DZone

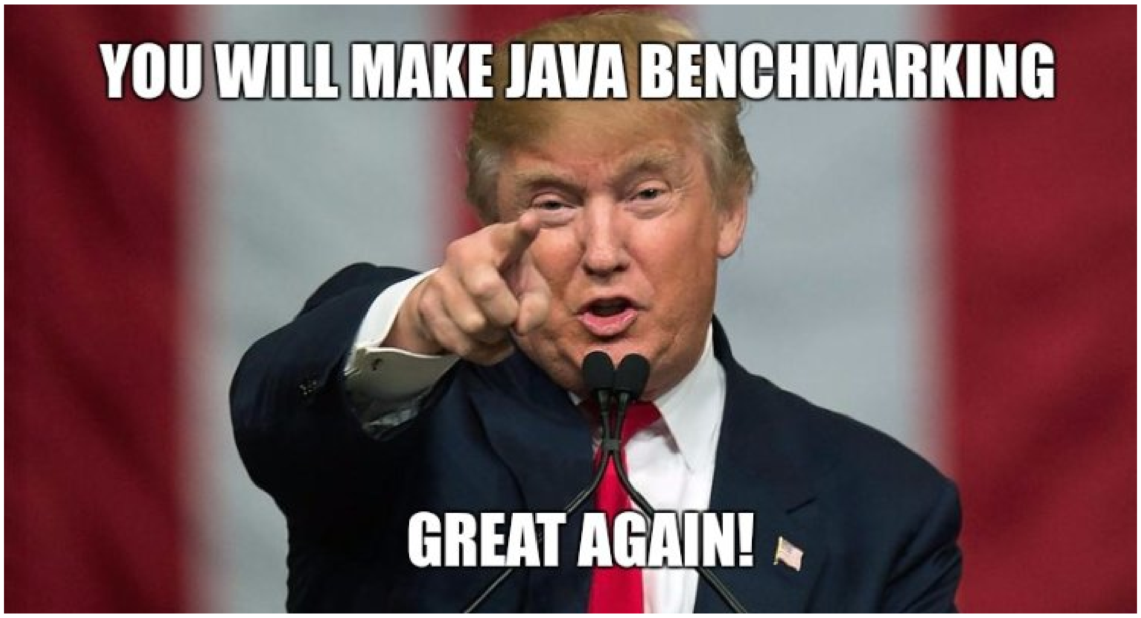
JMH - Great Java Benchmarking

by Dmitry Egorov · Oct. 31, 17 · Performance Zone · Tutorial

If you still measure execution time like this:

```
long before = System.currentTimeMillis();
doMagic();
long now = System.currentTimeMillis();
System.out.println("Seconds elapsed: " + (now-before)/1000F + " seconds." );
```

Then it's time to use JHM framework.



This rich open source framework provides you a proper way to measure the performance of your Java code. With JHM, you can easily

Getting Started

To generate a hello world project, just execute this Maven command:

```
mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.openjdk.jmh -Darchety
```

Writing Your First JHM Hello World Benchmark

In our simple example, we will estimate average time of Thread.sleep (2000). In MyBenchmark.java, we put:

```
package org.sample;
import org.openjdk.jmh.annotations.*;
import java.util.concurrent.TimeUnit;

public class MyBenchmark {
    @Benchmark@BenchmarkMode(Mode.AverageTime) @OutputTimeUnit(TimeUnit.MICROSECONDS)
    public void testMethod() {
        doMagic();
    }
    public static void doMagic() {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException ignored) {
        }
    }
}
```

Now we need to build it executing the famous Maven command:

```
maven clean install
```

Starting the Benchmark

Once the Maven command is being executed in the target folder, you can find executable benchmark.jar. We will execute this jar with the next benchmark parameters:

