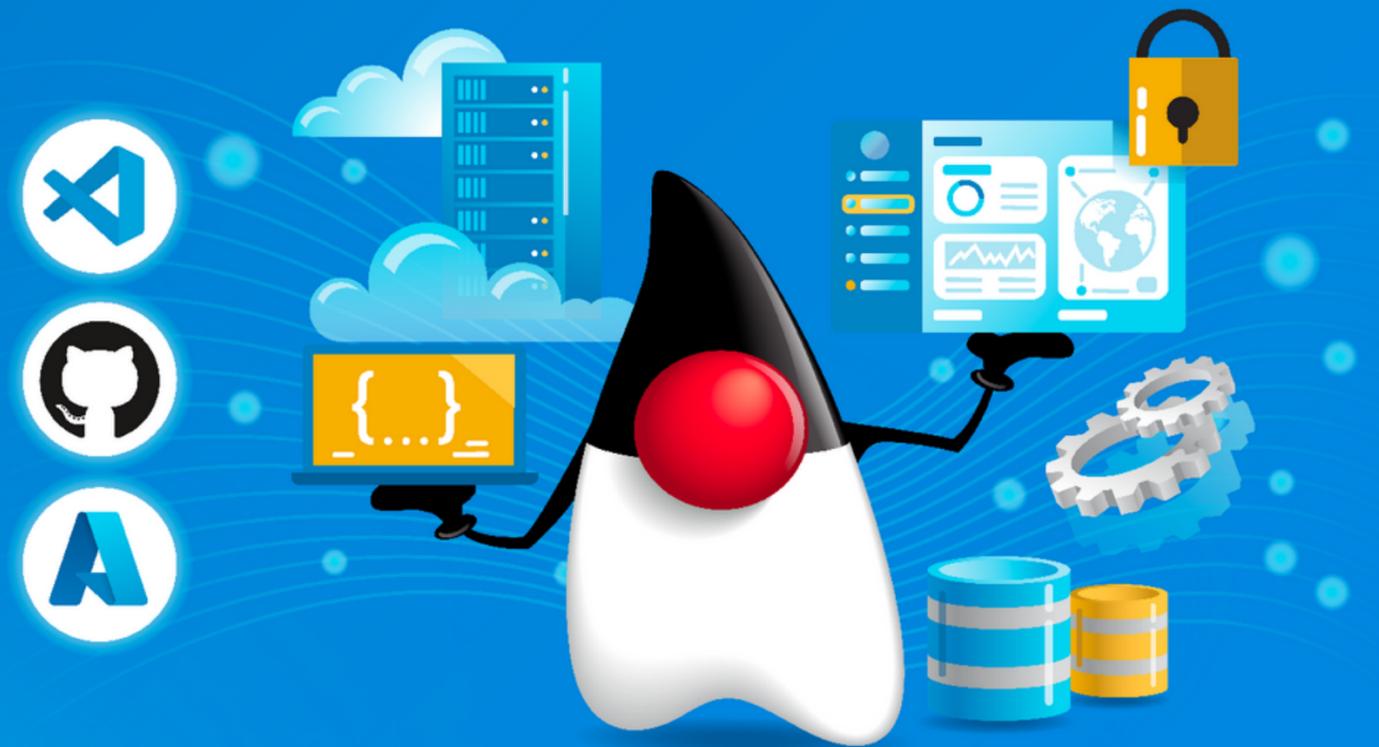


# Simple and scalable performance testing with JMeter DSL

Improving the DX



<https://charlyautomatiza.tech>

# Carlos Gauto

- 17+ years of IT experience
- Project Leader
- Playwright Ambassador
- K6 Champion
- SelectorsHub Ambassador
- Technology Content Creator



@Char\_Automatiza



CharlyAutomatiza



CharlyAutomatiza

@CharlyAutomatiza | @char\_automatiza

# What is JMeter?

A look at what we could do and what it cost us a little.

# JMeter - what its user interface looks like

The screenshot displays the Apache JMeter (5.6.3) interface. The title bar shows the file path: `oapi_onboarding.jmx (C:\ws\dev\jmeter\oapi_onboarding.jmx) - Apache JMeter (5.6.3)`. The menu bar includes File, Edit, Search, Run, Options, Tools, and Help. The toolbar contains various icons for file operations, execution, and help.

