# Performance and load testing with JMeter

- Overview
  - JMeter concepts
    - Mapping of concepts to Xray
      - Test
      - Test status
        - Other relevant performance test results
- Requirements
- Description
  - JPetStore example
    - Setup: checking out the JMeter project and setup of auxiliary variables
    - Configuring the Build steps
    - Configuring the Post-build actions
  - JPetStore with assertions example
- Room for improvement
- References

# Overview

JMeter is an open-source tool used for performance and load testing.

Normally used to measure web site performance, it can be also used in broader contexts.

Native features provide a reasonable set of samplers and reports; however, this may be extended using plugins.

JMeter does not provide, by default, a SLA/SLO mechanism. Basic SLAs may be implemented using assertions (e.g. duration/response assertion or custom assertion) though.

JMeter has a GUI but it can be run in command line mode using its CLI. It can produce JTL/CSV based reports or XML based reports; the latter provide additional information.

### JMeter concepts

The following table provides an overview of JMeter concepts; if you're used to it, you can probably skip it.

By having these concepts present, we may reflect on their mapping somehow to Xray.

JMeter concept	What it means?
Test Plan	a high-level testing scope, consisting of multiple "users"/threads doing multiple acctions
Thread Group	users
Controller	what drives the actions and flow of tests
Sampler (controller)	request
Logic (controller)	a way to group and determine which samplers to run
Transaction (controlller)	one type of logic controller that provides a way to group multiple samplers and its samples (i.e. requests) in order to obtain an additional sample based on them
Sample	obtained sample (i.e. the "response")
(sampler) Assertion	Assertions are used to perform additional checks on samplers, validating samples accordingly with a criteria, marking it as successful or not.
Listener	test results/samples listener (e.g. for producing reports)

### Mapping of concepts to Xray

JMeter is not a functional testing tool; it's essentialy a load tool simulating multiple users (threads), doing several actions as they would in a typical usage scenario.

Mapping of concepts may no be straighforward thiugh.

If we aim to have visibility of the performance testing results, we need to think in the following questions:

• What can we consider the Test?

- · How can we assess if was successful or not?
- What information is relevant for analysis?

#### Test

The Test could be the whole JMeter's test plan; this is a valid and simple approach. It depends on how you use the test plan.

A Test could also represent each user/thread on that test plan; this would create tons of Tests that would be meaningless as they would not clearly identify anything in particular and could not be reused whatsoever.

Another approach would be to use each sampler as a Test. However, samplers are normally grouped and nested under other controllers. Thus, a better approach would be to represent all controllers (samplers and logic controllers) as Tests.

### **Test status**

Determining whether a test was successful or not, first depends on what you define as being the "Test".

In this tutorial we'll consider each controller as a Test in Xray. Classifying it as failed or not can be done based on the nested assertion results or simply on the implicit sampler' (un)successful classification.

#### Other relevant performance test results

As part of performance testing, the following metrics are common:

- errors (count, %)
- total elapsed time (e.g average, min, max, std dev, 90th percentile)
- latency time/TTFB (e.g average, min, max
- connect time (e.g average, min, max)
- requests throughput/requests per time unit (e.g. average)
- received bytes (total, throughput)
- sent bytes (total, throughput)
- requests (count)

Some of these may be considered as KPIs and used to define SLA/SLOs. JMeter does not provide a way to implement SLAs though.

SLAs are usually marked as being successful/met, warning or as failed/unmet.

## Requirements

- JMeter
- JMeter Plugins Manager and some plugins (jmeter-http, jpgc-httpraw, jpgc-graphs-basic, jpgc-graphs-additional, jpgc-synthesis, jpgc-cmd)
- Jenkins (optional)

# Description

The overall approach to have visibility of the performance results in Xray will be as follows:

- 1. run JMeter in command line
- 2. generate results in JTL (CSV based) format
- 3. post-process results to
  - a. generate a JUnit XML report, mapping each controller as a Test
  - b. generate dashboard report, containing multiple reports/charts
  - c. produce aggregate report or similar (e.g. synthesis report) in CSV
  - d. produce one or more charts
- submit results to Xray along with the previously generated report assets
  - a. fill out the "Description" field of the corresponding created Test Execution issue with
    - i. link to project/job in Jenkins
    - ii. link to dashboard HTML report in Jenkins workspace
    - iii. aggregate report content formatted as a table

### JPetStore example

In this example, we're load testing a fictitious pet store site name JPetStore (this site is kindly provided by Octoperf for demo purposes).