- number of forks = 1
- warm up iterations = 2

```
# VM options: <none>
# Warmup: 2 iterations, 1 s each
# Measurement: 5 iterations, 1 s each
# Timeout: 10 min per iteration
# Threads: 1 thread, will synchronize iterations
# Benchmark mode: Average time, time/op
# Benchmark: org.sample.MyBenchmark.testMethod

# Run progress: 0,00% complete, ETA 00:00:07
# Fork: 1 of 1
# Warmup Iteration   1: 1999653,736 us/op
# Warmup Iteration   2: 1999914,772 us/op
Iteration   1: 2000066,040 us/op
Iteration   2: 1999286,499 us/op
Iteration   3: 1999159,327 us/op
Iteration   4: 1999529,242 us/op
Iteration   5: 1999628,748 us/op

Result "org.sample.MyBenchmark.testMethod":
  1999533,971 (99.9%) 1352,808 us/op [Average]
  (min, avg, max) = (1999159,327, 1999533,971, 2000066,040), stdev = 351,320
  CI (99.9%): [1998181,163, 2000886,779] (assumes normal distribution)
# Run complete. Total time: 00:00:14

Benchmark                Mode  Cnt        Score      Error  Units
MyBenchmark.testMethod   avgt    5  1999533,971  1352,808  us/op
```

As you can see, JHM measured average time as 1.999533971 seconds.

Alternative JMH Configuration

There are at least two ways to configure your JMH benchmark.

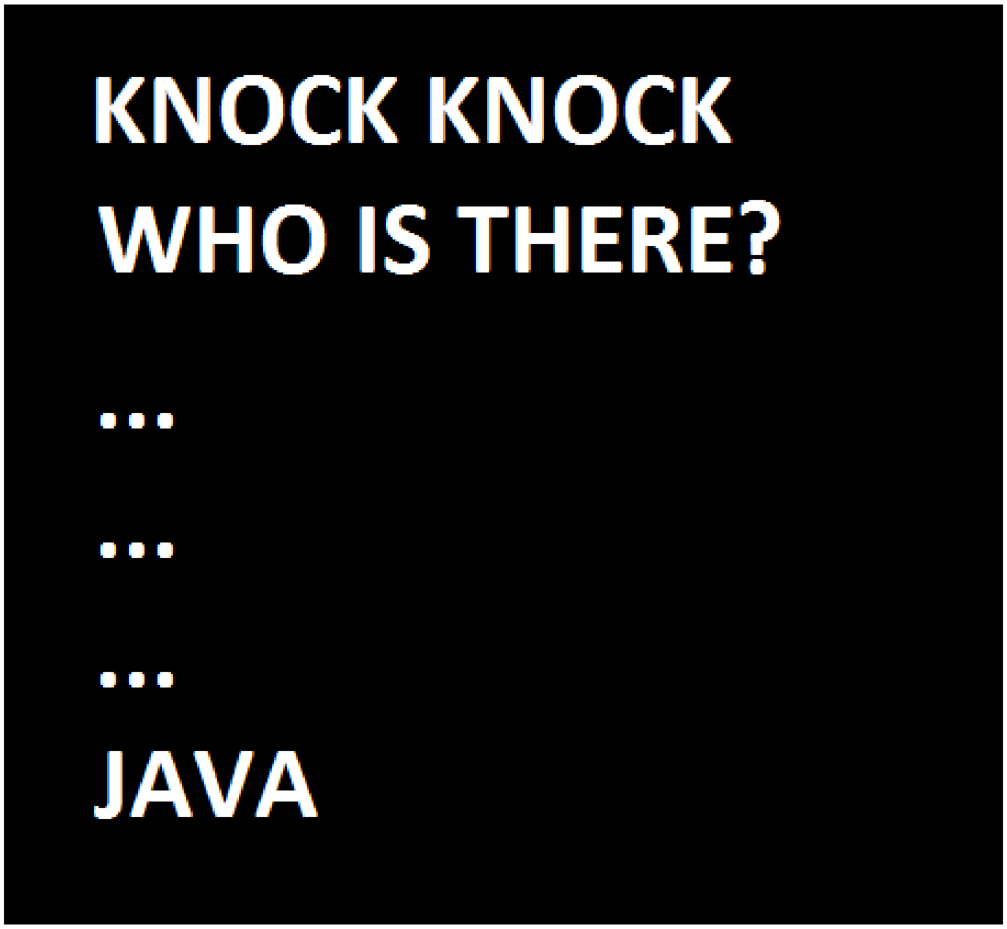
#1 Using annotations:

```
@Benchmark
@BenchmarkMode(Mode.AverageTime) @OutputTimeUnit(TimeUnit.MICROSECONDS)
@Fork(value = 1)
@Warmup(iterations = 2)
@Measurement(iterations = 5)
```

```
                      .forks(1)
                      .shouldDoGC(true)
                      .build();
```

You can download the source code and build files here.

Instead of a conclusion:


KNOCK KNOCK

WHO IS THERE?

...

...

...