The left sidebar shows a tree view of the test plan:

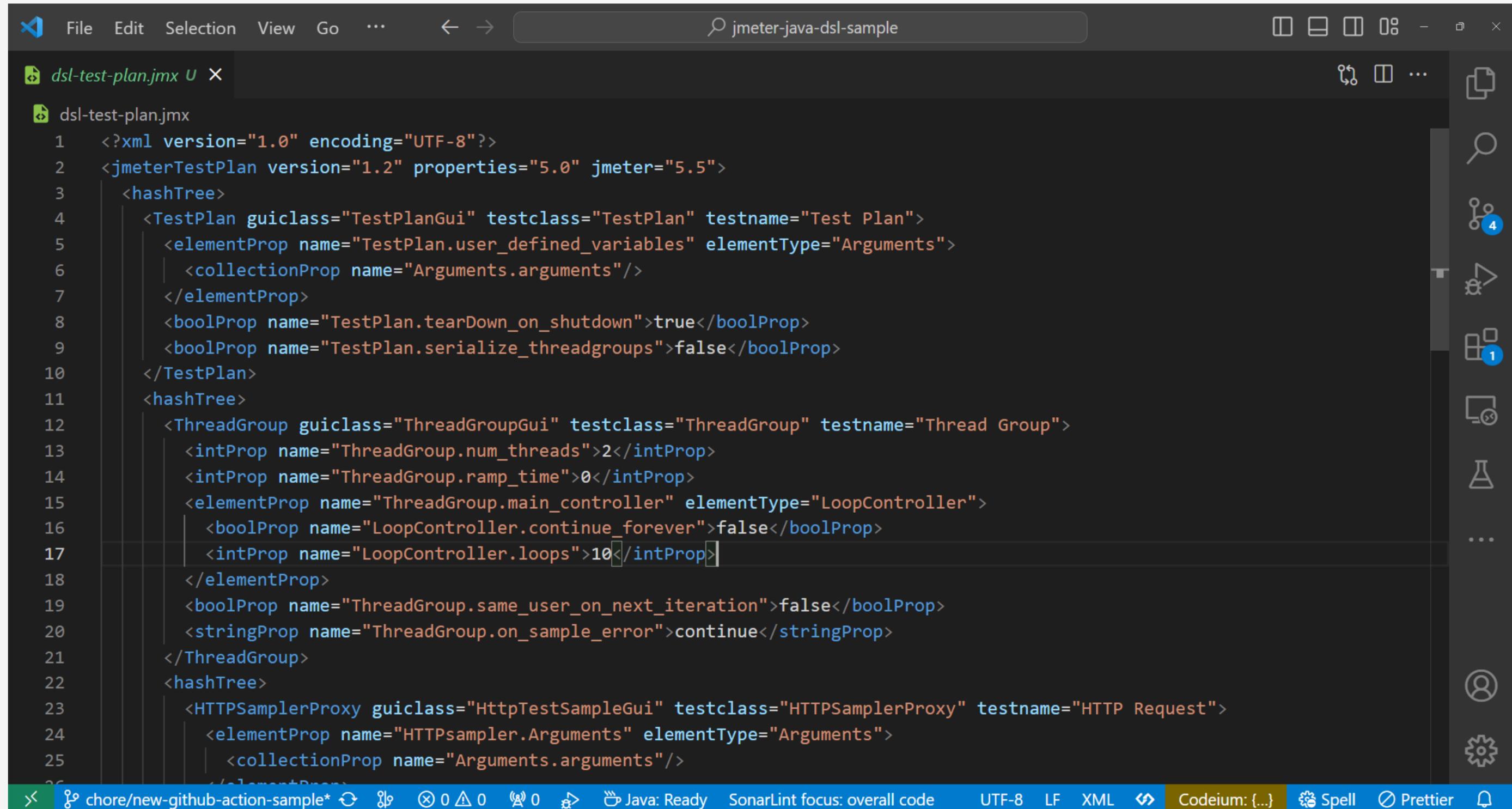
- Test Plan
  - Onboarding
    - Constant Timer
    - General Variables
    - Header Variables
    - Random data
    - Random Variables
    - 1. Onboard User**
      - HTTP Header Manager
      - Assertion - verification\_id
      - Regex Extract - verification\_id
      - Regex Extract - data
      - Regex Extract - phone\_number
    - 2. Confirm Verification
      - HTTP Header Manager
      - Assertion - access\_token
      - Regex Extract - refresh\_token
    - 3. Refresh Access Token
      - HTTP Header Manager
      - Assertion - access\_token
      - Regex Extract - access\_token
  - View Results Tree
  - Summary Report