The testing scenario exercises 20 users, with a ramp-up period of 240s, doing a standard user path/scenario: go to the site, login, view a category, then a product, add to cart, buy it and logout.

📔 🦉 🚔 🗒 👗 🛄 🗎 ( + )	- 🍫 🕨 👦 🗶 💐 🎬 🍂 🏷 🔚 🔋 🛛 00:04:00 🛕 0 0/20 🛞 🚣
🔻 👗 JMeter Demo	Thread Group
🔻 🔯 JPetstore	Thread Group
DNS Cache Manager	
r 🛛 💥 HTTP Cookie Manager	
HTTP Cache Manager	Comments:
HTTP Authorization Manager	Action to be taken after a Sampler error
🔻 🥫 Home page	
🔹 🕨 🎤 homepage	💿 Continue 🔷 Start Next Thread Loop 🔷 Stop Thread 🔷 Stop Test 🦳 Stop Test Now
🔻 🥫 Login page	
🕨 🧪 signinForm	Thread Properties
👻 👻 Signin	
🕨 🧪 viewCatalog	Number of Threads (users): 20
signinAccount	
🔻 🧝 ViewCategory	Ramp-up period (seconds): 120
viewCategory	
🔻 🥫 ViewProduct	
viewProduct	✓ Same user on each iteration
▼ 🔄 AddToCart	
AddItemToCart	
newOrderForm	
setBillingInfo	Specify Thread lifetime
confirmOrder	Duration (seconds): 240
► 🕒 LogOff	
🎿 View Results Tree	
🎿 jp@gc – Synthesis Report (filtered)	
🋃 Summary Report	

There are several transactions, grouping one or more HTTP requests (i.e. using the HTTP Request sampler).

However, there are no explicit assertions; thus, all failures (i.e. samples marked as being unsuccessful) will be based on the standard HTTP response codes.

Tests can be run using JMeter GUI or using the command line jmeter, which is the preferred approach if you wish to make it part of your CI.

We'll use Jenkins as our CI tool and we'll configure a freestyle project for running our tests.

### Setup: checking out the JMeter project and setup of auxiliary variables

We need to setup some variables related to the Jira instance to be able to attach some files to the Test Execution issue later on, if we want to, using the at tach\_files\_to\_issue.sh shell script.

These are somehow redundant with the Xray instance configuration but are necessary if we wish to expose them.

We start by defining one variable for the Jira server base URL as build pararamter.

This project is parameterized			0
	String Parame	ter X	0
	Name	JIRA_BASEURL	0
	Default Value	http://192.168.56.102	0
	Description		0
		Plain text] Preview	
			0

Using the Credentials Binding plugin, we will populate two variables for the Jira instance's username and password; these will be, in turn, obtained from the credentials already stored and linked to the Xray instance configuration in Jenkins.

Bindings					
Username and pass	word (separated)	×			
Username Variable	me Variable JIRA_USERNAME				
Password Variable	JIRA_PASSWORD				
Credentials	Specific credentials Parameter expression	0			
	admin/****** (Jira admin user)				
Add 👻					
Abort the build if it's stud	k				
Add timestamps to the C	Console Output				
Inject environment varia	oles to the build process	0			
Inject passwords to the l	build as environment variables				
With Ant					

The "code" will be checked out from our source code versioning system (e.g. Git), which contain the JMeter project(s) saved in .jmx format along with some additional scripts.

Jenkins > jmeter-petstore-octoperf >				
	General Source Code Management	Build Triggers	Build Environment Build Post-build Actions	
	Source Code Management	t		
	None			
	Git			
	Repositories			0
		Repository URL	ssh://git@localhost/home/git/repos/jmeter-exp.git	0
		Credentials	git/***** 🗘 🖝 Add	
			Advanced	
			Add Repository	

### Configuring the Build steps

The "build" is composed of several steps, starting with the one that runs JMeter.