JAVA

Take the Chaos Out of Container Monitoring. View the webcast on-demand!

Like This Article? Read More From DZone

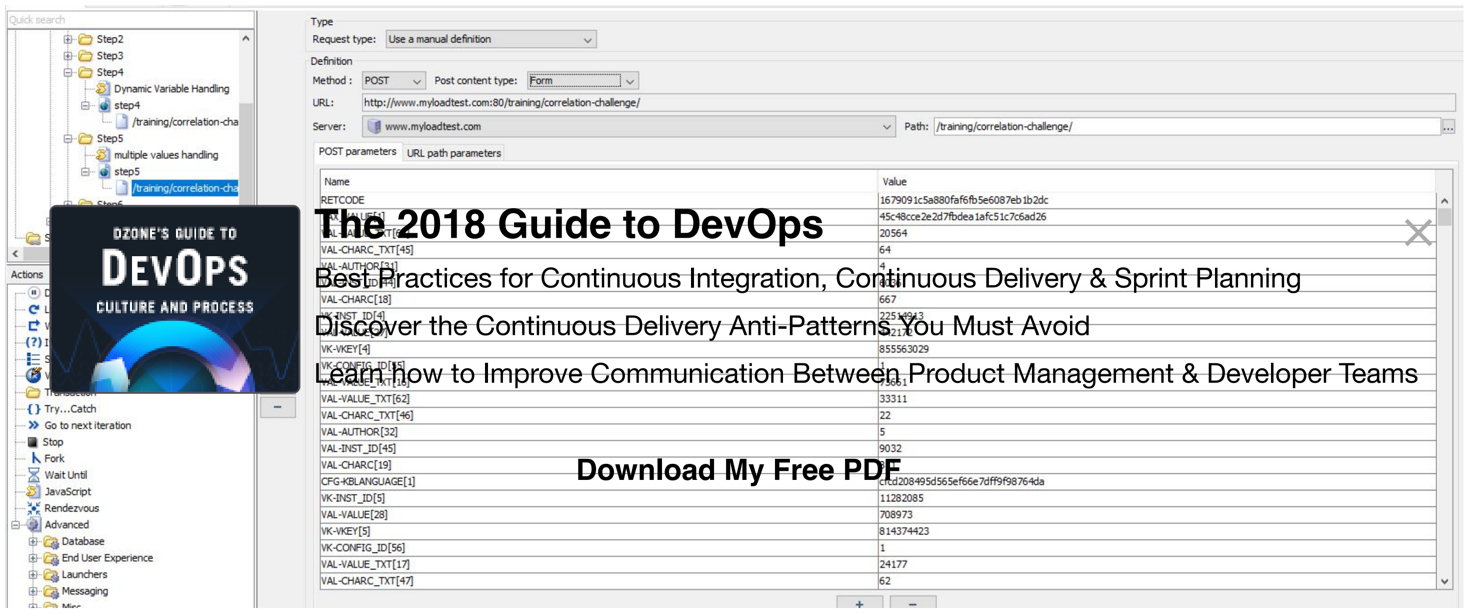
Using JMH to Find the Fastest Way to Encode/Decode UTF-8

Squeezing Another 10% Speed Increase out of jOOQ Using JMC and

Handling Variable Number of Request Parameters in Neoload

by Manojkumar Tenali · Mar 30, 18 · Performance Zone · Tutorial

If your request has a lot of parameters which are being passed to HTTP Request, creating a request to manually correlate the data will be a cumbersome process, when instead, we can use scripting capabilities using JavaScript in Neoload to make a much easier and effective way of handling them.