The main panel displays the configuration for the selected '1. Onboard User' element, which is an HTTP Request. The configuration is shown in the 'Basic' tab:

- Name: 1. Onboard User
- Comments: User onboarding flow
- Web Server
  - Protocol [http]: `https`
  - Server Name or IP: `${baseUrl}`
- HTTP Request
  - Method: `POST`
  - Path: `${users}`
  - Redirect Automatically
  - Follow Redirects
  - Use KeepAlive
  - Use multipart/form-data
  - Browser-compatible headers
- Body Data
  - Parameters
  - Files Upload

```
1 {
2     "dni": "${dni}",
3     "first_name": "${first_name}",
4     "last_name": "${last_name}",
5     "phone_number": "${phone_number}",
6     "email": "${email}"
7 }
```

# JMeter - what the project code looks like



The image shows a screenshot of an IDE (Visual Studio Code) displaying the XML code for a JMeter test plan. The file is named `dsl-test-plan.jmx`. The code is structured as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <jmeterTestPlan version="1.2" properties="5.0" jmeter="5.5">
3   <hashTree>
4     <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan">
5       <elementProp name="TestPlan.user_defined_variables" elementType="Arguments">
6         <collectionProp name="Arguments.arguments"/>
7       </elementProp>
8       <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
9       <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
10    </TestPlan>
11    <hashTree>
12      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Thread Group">
13        <intProp name="ThreadGroup.num_threads">2</intProp>
14        <intProp name="ThreadGroup.ramp_time">0</intProp>
15        <elementProp name="ThreadGroup.main_controller" elementType="LoopController">
16          <boolProp name="LoopController.continue_forever">false</boolProp>
17          <intProp name="LoopController.loops">10</intProp>
18        </elementProp>
19        <boolProp name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
20        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
21      </ThreadGroup>
22      <hashTree>
23        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="HTTP Request">
24          <elementProp name="HTTPSampler.Arguments" elementType="Arguments">
25            <collectionProp name="Arguments.arguments"/>
26          </elementProp>
27        </HTTPSamplerProxy>
28      </hashTree>
29    </hashTree>
30  </jmeterTestPlan>
31 </xml>
```

The IDE interface includes a menu bar (File, Edit, Selection, View, Go), a search bar, and a sidebar with various icons. The status bar at the bottom shows the current file path, Java version (Java: Ready), SonarLint focus, and various tool settings like Codeium, Spell, and Prettier.

# What is JMeter DSL?

Filling in the blanks to enhance the DX

# How to start using JMeter DSL



```
<dependency>  
  <groupId>us.abstracta.jmeter</groupId>  
  <artifactId>jmeter-java-dsl-azure</artifactId>  
  <version>1.25.3</version>  
  <scope>test</scope>  
</dependency>
```



```
testImplementation 'us.abstracta.jmeter:jmeter-java-dsl-azure:1.25.3'
```



# Hello world using JMeter DSL



```
package us.abstracta.jmeter.javadsl.sample;

import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsl.JmeterDsl.*;

import java.io.IOException;
import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsl.core.TestPlanStats;

class PerformanceTest {

    @Test
    void testPerformance() throws IOException {
        TestPlanStats stats = testPlan(
            threadGroup(8, 2,
                httpSampler("https://restful-booker.herokuapp.com/booking")))
            .run();
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

THREADS

# Hello world using JMeter DSL



```
package us.abstracta.jmeter.javadsl.sample;

import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsl.JmeterDsl.*;

import java.io.IOException;
import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsl.core.TestPlanStats;

class PerformanceTest {

    @Test
    void testPerformance() throws IOException {
        TestPlanStats stats = testPlan(
            threadGroup(8, 2,
                httpSampler("https://restful-booker.herokuapp.com/booking")))
            .run();
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

ITERATIONS

# Hello world using JMeter DSL



```
package us.abstracta.jmeter.javadsl.sample;

import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsl.JmeterDsl.*;

import java.io.IOException;
import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsl.core.TestPlanStats;

class PerformanceTest {

    @Test
    void testPerformance() throws IOException {
        TestPlanStats stats = testPlan(
            threadGroup(8, 2,
                httpSampler("https://restful-booker.herokuapp.com/booking")))
            .run();
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

