the bottom right, there are 'OK' and 'Cancel' buttons.

# JDBC - docker

- `docker pull mysql`
- `docker run -p 3306:3306 --name=mysqltest -e MYSQL_ROOT_PASSWORD=pass -d mysql`
- Launch HeidiSQL / or skip to next page
  - <https://www.heidisql.com/download.php?download=portable-64>

## JDBC – no heidi

```
docker exec -it mysqltest /bin/bash
mysql -uroot -ppass
(...) see on the next page
```



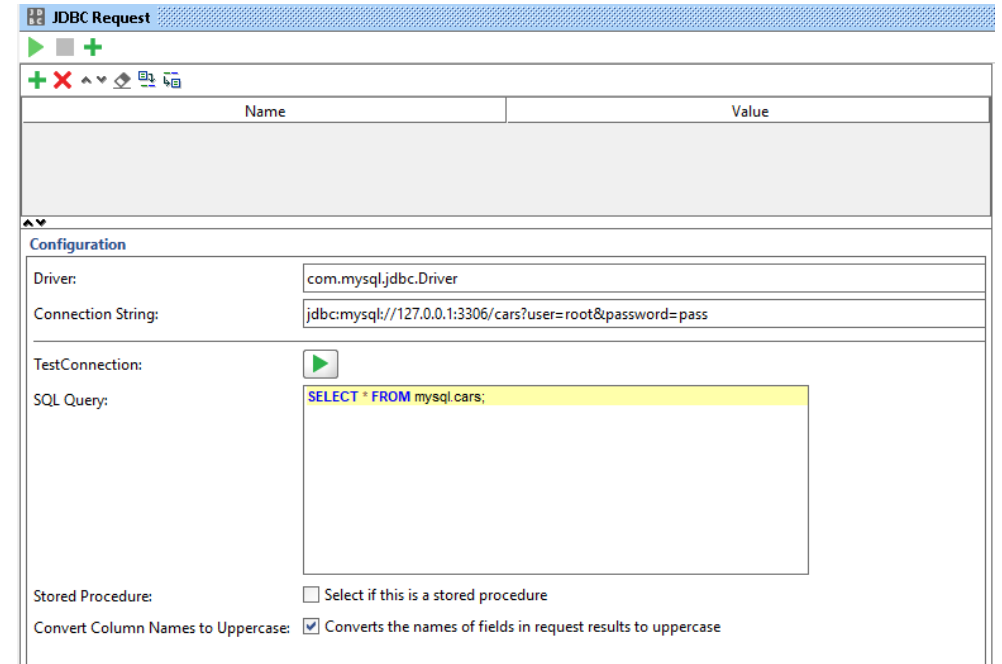
# JDBC – Create Table

```
USE mysql;
CREATE TABLE cars (
ID INT AUTO_INCREMENT PRIMARY KEY,
carname VARCHAR(200),
prodyear VARCHAR(50)
);
INSERT INTO cars (carname, prodyear)
VALUES ('Polo', '1999');
INSERT INTO cars (carname, prodyear)
VALUES ('Golf', '2003');
```

Line by line

# JDBC – Driver & Select

- Add mysql-connector-java-5.1.48.jar
  - SoapUI-5.5.0\bin\ext
- com.mysql.jdbc.Driver
- jdbc:mysql://127.0.0.1:3306?user=root&password=pass
- SELECT \* FROM mysql.cars;



The screenshot shows the 'JDBC Request' configuration window in SoapUI. It features a table with 'Name' and 'Value' columns. Below the table is a 'Configuration' section with the following fields:

- Driver:** com.mysql.jdbc.Driver
- Connection String:** jdbc:mysql://127.0.0.1:3306/cars?user=root&password=pass
- TestConnection:** A green play button icon.
- SQL Query:** A text area containing the query `SELECT * FROM mysql.cars;`, which is highlighted in yellow.
- Stored Procedure:** A checkbox labeled 'Select if this is a stored procedure'.
- Convert Column Names to Uppercase:** A checked checkbox labeled 'Converts the names of fields in request results to uppercase'.