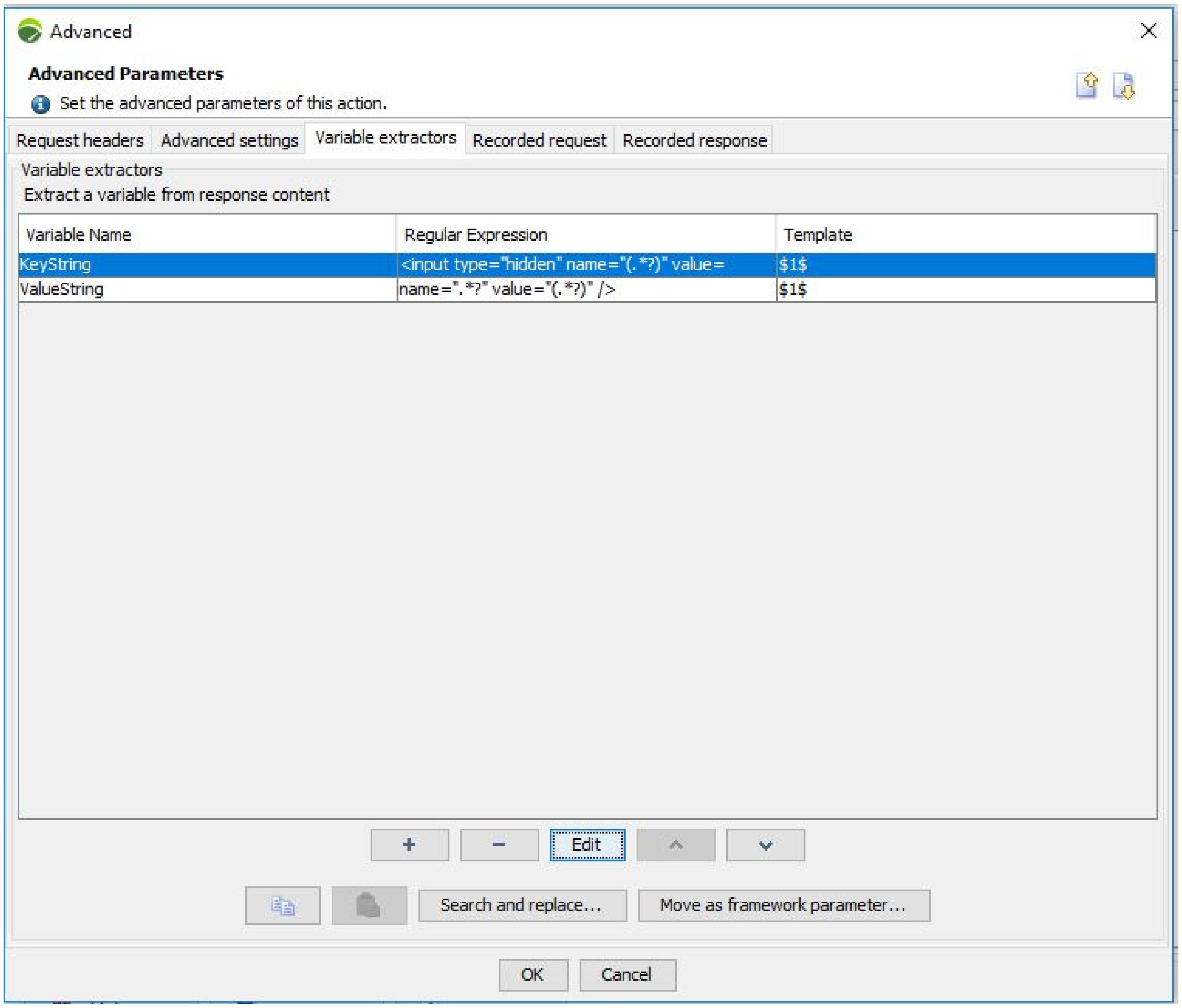
Type
Request type: Use a manual definition
Definition
Method : POST Post content type: Form
URL: http://www.myloadtest.com:80/training/correlation-challenge/
Server: www.myloadtest.com Path: /training/correlation-challenge/

Name	Value
RETCODE	1679091c5a880faf6fb5e6087eb1b2dc
[illegible]	45c48cce2e2d7fbdea1afc51c7c6ad26
[illegible]	20564
VAL-CHARC_TXT[45]	64
VAL-AUTHOR[31]	4
[illegible]	[illegible]
VAL-CHARC[18]	667
VK-INST_ID[4]	22514913
[illegible]	[illegible]
VK-VKEY[4]	855563029
[illegible]	1
[illegible]	[illegible]
VAL-VALUE_TXT[62]	33311
VAL-CHARC_TXT[46]	22
VAL-AUTHOR[32]	5
VAL-INST_ID[45]	9032
VAL-CHARC[19]	[illegible]
CFG-KBLANGUAGE[1]	cfcd208495d565ef66e7dff9f98764da
VK-INST_ID[5]	11282085
VAL-VALUE[28]	708973
VK-VKEY[5]	814374423
VK-CONFIG_ID[56]	1
VAL-VALUE_TXT[17]	24177
VAL-CHARC_TXT[47]	62

3. Change the **Post Content Type to Text** in HTTP Request.
4. Pass the variable into the request.

Practical Explanation

Capture the variables using variable extractors from the preceding request as below.