← URL

# Threads configuration



*// We can use and combine methods to configure more complex scenarios*

```
threadGroup( )  
    .rampToAndHold(10, Duration.ofSeconds(5), Duration.ofSeconds(20))  
    .rampToAndHold(100, Duration.ofSeconds(10), Duration.ofSeconds(30))  
    .rampTo(200, Duration.ofSeconds(10))  
    .rampToAndHold(100, Duration.ofSeconds(10), Duration.ofSeconds(30))  
    .rampTo(0, Duration.ofSeconds(5))  
    .children(  
        httpSampler("https://restful-booker.herokuapp.com/booking")  
    )
```

# Threads configuration - Threads timeline



# Test plan debugging - Results Tree Visualizer



```
import static us.abstracta.jmeter.javads1.JmeterDsl.*;

import java.io.IOException;
import org.junit.jupiter.api.Test;

public class PerformanceTest {

    @Test
    public void testPerformance() throws IOException {
        testPlan(
            threadGroup(1, 1,
                httpSampler("https://restful-booker.herokuapp.com/booking")
            ),
            resultsTreeVisualizer() // allows us to do debugging in a simple way
        ).run();
    }
}
```

# Test plan debugging - Results Tree Visualizer

The screenshot shows a window titled "View Results Tree" with the following components:

- Name:** View Results Tree
- Comments:** (empty text field)
- Write results to file / Read from file:** Includes a "Filename" field, a "Browse..." button, and "Log/Display Only:" options for  Errors and  Successes, along with a "Configure" button.
- Search:** Includes a search input field,  Case sensitive,  Regular exp., and "Search" and "Reset" buttons.
- Tree View:** A tree view showing a "Text" node expanded to reveal an "HTTP Request" node.
- Response Data:** A detailed view of the selected node with the following information:
  - Thread Name: test 1-1
  - Sample Start: 2021-08-11 21:06:10 UYT
  - Load time: 20
  - Connect Time: 20
  - Latency: 0
  - Size in bytes: 2340
  - Sent bytes: 0
  - Headers size in bytes: 0
  - Body size in bytes: 2340
  - Sample Count: 1
  - Error Count: 1
  - Data type ("text"|"bin"|"" ): text
  - Response code: Non HTTP response code: java.net.UnknownHostException
  - Response message: Non HTTP response message: my.service: nodename nor servname provided, or not known
- Raw/Parsed:** Buttons at the bottom of the response data view.
- Scroll automatically?:** A checkbox at the bottom left of the window.

# Test plan review in JMeter GUI - Show GUI



```
import static us.abstracta.jmeter.javads1.JmeterDsl.*;

import java.io.IOException;
import org.junit.jupiter.api.Test;

public class PerformanceTest {

    @Test
    public void testPerformance() throws IOException {
        testPlan(
            threadGroup(2, 10,
                httpSampler("https://restful-booker.herokuapp.com/booking")
            )
        ).showInGui();
    }
}
```

# Test plan review in JMeter GUI

The screenshot displays the Apache JMeter (5.4.3) GUI. The interface is dark-themed and shows a test plan structure on the left sidebar. The main area is titled "HTTP Request" and is divided into "Basic" and "Advanced" tabs. The "Basic" tab is active, showing the following configuration:

- Name: HTTP Request
- Comments: (empty text area)
- Web Server section:
  - Protocol [http]: http
  - Server Name or IP: my.service
  - Port Number: (empty text box)
- HTTP Request section:
  - Method: GET
  - Path: /
  - Content encoding: (empty text box)
- Options section:
  - Redirect Automatically
  - Follow Redirects
  - Use KeepAlive
  - Use multipart/form-data
  - Browser-compatible headers
- Parameters section:
  - Send Parameters With the Request: (checked)
  - Table with columns: Name, Value, URL Encode?, Content-Type, Include Equals?

At the bottom of the configuration area, there are buttons for "Detail", "Add", "Add from Clipboard", "Delete", "Up", and "Down". The top right corner of the window shows a timer at 00:00:00, a warning icon, and a status of 0 0/0.