# JDBC – transfer

- ResponseAsXML
- XPath
- Transfer
- `//Results/ResultSet/Row[@rowNumber=2]/CARS.CARNAME`

# Groovie - script

- Log.info
- Get Properties
  - Project - `testRunner.testCase.testSuite.project.getPropertyValue("PropName")`
  - TestSuite - `testRunner.testCase.testSuite.getPropertyValue("PropName")`
  - TestCase - `testRunner.testCase.getPropertyValue("PropName")`
  - TestStep
    - `testRunner.testCase.getTestStepByName("TCName"). getPropertyValue("PropName")`

# Groovie - script

- Set Properties
  - Project
    - `testRunner.testCase.testSuite.project.setPropertyValue("PropName", "PropValue")`
  - TestSuite
    - `testRunner.testCase.testSuite.setPropertyValue("PropName", "PropValue")`
  - TestCase – 😊
  - TestStep
    - `testRunner.testCase.getTestStepByName("TCName"). setPropertyValue("PropName", "PropValue")`

# Groovie – script

- If else
  - *if ()*  
*{}*  
*Else {}*
- For / while  
*(0..3).each {*  
*log.info "it"*  
*}*
- *Collection colection = new String[2]*

# Groovie – run TestStep, TestCase

- Run Test Step

```
def testStep = testRunner.testCase.testSteps['Run TestCase']
testStep.run(testRunner, context)
```

- Run Test Case

```
project =
testRunner.getTestCase().getTestSuite().getProject().getWorkspace().getProjectByName("CarAPI_REST")
testSuite = project.getTestSuiteByName("Service_Add");
testCase = testSuite.getTestCaseByName("Add Positive");
runner = testCase.run(new
com.eviware.soapui.support.types.StringToObjectMap(), false);
```

# Groovie – context XML (3 different ways)

```
def response = context.expand('${Add#Response}')
def xmlSlurper = new XmlSlurper().parseText(response)
log.info xmlSlurper.Body.AddResponse.AddResult
```

```
def groovyUtils = new com.eviware.soapui.support.GroovyUtils(context)
responseContent = testRunner.testCase.getTestStepByName("Add").getPropertyValue("response")
def holder = groovyUtils.getXmlHolder(responseContent)
log.info holder.getNodeValue("//*:AddResult")
```

```
def response =
testRunner.testCase.getTestStepByName('ListOfContinentsByCode').getPropertyValue("response")
def xml = new XmlSlurper().parseText(response)
def code=
xml.Body.ListOfContinentsByCodeResponse.ListOfContinentsByCodeResult.tContinent[1].sCode
```



## Groovie – context JSON

```
import groovy.json.JsonSlurper
def response = context.expand('${Car#Response}')
def jsonSlurper = new JsonSlurper().parseText(response)
```

# Groovie – script data sink file

```
def f = new File("C:\\Cars.txt")
f.eachLine() {
 log.info it
 testRunner.testCase.setPropertyValue("carname", it)
 testRunner.runTestStepByName("Car")
}
```

# Groovie – Excel Data Source JXL

```
import jxl.*
import jxl.write.*
import jxl.Workbook;

def groovyutils = new com.eviware.soapui.support.GroovyUtils(context)
def projectpath = groovyutils.projectPath

def f = new File("C:\\test.xls");
def wk = Workbook.getWorkbook(f);

def s1 = wk.getSheet(0);
def r = s1.getRows();
def c1
for(def i=0;i<r;i++)
{
 c1 = s1.getCell(0, i).getContents()
 log.info c1
}
```

# Groovie – Excel Data Sink JXL

```
import jxl.*
import jxl.write.*
import jxl.write.Label
import jxl.Workbook;
import java.io.File;
import groovy.json.*

def f = new File("C:\\test.xls");
WritableWorkbook wk = Workbook.createWorkbook(new File("c:\\test.xls"))
WritableSheet sheet = wk.createSheet("Worksheet", 0)

testRunner.runTestStepByName("car")
def response = context.expand('${car#Response}').toString()
def json = new JsonSlurper().parseText (response)
def carName = json.name
Label label = new Label(0, 0, carName); //col=0=A,row=0=1
sheet.addCell(label);
wk.write()
wk.close()
```