## Build

Execute shell								
Command	./run_petstore_octoperf.sh							
	See the list of available environment variables							

//un\_petstore\_octoperf.sh
#!/bin/bash
JMETERPLUGINSCMD=JMeterPluginsCMD.sh
./cleanup.sh
# run jmeter and produce a JTL csv report
jmeter -n -t examples/jpetstore/jpetstore\_configurable\_host.jmx -l results.jtl -e -o dashboard
# process JTL and covert it to a synthesis report as CSV
\$JMETERPLUGINSCMD --generate-csv synthesis\_results.csv --input-jtl results.jtl --plugin-type SynthesisReport
\$JMETERPLUGINSCMD --generate-png reports/ResponseTimesOverTime.png --input-jtl results.jtl --plugin-type
ResponseTimesOverTime --width 800 --height 600
\$JMETERPLUGINSCMD --generate-png reports/TransactionsPerSecond.png --input-jtl results.jtl --plugin-type
TransactionsPerSecond --width 800 --height 600
./convert.sh "jmeter.jpetstore"

We need to process the JTL file and produce a report that can be submitted to Xray; we'll use a JUnit XML based report that will be generated using a specific tool.

About JMeter to JUnit XML converts

(i)

There are several JMeter JTL to JUnit XML converters out there. However, most of them do neither a implement a mapping of concepts that is useful nor provide additional information about the failures.

This tutorial uses a modified version (pre-built JAR) of the the jmeter-junit-xml-converter code.

It will produce a JUnit XML report containing:

- one Test Suite per each Thread
- multiple <testcase> elements, one per each controller
- add information about the duration (i.e "time" attribute) on each <testcase>
  - add failure message, if available

The modified jmeter-junit-xml-converter utility will produce a junit.xml and an alternate\_junit.xml file; we want the latter as it better suits our needs. We'll call it using the converter.sh shell script along with a parameter that will allow us to uniquely identify the Tests afterwards (e.g. "jmeter. jpetstore").

./convert.sh
#!/bin/bash
if [ \$# == 1 ];
then
TESTSUITE=\$1
else
TESTSUITE="jmeter"
fi
JAR=./converters/jmeter-junit-xml-converter-0.0.1-SNAPSHOT-jar-with-dependencies.jar
java -jar \$JAR results.jtl junit.xml \$TESTSUITE

Optionally, we'll add two build steps to store the tabular aggregate report in an environment variable (e.g. AGGREGATE\_TABLE) as a string. This requires the Environment Injector plugin.

Execute shell								
Command	#!/bin	/bash						
	<pre>echo AGGREGATE_TABLE="\$(./process_aggregate.sh)" &gt; envvars.properties</pre>							
:	See <u>the I</u>	ist of available environment variables						
			Advanced					
			X					
Inject enviro	onment v	rariables		Ð				
Properties F	ile Path	envvars.properties		0				
Properties C	Content			0				
./process_a	iggreg	ate.sh						
#!/bin/ba	.sh							
cat repor	ts/ag	ggregate_results.csv  tr "," " "   sed -e 's/^/ /'   sed -e 's/\$/ \\\\n\\/'   sed -e '1	s/ /	/g'				

### **Configuring the Post-build actions**

()	Bonus tip! The Jenkins' Perform unstable depending	n <mark>ance plugin</mark> can <u>optio</u> on absolute or relative	<u>onally</u> be u threshold	ised ds.	to create s	some tr	rend cł	narts in .	Jenkins	and als	o as me	eans to	mark the	build as	failed o	r
	Publish Performance test result report	rt														
	Source data files (autodetects format):	results.jtl														
	Regex for included samplers															
	Select evaluation mode	Expert Mode  Standard Mode														
		Standard Mode														
		Select mode:	Relative Threshol	ld 😑 Error	Threshold											
		Use Error thresholds on single build:	Unstable -1													
			Failed	-1												
						A	dvanced									
		Use Relative thresholds for build comparison:	Unstable % Range Failed % Range O Compare with p Compare based on	previous Bu	(-) -1,0 -1,0 uild  Compare with Build Average Respo	(+) -1,0 -1,0 Id number ( nse Time \$	0									
		Expert Mode														
		Constraint settings  Beginner Failed Beginner Unstat Beginner	Builds ble Builds int log to workspace													
		Constraints Add a new const	traint 👻													

Test results can be submitted to Xray either by using a command line tool (e.g. curl) or by using a specific CI plugin which in our case will be the "Xray – Test Management for Jira Plugin".

We could choose the "JUnit XML" as the format in the "Xray: Results Import Task", that would be simpler to setup.

However, if we use the "JUnit XML multipart" format, we can further customize the Test Execution issue. We'll use this as means to provide a link to the Jenkins build along with a link to dashboard report generated by JMeter. We may also provide the aggregate report table stored previously as an environment variable.