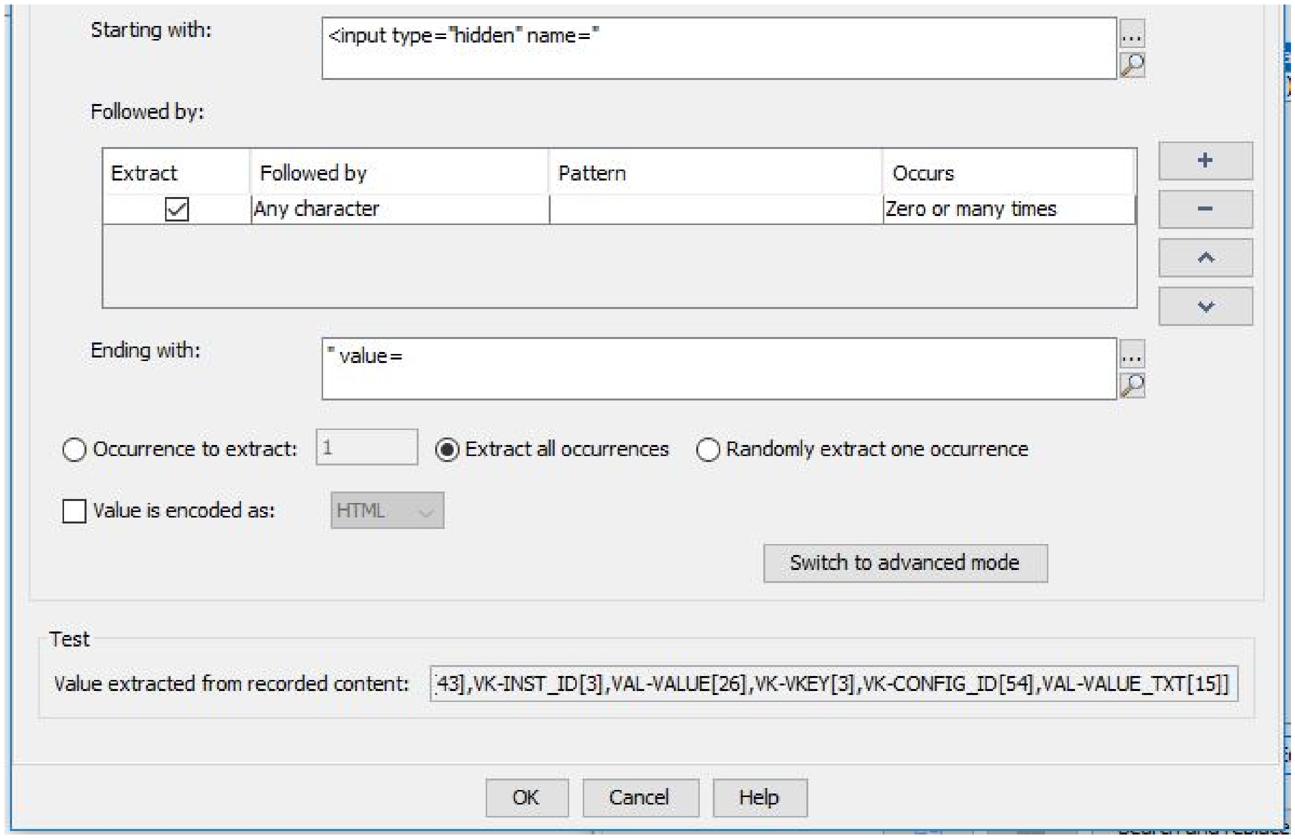


Variable extractors
Extract a variable from response content

Variable Name	Regular Expression	Template
KeyString	<input type="hidden" name="(.*?)" value=	1
ValueString	name=".*?" value="(.*?)" />	1

Click on edit and create the Regular expression as below and select *Extract All Occurrences*.

