

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

oapi_onboarding.jmx (C:\ws\dev\jmeter\oapi_onboarding.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

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

HTTP Request

Name: 1. Onboard User

Comments: User onboarding flow

Basic Advanced

- Web Server

Protocol [http]: https Server Name or IP: \${baseUrl}

- HTTP Request

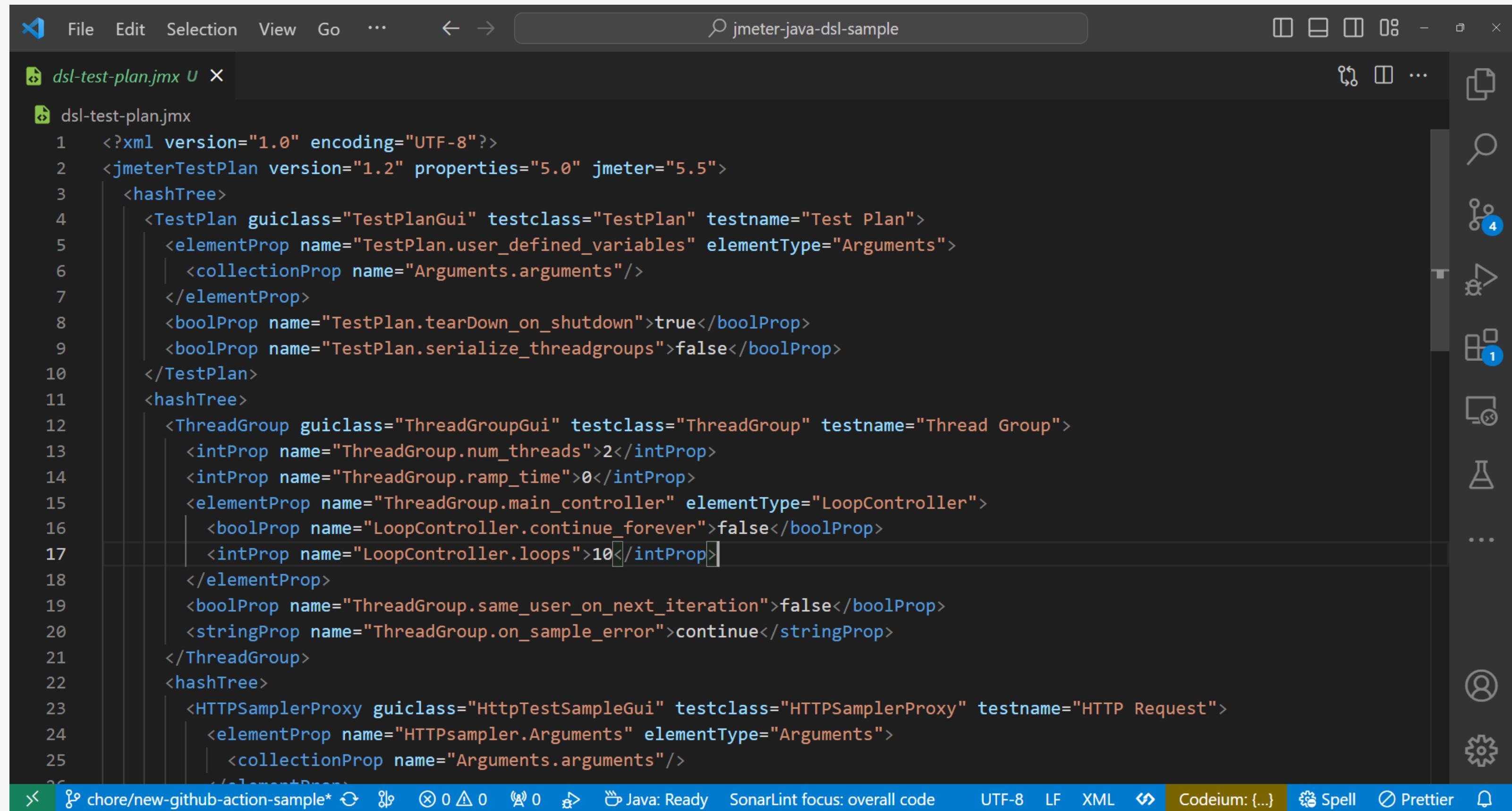
Method: POST Path: \${users}

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data 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 screenshot shows a dark-themed code editor interface with a sidebar on the right containing various icons. The main area displays an XML file named `dsl-test-plan.jmx`. The code is a JMeter test plan structure:

```
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>
```

The code editor has a status bar at the bottom with various icons and text, including "chore/new-github-action-sample*" and "Java: Ready".