# JMeter DSL - Demo

Filling in the blanks to enhance the DX

```
src > test > java > us > abstracta > jmeter > javadsl > sample > PerformanceTest.java > ...
You, 7 minutes ago | 2 authors (rabelenda and others)
1 package us.abstracta.jmeter.javadsl.sample;
2
3 import static org.assertj.core.api.Assertions.assertThat;
4 import static us.abstracta.jmeter.javadsl.JmeterDsl.*;
5
6 import java.io.IOException;
7 import java.time.Duration;
8 import org.junit.jupiter.api.Test;
9 import us.abstracta.jmeter.javadsl.core.TestPlanStats;
10
11 You, 7 minutes ago | 2 authors (You and others) | Codeium: Refactor | Codeium: Explain
12 class PerformanceTest {
13     Codeium: Refactor | Explain | Generate Javadoc | X
14     @Test
15     void testPerformance() throws IOException {
16         TestPlanStats stats = testPlan(
17             threadGroup(threads:5, iterations:2,
18                 httpSampler(url:"https://restful-booker.herokuapp.com/booking")))
19             .run();
20         assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSe
21     }
22 }
```

charlyautomatiza/jmeter-java-dsl-sample

gith... | Twitch | Chrome | GDE | AUTOMATIZACIÓN | Accesos | Producción

charlyautomatiza / jmeter-java-dsl-sample

Code | Pull requests 1 | Actions

0 stars | 6 forks | 0 watching | 2 Branches

0 Tags | Activity

Public repository · Forked from abstracta/jmeter-java-dsl-sample

master | Code

This branch is up to date with abstracta/jmeter-java-dsl-sample:master

Contribute | Sync fork

rabelenda last month

src/test | 2 years ago

How to run test at scale  
with JMeterDSL?

# Run test at scale - JMeter remote testing



```
import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsl.JmeterDsl.*;

import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsl.core.engines.DistributedJmeterEngine;
import us.abstracta.jmeter.javadsl.core.TestPlanStats;

public class PerformanceTest {

    @Test
    public void testPerformance() throws Exception {
        TestPlanStats stats = testPlan(
            threadGroup(200, Duration.ofMinutes(10),
                httpSampler("https://restful-booker.herokuapp.com/booking")
            )
        ).runIn(new DistributedJmeterEngine("host1", "host2"));
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

**docker-compose sample:** <https://bit.ly/jmeter-dsl-distributed>

# Run test at scale - Azure Load Testing



```
import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsL.JmeterDsl.*;

import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsL.azure.AzureEngine;
import us.abstracta.jmeter.javadsL.core.TestPlanStats;

public class PerformanceTest {

    @Test
    public void testPerformance() throws Exception {
        TestPlanStats stats = testPlan(
            threadGroup(2, 10,
                httpSampler("https://restful-booker.herokuapp.com/booking")
            )
        ).runIn(new AzureEngine(System.getenv("AZURE_CREDS")) // AZURE_CREDS=tenantId:clientId:secretId
            .testName("dsl-test")
            .engines(2)
            .testTimeout(Duration.ofMinutes(20)));
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