# Groovie – CSV

- \bin\ext -> opencsv2.3.jar & groovycsv-1.1.jar & restart soapUI

```
import static com.xlson.groovycsv.CsvParser.parseCsv
def csv = '''id,carName,prodyear
1,Saab,2009
2,Opel,2005'''
def data = parseCsv(csv)
for(line in data) {
 log.info "$line.id $line.carName $line.prodyear"
}
```

# Groovie – JDBC

```
com.eviware.soapui.support.GroovyUtils.registerJdbcDriver("com.mysql.jdbc.Driver")
```

```
import groovy.sql.Sql
```

```
def sql = Sql.newInstance('jdbc:mysql://127.0.0.1:3306', 'root', 'pass',
'com.mysql.jdbc.Driver')
```

```
def query = sql.rows("select * from mysql.cars")
query.each { it ->
 log.info it.id + " " + it.carname
}
```

# Groovie – JsonSlurper read json

- <https://groovy-lang.org/json.html>

```
import groovy.json.*
def response = context.expand('${AddCAR#Response}').toString()
def json = new JsonSlurper().parseText (response)
log.info json.name
```

# Groovie – JsonSlurper build json

```
import groovy.json.*
```

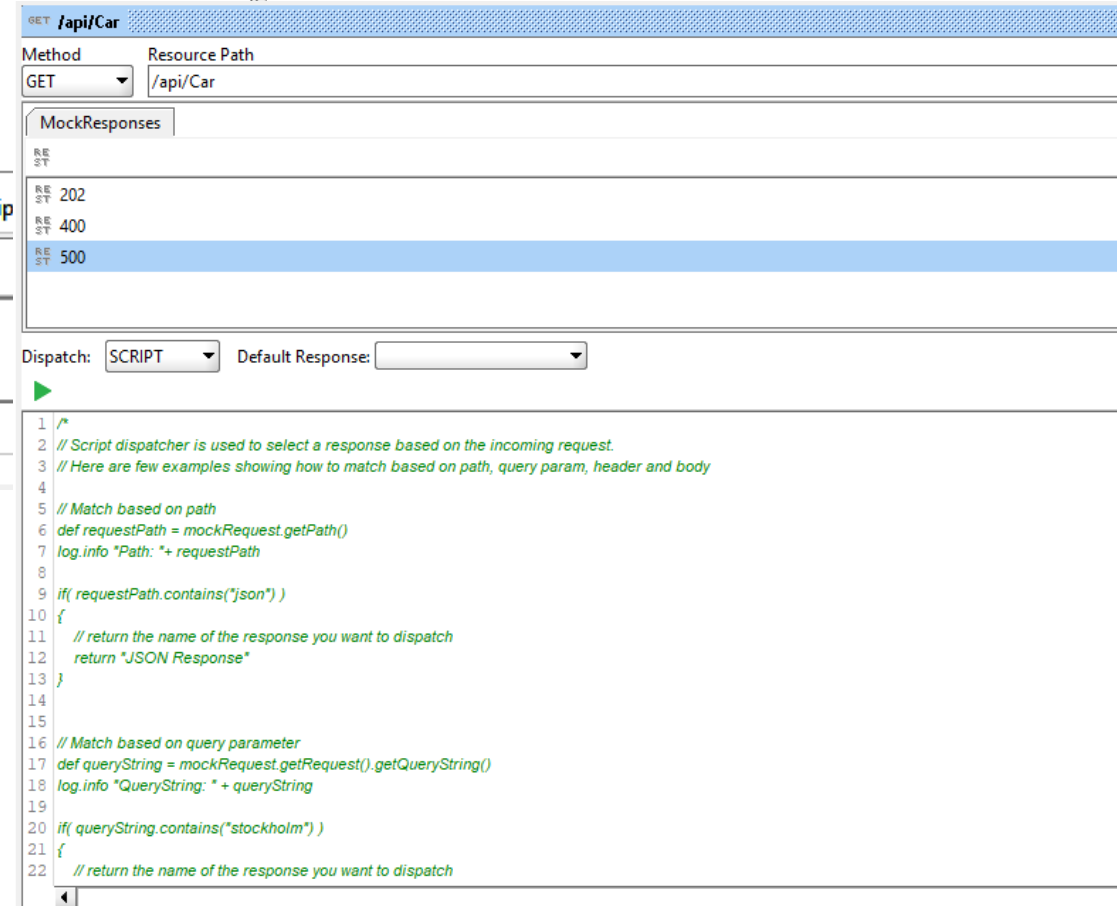
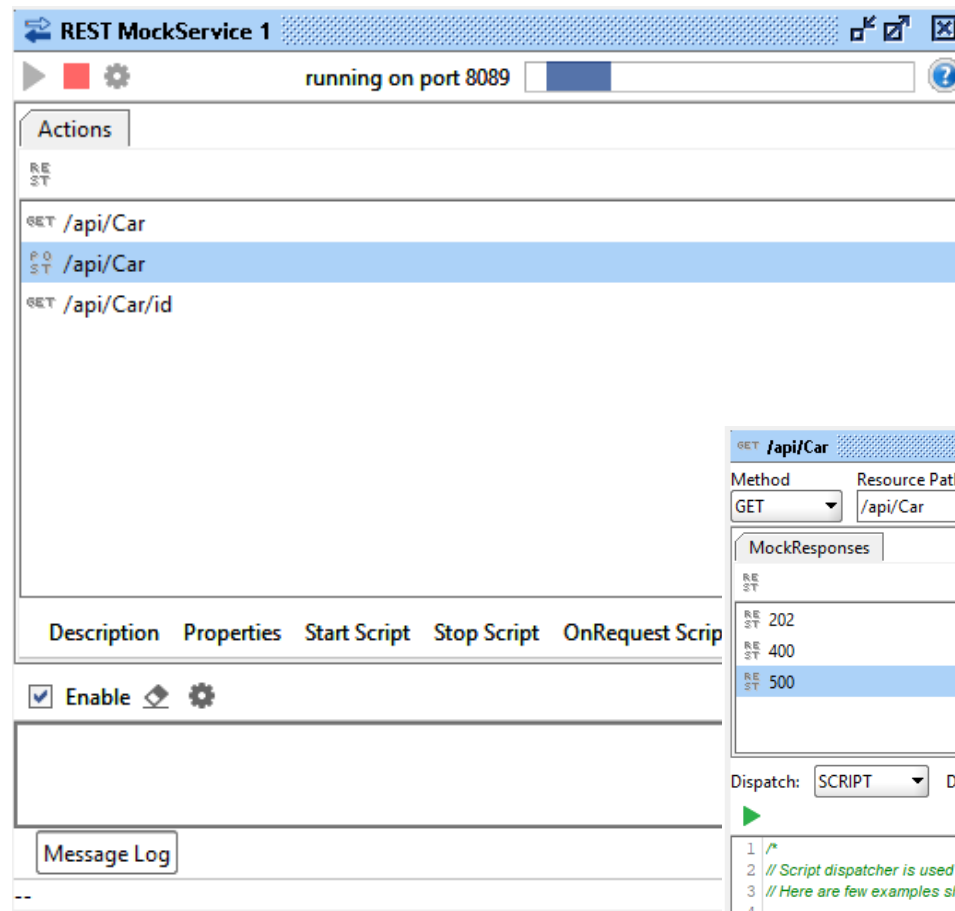
```
def jsonSlurper = new JsonSlurper()
def object = jsonSlurper.parseText '''
{
 "id": 1,
 "nameCar": "fiat",
 "prodYear": "1987"
}'''
```



# Dokumentacja / Usługi

- Dokumentacja <> Service
- GUI <> Dokumentacja
- GUI == Service
- F12 -> filtr XHR