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));
    }
}
```



The diagram shows a red arrow originating from the word "THREADS" and pointing towards the "threadGroup" method call in the Java code. The "threadGroup" method is part of the JMeter DSL API, used to define a group of threads for a performance test.

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

```
● ○ ●

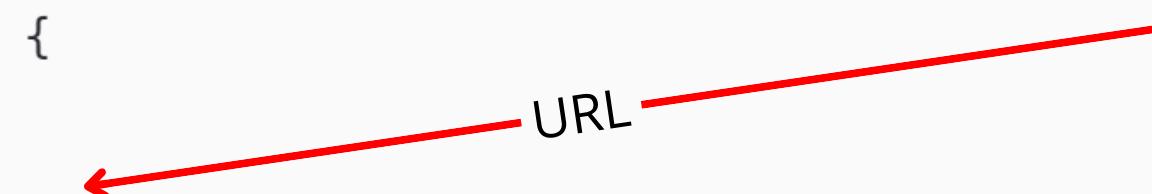
package us.abstracta.jmeter.javadsl.sample;

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")
    )
```

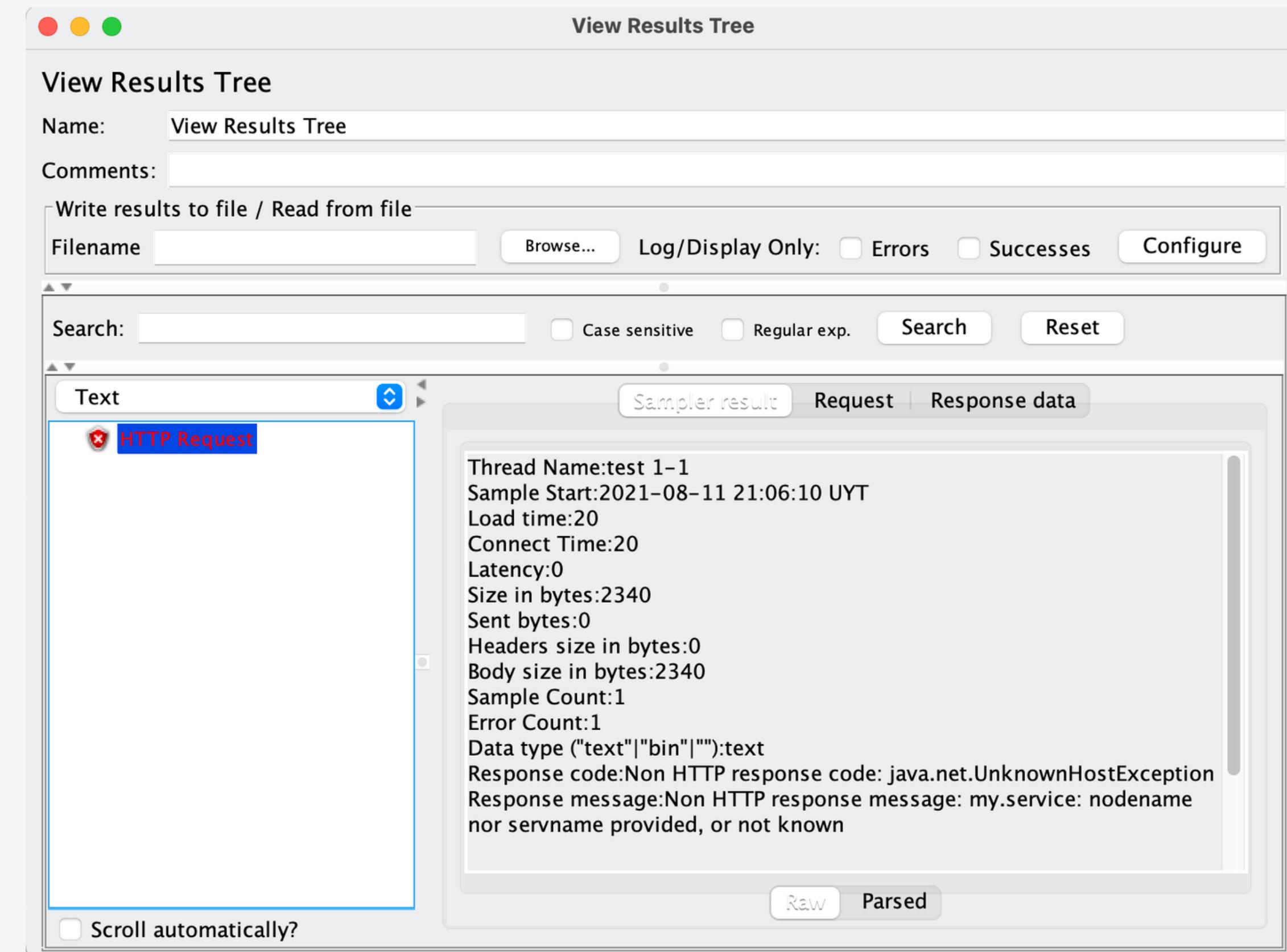
Threads configuration - Threads timeline



Test plan debugging - Results Tree Visualizer

```
import static us.abstracta.jmeter.javDSL.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