# Run test at scale - Azure Load Testing



```
import static org.assertj.core.api.Assertions.assertThat;
import static us.abstracta.jmeter.javadsL.JmeterDsl.*;

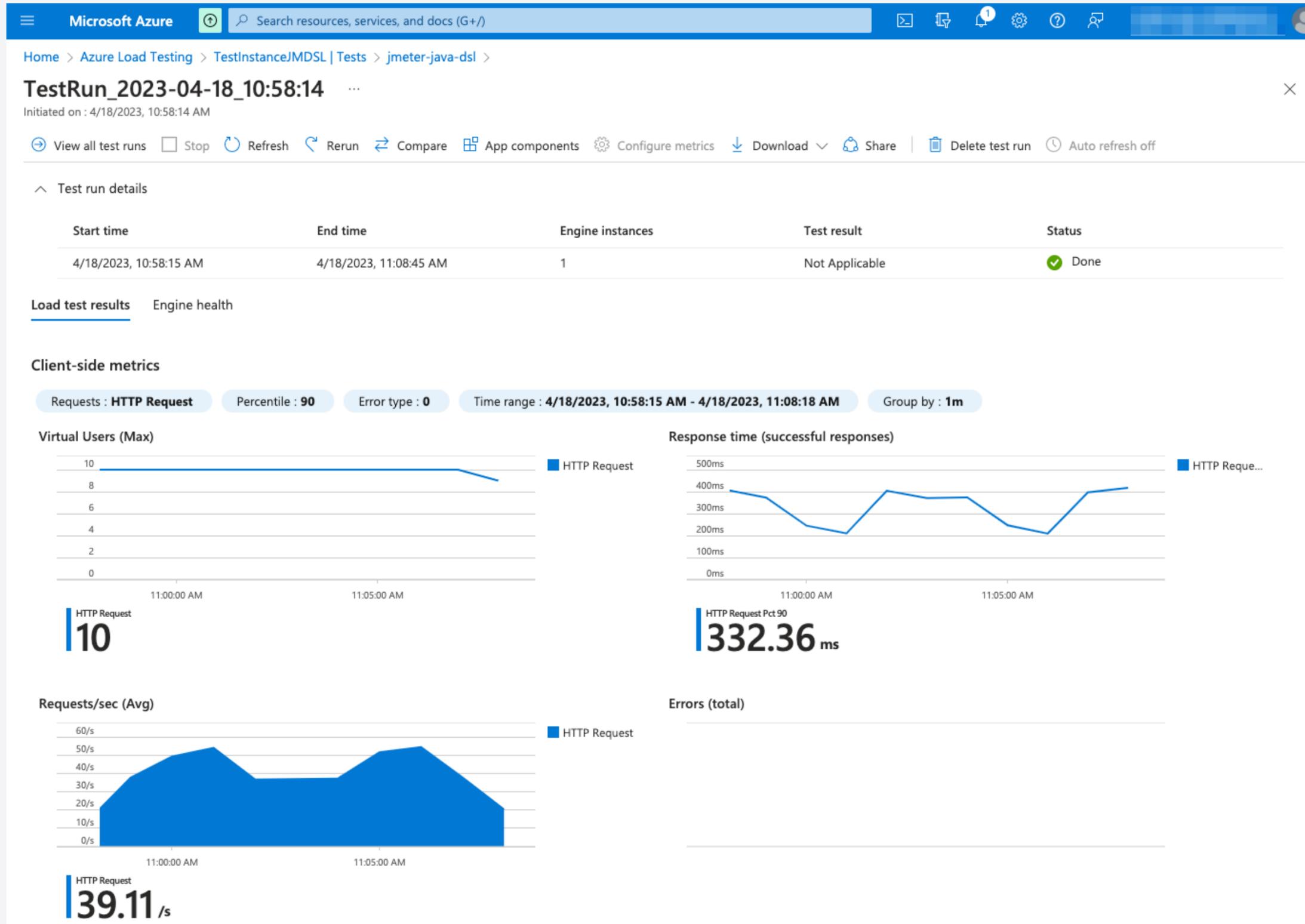
import java.time.Duration;
import org.junit.jupiter.api.Test;
import us.abstracta.jmeter.javadsL.azure.AzureEngine;
import us.abstracta.jmeter.javadsL.core.TestPlanStats;

public class PerformanceTest {

    @Test
    public void testPerformance() throws Exception {
        TestPlanStats stats = testPlan(
            threadGroup(2, 10,
                httpSampler("https://restful-booker.herokuapp.com/booking")
            )
        ).runIn(new AzureEngine(System.getenv("AZURE_CREDS")) // AZURE_CREDS=tenantId:clientId:secretId
            .testName("dsl-test")
            .engines(2)
            .testTimeout(Duration.ofMinutes(20)));
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

ENGINES - AGENTS

# Azure Load Testing - Reports



# JMeterDSL - Summary

Pros and cons

# JMeterDSL - Summary

## Pros

- 👍 IDE friendly + CI/CD integration
- 👍 Code modularization
- 👍 Support for JMeter supported protocols
- 👍 Interact with JMX files and take advantage of JMeter ecosystem
- 👍 All details of simple test plans at a glance
- 👍 Simple way to do assertions on statistics

## Cons

- 👎 Basic Java knowledge required
- 👎 Same resources (CPU & RAM) usage as JMeter

<https://charlyautomatiza.tech>

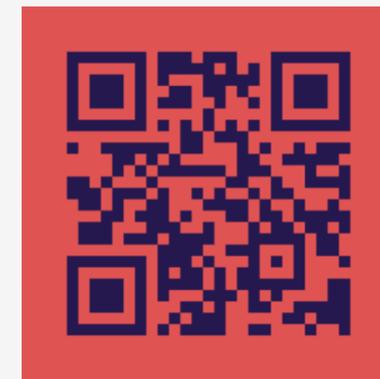
# Thank you



@Char\_Automatiza



CharlyAutomatiza



CharlyAutomatiza