# Mocks



# Mocks – scripts

```
// Match based on body
def requestBody = mockRequest.getRequestContent()
log.info "Request body: " + requestBody

if(requestBody.contains("Porsche"))
{
 // return the name of the response you want to dispatch
 return "Response 1"
}
else
 return "Response 2"
```

# Enviroments

- Add folder and file
  - config/config.ini
  - endpoint = endpoint:port
- Project -> Load Script

# Enviroments

REST Project 1

Overview TestSuites WS-Security Configurations Security Scan Defaults

**Project Summary**

File Path <C:\Users\michalcz\Documents\SoapUIProjects\REST-Project-1-soapui-project.xml>

**Interface Summary**

<http://localhost:8080> <http://localhost:8080.wadl>

**Test Summary**

TestSuites	2
TestCases	2
TestSteps	4
Assertions	7

Edit ▼ Load Script is invoked with log, project variables

```
1 import com.eviware.soapui.SoapUI
2 import com.eviware.soapui.model.testsuite.TestRunner
3
4 def utils = new com.eviware.soapui.support.GroovyUtils(context)
5
6 // load config file
7 def pathConfig = utils.projectPath + "/config/"
8 def config = new java.util.Properties()
9 config.load(new java.io.FileInputStream(pathConfig + "config.ini"))
10
11 // set the project endpoint from the config
12 props = project
13 props.setPropertyValue("Env", config.getProperty("endpoint"))
14
15 log.info "Running test for: " + config.getProperty("endpoint");
16
17 //http://ATN0582:8089/
18 //http://localhost:8080/
19
20
```

Description Properties Load Script Save Script

# Enviroments

```
import com.eviware.soapui.SoapUI
import com.eviware.soapui.model.testsuite.TestRunner

def utils = new com.eviware.soapui.support.GroovyUtils(context)

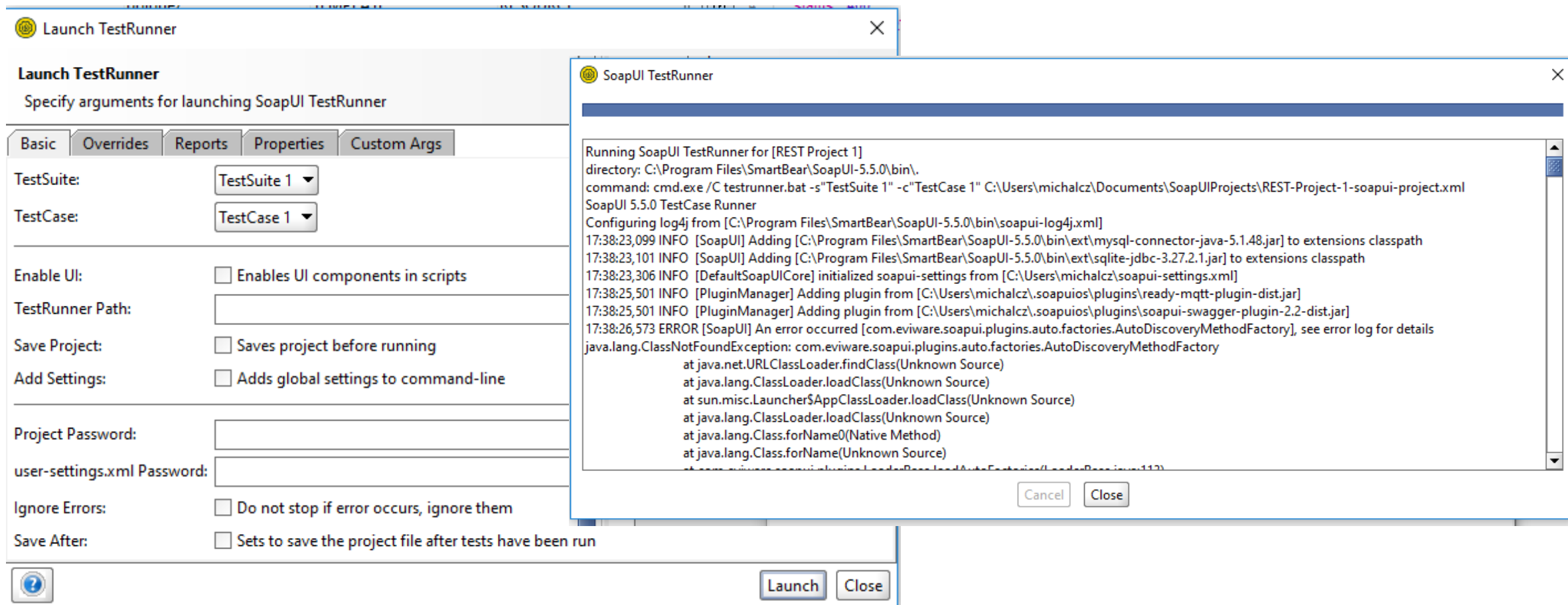
//load config file
def pathConfig = utils.projectPath + "/config/"
def config = new java.util.Properties()
config.load(new java.io.FileInputStream(pathConfig + "config.ini"))

// set the project endpoint from the config
props = project
props.setPropertyValue("Env", config.getProperty("endpoint"))

log.info "Running test for: " + config.getProperty("endpoint");

http://mock:8089/
http://localhost:8080/
```