Variable Extractor
Extract a value from the response and assign it to a variable. Press F1 for Help



Starting with: <input type="hidden" name="

Followed by:

Extract	Followed by	Pattern	Occurs
✓	Any character		Zero or many times

Ending with: " value=

Occurrence to extract: 1 [✓ Extract all occurrences] Randomly extract one occurrence

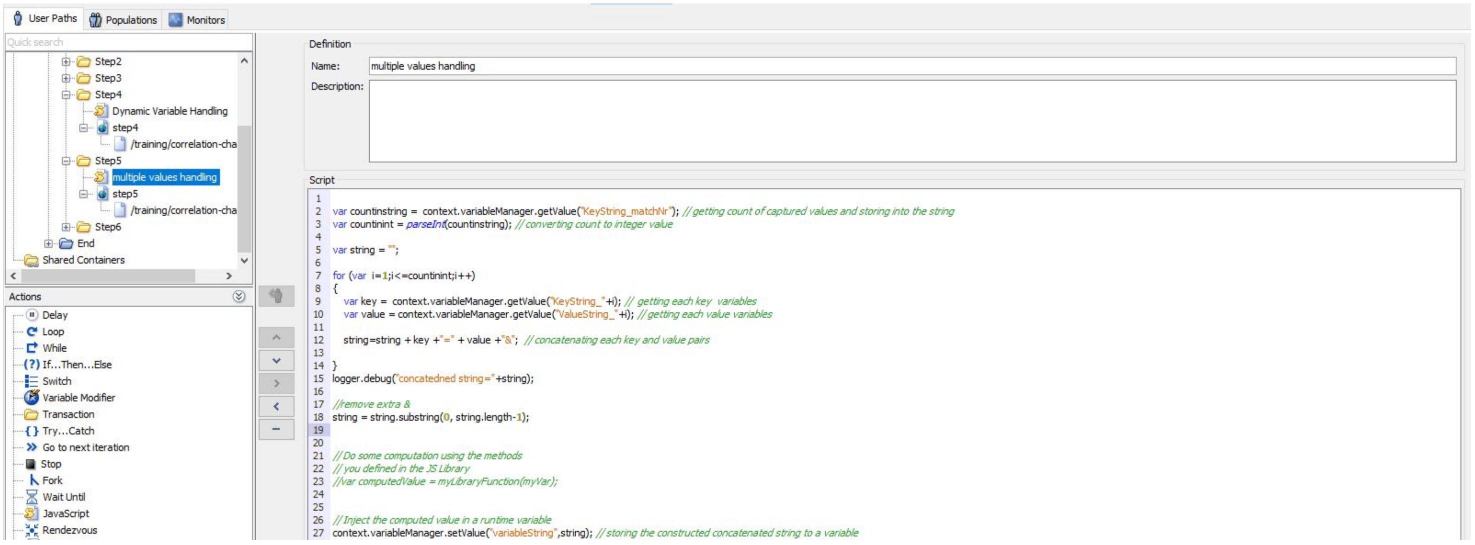
Value is encoded as: HTML

Test

Value extracted from recorded content: 43],VK-INST_ID[3],VAL-VALUE[26],VK-VKEY[3],VK-CONFIG_ID[54],VAL-VALUE_TXT[15]]

You would be able to see extracted value under test header as above.

Now drag and drop Javascript action on to userpath as below and enter the following JavaScript Code:



Definition

Name: multiple values handling

Description:

Script

```

var countinstring =  context.variableManager.getValue("KeyString_matchNr"); //getting count of captured values and storing into the string
var countinint = parseInt(countinstring); // converting count to integer value

var string = "";

for (var  i=1;i<=countinint;i++)
{
   var key =  context.variableManager.getValue("KeyString_"+i); // getting each key  variables
   var value = context.variableManager.getValue("ValueString_"+i); //getting each value variables

   string=string + key +"=" + value +"&";  // concatenating each key and value pairs

}
logger.debug("concatedned string="+string);

//remove extra &
string = string.substring(0, string.length-1);


//Do some computation using the methods
// you defined in the JS Library
//var computedValue = myLibraryFunction(myVar);


// Inject the computed value in a runtime variable
context.variableManager.setValue("variableString",string); // storing the constructed concatenated string to a variable
```