Xray: Results	Import Task		X
Jira Instance	xray-vm		
Format	JUnit XML multipart		
Parameters	Import to Same Test Execution	٥	
		When this option is check, if you are importing multiple execution report files using a glob expression, the results will be imported to the same Test Execution	
	Execution Report File (file path with file name)	alternate_junit.xml	
	Test Execution fields	JSON Content	\$
		{     "fields": {         "project": {             "fields": {             "key": "CALC"             },             "summary": "JMeter performance results",             "description": "Build URL: \$(BUILD_URL).\n\nDetailed dashboard report at:             \$(JOE_URL)ws/dashboard/index.htmln\n^Aggregate results summary"\n\n \$(AGGREGATE_TABLE)\n",             "issuetype": {             "name": "Test Execution"             }             }	

If using this format, you'll need to provide the Test Execution's issue type name (or the id) and the project key.

```
Test Execution fields (JSON content) - example1
{
    "fields": {
        "project": {
            "key": "CALC"
        },
        "summary": "JMeter performance results",
        "description": "Build URL: ${BUILD_URL}.\n\nDetailed dashboard report at: ${JOB_URL}ws/dashboard/index.
html\n\n*Aggregate results summary*\n\n ${AGGREGATE_TABLE}\n",
        "issuetype": {
            "name": "Test Execution"
        }
    }
}
```

You may also specify the Test Plan, Revision and Test Environments fields but you'll need to obtain their custom field ID from Jira's administration. Note that these IDs are specific to each Jira instance. In the following example, "customfield\_10033" corresponds to the Revision CF, "customfield\_11805" to the Test Environments CF and "customfield\_11807" to the Test Plan CF.

```
Test Execution fields (JSON content) - example2
{
   "fields": {
      "project": {
         "key": "CALC"
      },
      "summary": "JMeter performance results",
      "description": "Build URL: ${BUILD_URL}.\n\nDetailed dashboard report at: ${JOB_URL}ws/dashboard/index.
html\n\n*Aggregate results summary*\n\n ${AGGREGATE_TABLE}\n",
      "issuetype": {
         "name": "Test Execution"
      },
      "customfield_10033": "123",
          "customfield_11805" : [
            "staging"
      ],
      "customfield_11807": [
         "CALC-1200"
      ]
   }
}
```

	ld task	
Tasks		
		Log text
		Add
	Script	ISSUEKEY=\$(echo \$XRAY_TEST_EXECS   cut -d ";" -f 1)
		./attach_files_to_issue.sh \$ISSUEKEY reports/*.png dashboard.zip
	Run script only if all previous steps were successful	
	Escalate script execution status to job status	
	Add another task	
For the tin for that.	Add another task	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue
For the tin for that.	Add another task ne being, the Jenkins plugin can't uploa iles_to_issue.sh /bash	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue
For the tin for that. attach_f #!/bin BASEUR:	Add another task The being, the Jenkins plugin can't uploa Tiles_to_issue.sh /bash L=\${JIRA_BASEURL:-http://you:	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue
For the tim for that. attach_f #!/bin BASEUR: USERNAI	Add another task The being, the Jenkins plugin can't uploa Tiles_to_issue.sh The bash L=\${JIRA_BASEURL:-http://you: WE=\${JIRA_USERNAME:-admin}	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue
For the tim for that. attach_f #!/bin BASEUR: USERNAI PASSWOI	Add another task the being, the Jenkins plugin can't uploa iles_to_issue.sh /bash L=\${JIRA_BASEURL:-http://you: ME=\${JIRA_USERNAME:-admin} RD=\${JIRA_PASSWORD:-admin}	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue
For the tin for that. attach_f #!/bin BASEUR USERNAI PASSWOI ISSUEKI	Add another task The being, the Jenkins plugin can't uplow illes_to_issue.sh /bash L=\${JIRA_BASEURL:-http://you: ME=\${JIRA_USERNAME:-admin} RD=\${JIRA_PASSWORD:-admin} EY=\$1	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue rjiraserver.example.com}
For the tim for that. attach_f #!/bin BASEUR: USERNAI PASSWOI ISSUEKI	Add another task he being, the Jenkins plugin can't uplow illes_to_issue.sh /bash L=\${JIRA_BASEURL:-http://you: WE=\${JIRA_USERNAME:-admin} RD=\${JIRA_PASSWORD:-admin} EY=\$1	ad other files; however, we can make a basic shell script (e.g. attach_files_to_issue

After running Jenkins job, we may track some performance trend charts directly in the project's page. This requires previous configuration of the Performance Plugin as mentioned earlier.