# Testrunner - GUI



# Testrunner – Jenkins.war


- <https://jenkins.io/download/>
    - Generic Java package (.war)
    - `java -jar jenkins.war --httpPort=9292`
- `cd C:\Program Files\SmartBear\SoapUI-5.5.0\bin\  
cmd.exe /C testrunner.bat -s"TestSuite 1" -c"TestCase 1"  
C:\Users\michalcz\Documents\SoapUIProjects\REST-Project-1-soapui-  
project.xml`




# Testrunner – docker Jenkins

- Docker
  - Mkdir Jenkins\_home
  - Shared C: docker settings
  - Reset credentials
- `docker run --name Jenkins -p 8080:8080 -p 50000:50000 -v C:/Jenkins_home:/var/jenkins_home Jenkins`
- Powershell
- `set-executionpolicy remotesigned`

# Update definition

 Update Definition ✕

**Update Definition** 

Specify Update Definition options

Definition URL:  Browse...

Create New Requests: ☒ Create default requests for new methods

Recreate Requests: ☒ Recreate existing request with the new schema

Recreate Optional: ☐ Recreate optional content when updating requests


Keep Existing: ☒ Keeps existing values when recreating requests

Keep SOAP Headers: ☒ Keeps any SOAP Headers when recreating requests

Create Backups: ☒ Create backup copies of changed requests

Update TestRequests: ☐ Updates all TestRequests for operations in this Interface also


Open Request List: ☒ Opens a list of all requests that have been updated

 OK Cancel

Workshop

- Project 1
  - AWSECommerceServiceBinding**
    - ↻ BrowseNodeLookup
    - ↻ CartAdd
    - ↻ CartClear
    - ↻ CartCreate
    - ↻ Request 1
    - ↻ CartGet
    - ↻ CartModify
    - ↻ ItemLookup
    - ↻ ItemSearch
    - ↻ SimilarityLookup
  - ↻ CalculatorSoap
  - ↻ CalculatorSoap12
  - ↻ REST

Update Definition

 Update of interface successfull, [8] Requests/TestRequests have been updated.