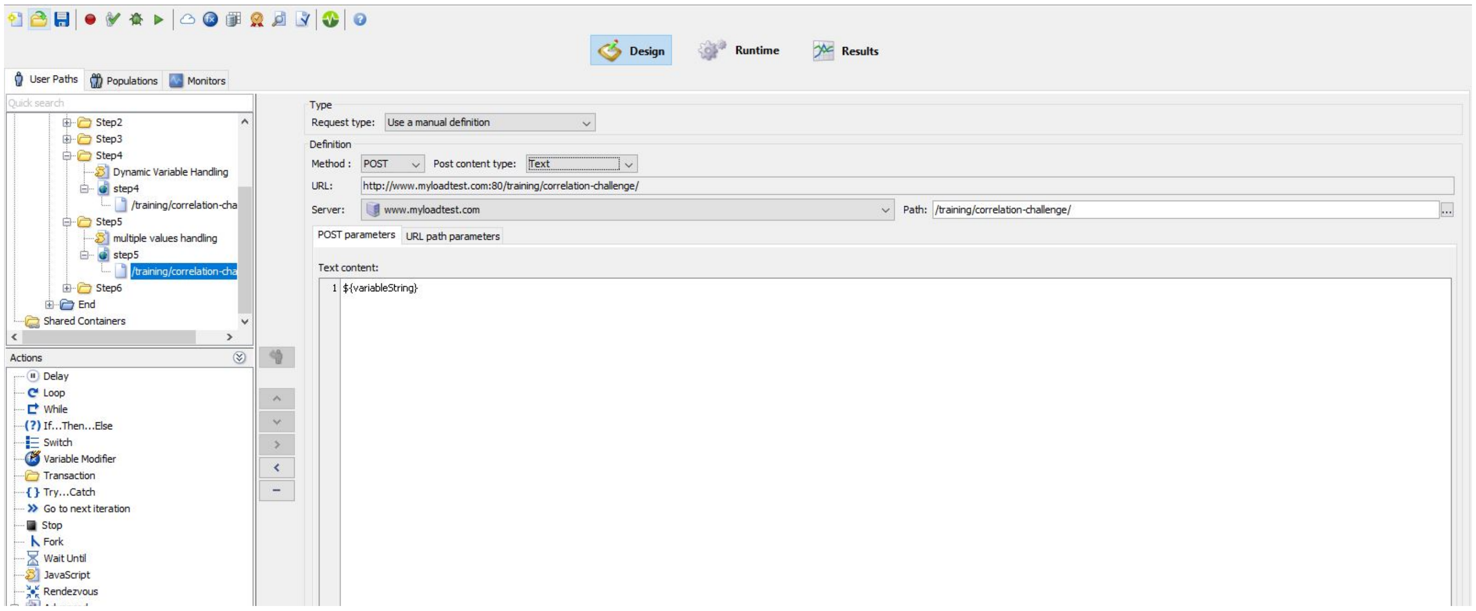
```
var string = "";

for (var i = 1; i <= countinint; i++) {
    var key = context.variableManager.getValue("KeyString_" + i); // getting each key varia
    var value = context.variableManager.getValue("ValueString_" + i); // getting each value

    string = string + key + "=" + value + "&"; // concatenating each key and value pairs

}
logger.debug("concatenated string=" + string);
string = string.substring(0, string.length - 1); //remove extra &

// Inject the computed value in a runtime variable
context.variableManager.setValue("variableString", string); // storing the constructed conc
```