Test plan review in JMeter GUI - Show GUI



```
import static us.abstracta.jmeter.javDSL.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

Apache JMeter (5.4.3)

The screenshot shows the Apache JMeter 5.4.3 interface. The left sidebar displays a tree structure of the test plan, with 'HTTP Request' selected. The main panel shows the configuration for this selected sampler.

HTTP Request

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

Method: Path: Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Detail Add Add from Clipboard Delete Up Down

JMeter DSL - Demo

Filling in the blanks to enhance the DX

The image shows a developer's workspace with two main windows. On the left is a code editor for a Java project named 'jmeter-java-dsl-sample'. The current file is 'PerformanceTest.java'. The code defines a test class 'PerformanceTest' with a single test method 'testPerformance'. This method creates a thread group with 5 threads and 2 iterations, then runs an HTTP sampler against the URL 'https://restful-booker.herokuapp.com/booking'. Finally, it asserts that the overall sample time percentile 99 is less than 5 seconds. The code editor includes a sidebar with various icons for navigation and tools like Codeium.

The right window is a GitHub repository page for 'charlyautomatiza/jmeter-java-dsl-sample'. It shows basic repository details: Apache-2.0 license, 0 stars, 6 forks, 0 watching, and 2 branches. It also indicates that the 'master' branch is up-to-date with the forked 'abstracta/jmeter-java-dsl-sample' master branch. A recent commit by 'rabelenda' from last month is visible.

```
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  
You, 7 minutes ago | 2 authors (You and others) | Codeium: Refactor | Codeium: Explain  
11 class PerformanceTest {  
12  
    Codeium: Refactor | Explain | Generate Javadoc | X  
13     @Test  
14     void testPerformance() throws IOException {  
15         TestPlanStats stats = testPlan(  
16             threadGroup(threads:5, iterations:2,  
17                 httpSampler(url:"https://restful-booker.herokuapp.com/booking")))  
18             .run();  
19             assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSe  
20         }  
21     }  
22     Codeium: Command Generate (Ctrl+I)
```

charlyautomatiza/jmeter-java-dsl-sample

Code Pull requests Actions

jmeter-java-dsl simple sample maven project

Apache-2.0 license

0 stars 6 forks 0 watching 2 Branches

0 Tags Activity

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

master

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) ← ENGINES - AGENTS
            .testTimeout(Duration.ofMinutes(20)));
        assertThat(stats.overall().sampleTimePercentile99()).isLessThan(Duration.ofSeconds(5));
    }
}
```

Azure Load Testing - Reports

Microsoft Azure  Search resources, services, and docs (G+/-)

Home > Azure Load Testing > TestInstanceJMDSL | Tests > jmeter-java-dsl >

TestRun_2023-04-18_10:58:14 ...

Initiated on : 4/18/2023, 10:58:14 AM

 Stop  Refresh  Rerun  Compare  App components  Configure metrics  Download  Share  Delete test run  Auto refresh off

Test run details

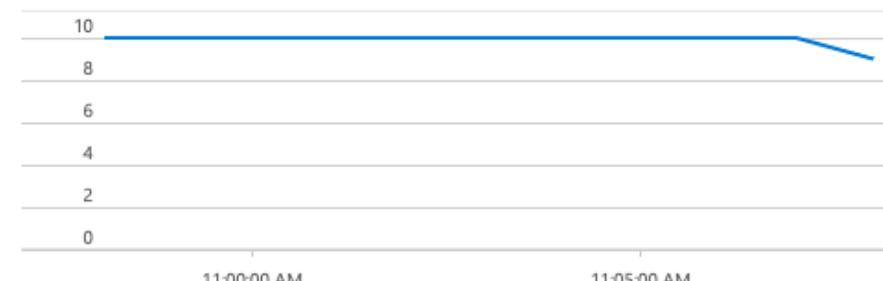
Start time	End time	Engine instances	Test result	Status
4/18/2023, 10:58:15 AM	4/18/2023, 11:08:45 AM	1	Not Applicable	 Done

Load test results 

Client-side metrics

Requests : **HTTP Request** Percentile : **90** Error type : **0** Time range : **4/18/2023, 10:58:15 AM - 4/18/2023, 11:08:18 AM** Group by : **1m**

Virtual Users (Max)



10
8
6
4
2
0

11:00:00 AM 11:05:00 AM

Response time (successful responses)

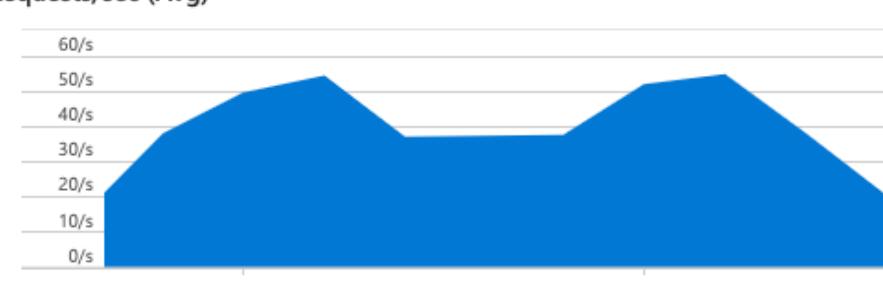


500ms
400ms
300ms
200ms
100ms
0ms

11:00:00 AM 11:05:00 AM

HTTP Request Pct 90 **332.36 ms**

Requests/sec (Avg)



60/s
50/s
40/s
30/s
20/s
10/s
0/s

11:00:00 AM 11:05:00 AM

HTTP Request **39.11 /s**

Errors (total)

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

CharlyAutomatiza @char_automatiza