🛞 Je	enkins		3 Qsearch Ø admin   log out
Jenkins	imeter-petstore-octoperf		ENRLE AUTO REFRESH
🛧 Back	to Dashboard	Project imater-patetore-extensi	
🔍 Statu	us	Project jineter-persione-octopen	
🔁 Char	nges		add description
Work	space		Disable Project
🔊 Build	Now	-	Performance Trend
S Delet	te Project	Workspace	Response time
🎂 Confi	igure	Last Successful Artifacts	120
Nerfo	ormance Trend	dashBoard_results.xml 267 B ≤ view     standardBesults.xml 377 KB ≤ view	100
🔁 Rena	ame		ů
		Hecent Changes	40 20
Ø Bu	uild History trend -	Permalinks	<del>د</del> ۲ ۲ ۲ ۲
find		X Lost hulld (40) 2 ha 8 min and	90% line — average — median Percentage of errors
#3	May 26, 2020 6:21 AM	Last stable build (#3), 3 hr 8 min ago	100
. #2	May 26, 2020 6:01 AM	Last successful build (#3), 3 hr 8 min ago	80
#1	May 25, 2020 5:51 PM	<u>Last completed build (iiis), 3 nr 6 min ago</u>	
			20
	S RSS for all RSS for failur	8	
			errors and a second s

As we submitted the processed test results to Xray (alternate\_junit.xml), we can now track them in Jira.

A Test Execution will be created containing a summary of results along with some useful links to access additional information in Jenkins.

Using the link provided in the description field of the Test Execution, we can access an extensive dashboard report generated by JMeter and stored in Jenkins project's workspace.

In order to correctly view it, you may need to change one settings in Jenkins: go to Manage Jenkins > Script console and execute:

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")

Finally, we should be able to correctly display the HTML based dashboard report.



# Calculator / CALC-6540 JMeter performance results

 Image: Edit
 Q Comment
 Synchronize Tests from...
 More \*
 Close Issue
 Reopen Issue
 Admin \*

#### Details

Type: D Test Priority: ⊗ Majo Affects Version/s: None Test Execution 🔶 Major Component/s: None Labels: None Test Environments: None None

#### Test Plan:

#### Description

Build URL: http://192.168.56.102:8081/job/jmeter-petstore-octoperf/3/.

Detailed dashboard report at: http://192.168.56.102:8081/job/jmeter-petstore-octoperf/ws/dashboard/index.html

### Aggregate results summary

Label	1. Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/sec	Std. Dev.
homepage	145	129	122	131	142	231	111	710	0.00%	.6	.2	51.12
Home page	145	129	122	131	142	231	111	710	0.00%	.6	.2	51.12
signinForm	144	64	57	66	76	367	52	369	0.00%	.6	.2	38.48
Login page	145	63	57	65	76	367	0	369	0.00%	.6	.2	38.71
viewCatalog	269	66	55	61	66	105	50	2651	0.00%	1.1	.4	158.26
signinAccount	144	63	55	62	66	312	51	339	0.00%	.6	.2	37.54
Signin	144	138	111	122	190	397	103	2704	0.00%	.6	.5	218.82
viewCategory	141	70	55	63	69	321	50	1634	0.00%	.6	.3	134.50
ViewCategory	144	69	55	63	69	321	0	1634	0.00%	.6	.2	133.48
viewProduct	137	77	55	61	66	566	51	2221	0.00%	.6	.3	189.30
ViewProduct	141	74	55	61	66	566	0	2221	0.00%	.6	.2	187.03
addItemToCart	135	74	55	62	70	400	50	1598	0.00%	.6	.3	138.03
newOrderForm	133	67	54	60	63	408	50	1285	0.00%	.6	.2	110.36
setBillingInfo	130	84	55	63	71	547	51	2550	0.00%	.6	.2	223.43
confirmOrder	128	72	55	61	63	398	50	1681	100.00%	.6	.2	146.61
AddToCart	130	297	223	252	532	1949	164	2722	98.46%	.6	1.0	331.96
signOff	125	69	55	62	66	297	51	1295	0.00%	.6	.2	112.87
LogOff	125	125	111	122	129	402	102	1348	0.00%	.6	.5	113.60
TOTAL	2605	94	57	127	220	408	0	2722	9.83%	10.9	5.4	161.88

RESOLVED (View Workflow)

Fixed None

Status: Resolution: Fix Version/s:

#### **Overall Execution Status**

16 PASS 2 FAIL

Total Tests: 18

〒 Filter(s)