Please refer to JavaScript API of Neoload community

https://easyperformanceautomation.com/category/neoload/

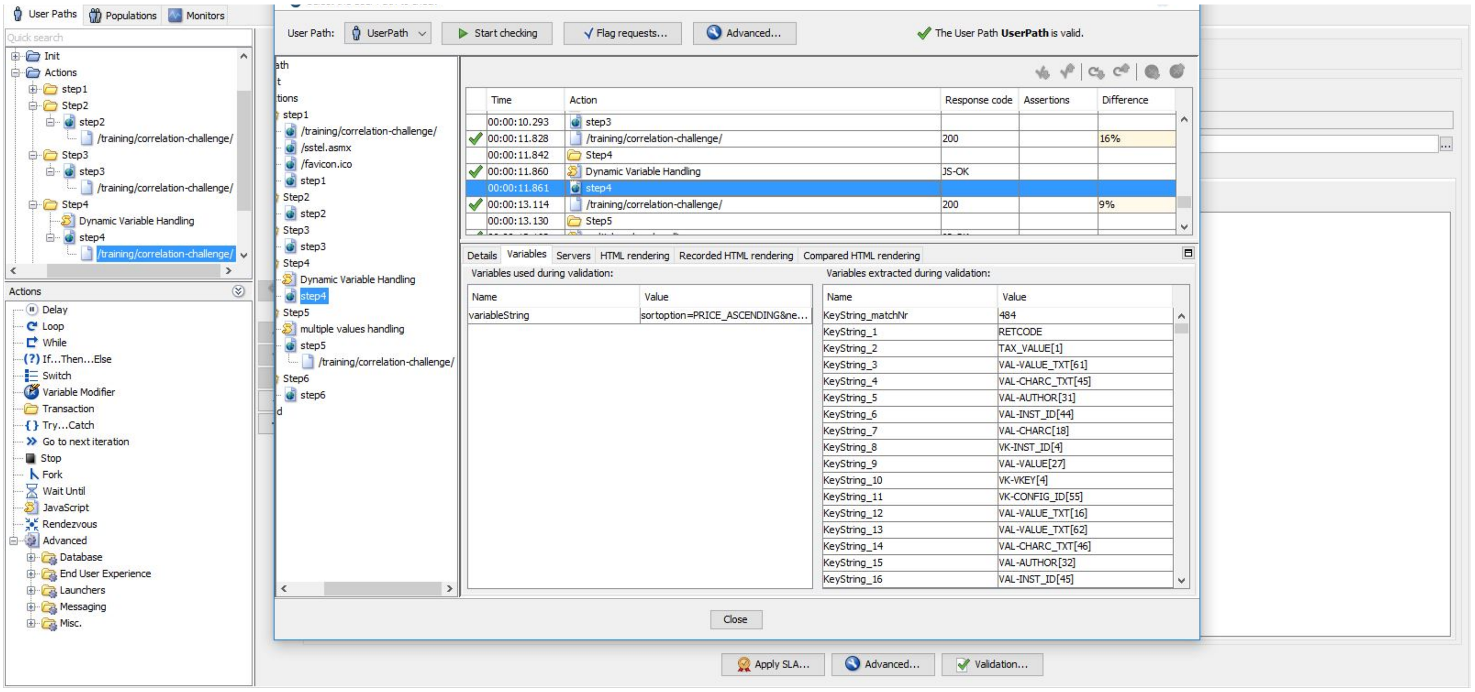
The final step is to change the Post Content type to Text as below and pass the captured value as a parameter.

POST requests with a `text/...` -type content. The contents of these requests may contain NeoLoad variables for generating dynamic content.



Type
Request type: Use a manual definition
Definition
Method : POST   Post content type: Text
URL: http://www.myloadtest.com:80/training/correlation-challenge/
Server: www.myloadtest.com   Path: /training/correlation-challenge/

Text content:
```
${variableString}
```



The User Path **UserPath** is valid.

	Time	Action	Response code	Assertions	Difference
	00:00:10.293	step3			
✓	00:00:11.828	/training/correlation-challenge/	200		16%
	00:00:11.842	Step4			
✓	00:00:11.860	Dynamic Variable Handling	JS-OK		
	00:00:11.861	step4			
✓	00:00:13.114	/training/correlation-challenge/	200		9%
	00:00:13.130	Step5			

Details | Variables | Servers | HTML rendering | Recorded HTML rendering | Compared HTML rendering

Variables used during validation:

Name	Value
variableString	sortoption=PRICE_ASCENDING&ne...

Variables extracted during validation:

Name	Value
KeyString_matchNr	484
KeyString_1	RETCODE
KeyString_2	TAX_VALUE[1]
KeyString_3	VAL-VALUE_TXT[61]
KeyString_4	VAL-CHARC_TXT[45]
KeyString_5	VAL-AUTHOR[31]
KeyString_6	VAL-INST_ID[44]
KeyString_7	VAL-CHARC[18]
KeyString_8	VK-INST_ID[4]
KeyString_9	VAL-VALUE[27]
KeyString_10	VK-VKEY[4]
KeyString_11	VK-CONFIG_ID[55]
KeyString_12	VAL-VALUE_TXT[16]
KeyString_13	VAL-VALUE_TXT[62]
KeyString_14	VAL-CHARC_TXT[46]
KeyString_15	VAL-AUTHOR[32]
KeyString_16	VAL-INST_ID[45]

This way we can handle any number of dynamic parameters.

Like This Article? Read More From DZone

How to Handle a Variable Number of Parameters in Apache JMeter

JMeter Parameterization: The Complete Guide

Optional Parameters Handling Strategy in Java

Free DZone Refcard
Improving Web Performance With Varnish

Topics: NEOLOAD, SCRIPTING, PERFORMANCE, JAVA-TO-JAVASCRIPT, ORGANIZATION, REQUEST PARAMETERS