OK

Threads: 22 Strategy Th

Test Step

all

</ns:  
</soaper  
</soapenv

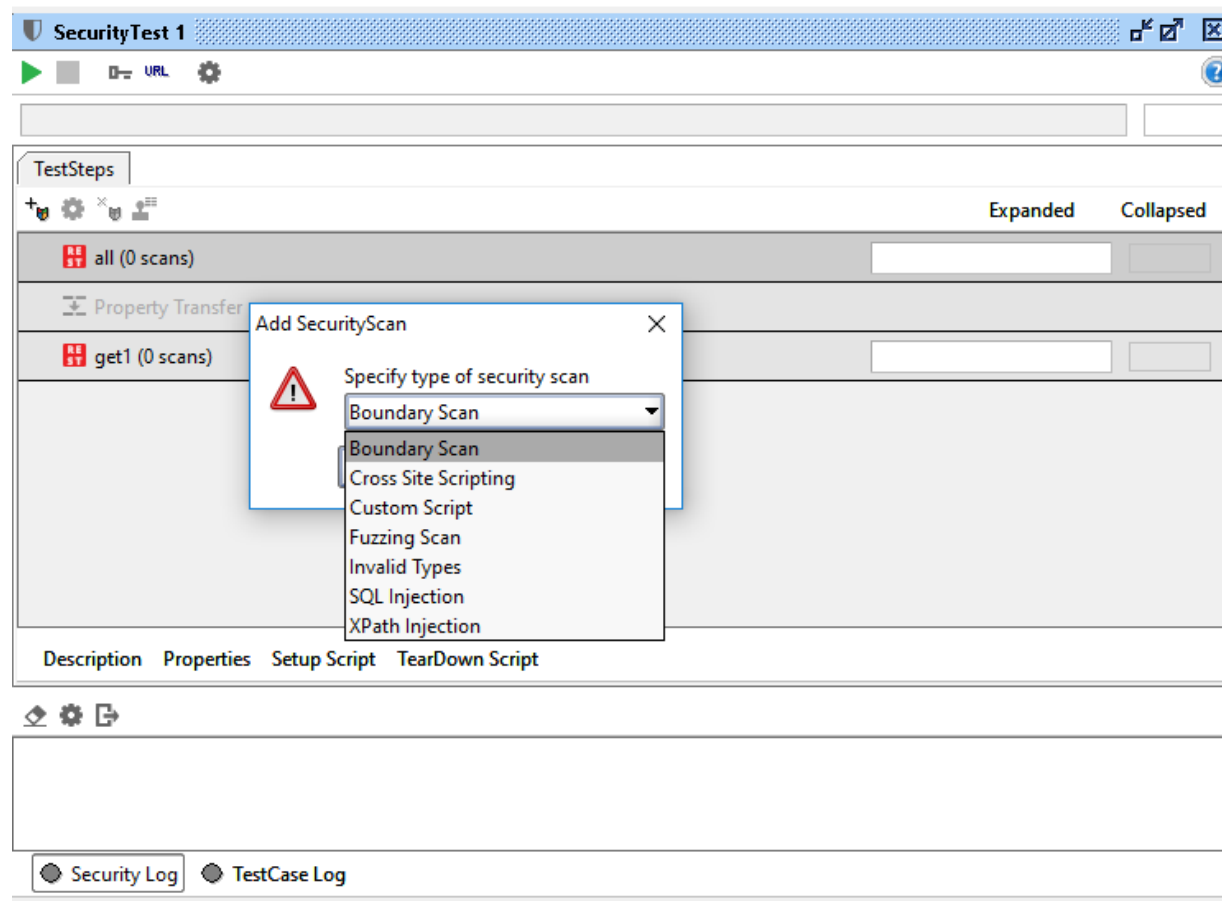
# Git

- One project in XML
- Nothing special here 😊

# Load Testing

LoadTest 1													
<div>▶ ■ 📊 📉 0.0 ⚙️ ⓘ</div> <div>Limit: 60 Seconds 11%</div>													
Threads: 22 Strategy: Variance Interval: 60 Variance: 0.5													
Test Step	min	max	avg	last	cnt	tps	bytes	bps	err	rat			
all	1	154	13,4	3	185	25,99	17575	2469	185	100			
Property Transfer	0	0	0	0	0	0	0	0	0	0			
get1	0	0	0	0	0	0	0	0	0	0			
TestCase:	1	154	13,33	0	186	28,96	17575	2736	185	99			

# Security Testing



# Dobry test

- Statusy
- Nagłówki
- Parametry
- Izolacja
- Dymne
- Regresja
- Data Driven Tests
- Integracja
- 1 Test vs 1 Assert

# Pro vs. Free

Zielone



Niebieskie



That's all folks!!!