E! ~								Show 100 \$ entries	Columns -
	A Rank	Key	Summary	🖕 Test Type	#Req	#Def	Assignee	Status	
	1	CALC-6533	signinAccount	Generic	0	0	Administrator	PASS	• …
	2	CALC-6522	signOff	Generic	0	0	Administrator	PASS	• …
	3	CALC-6532	newOrderForm	Generic	0	0	Administrator	PASS	• …
	4	CALC-6531	signinForm	Generic	0	0	Administrator	PASS	• …
	5	CALC-6530	viewProduct	Generic	0	0	Administrator	PASS	• …
	6	CALC-6529	confirmOrder	Generic	0	0	Administrator	FAIL	• …
	7	CALC-6539	viewCatalog	Generic	0	0	Administrator	PASS	• …
	8	CALC-6528	LogOff	Generic	0	0	Administrator	PASS	• …
	9	CALC-6527	ViewProduct	Generic	0	0	Administrator	PASS	• …
	10	CALC-6538	AddToCart	Generic	0	0	Administrator	FAIL	• …
	11	CALC-6537	homepage	Generic	0	0	Administrator	PASS	• …
	12	CALC-6526	Login page	Generic	0	0	Administrator	PASS	• …
	13	CALC-6525	Home page	Generic	0	0	Administrator	PASS	• …
	14	CALC-6536	ViewCategory	Generic	0	0	Administrator	PASS	• …
	15	CALC-6535	addItemToCart	Generic	0	0	Administrator	PASS	• …
	16	CALC-6524	setBillingInfo	Generic	0	0	Administrator	PASS	• …
	17	CALC-6523	Signin	Generic	0	0	Administrator	PASS	• ••
	18	CALC-6534	viewCategory	Generic	0	0	Administrator	PASS	• ••

Unstructured (i.e. "generic) Test issues will be auto-provisioned (unless they already exist), one per each controller. The "Generic Definition" field acts as the unique test identifier for subsequent imports and is composed by a prefix along with the controller's name (e.g. "jmeter.jpetstore. AddToCart").

The attachments section on the Test Execution issue provide direct access to some reports and also to a zipped file containing the dashboard report generated by JMeter.

<ul> <li>Attachments</li> </ul>			
		$\stackrel{\frown}{\frown}$ Drop files to attach, or browse.	
dashboard.zip Yesterday 1.05 MB	ResponseTimesOverTime. Yesterday 87 kB	TransactionsPerSecond.pr Yesterday 103 kB	

The execution details of a specific Test Run show multiple entries, each one representing a sample.

The following screenshot showcases the details of the sample produced by the Transaction Controller named "AddToCart". We can see that it was executed multiple times, in the context of different "users" (i.e. JMeter's threads).

Calculator / Test Execution: CALC-6540 / Test: CALC-6538 AddToCart		Export Test as Text	Return to Test Execution	
Execution Status FAIL + Started On: 26/May/20 6:25 AM			Assignee: Administra Executed By: Administrat Tests - environments:	tor Versions: - or Revision: -
Comment Preview Comment 🗸	Execution Defects (0) Create Defect   Create Sub-Task   Add Defects	Execution Evidence (0)		Add Evidence 🗸
Execution Details  Test Description None  Test Details  Test Type: Generic Definition: jmeter.jpestore.AddToCart  Results				^
Context	Qutout		Duration	Status
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	es : 1	219.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	es : 1	360.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	es : 1	215.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	s : 1	223.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	es : 1	219.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	es : 1	211.000 ms	FAIL
TestSuite JPetstore 1-6	Number of samples in transaction : 4, number of failing sample	is : 1	221.000 ms	FAIL

### JPetStore with assertions example

This example (JMeter project file) is similar to the previous one with the exception that it contains some assertions: one standard Size assertion and a custom BeanShell assertion that looks at the duration and marks the sample as unsuccessful after "maxErrors" failures .

We'll use a set of variables defined at JMeter's test plan-level to assist in the assertion logic.

jpetstore_octoperfsite_with_ass	sertions.jmx (/U	sers/smsf/jmete	r/examples/	jpetstore/jj	petstore_oc	toperfsite_wit	h_assertions.jr	mx) - Apache JM	leter (5.3)	)		
📰 🏟 🚔 🗮 📈 📭 💼 🕂 +	- 4	🕨 💊 💿	) 🔕 🔤	¥ 🐝	676 🏷	1		00:00:00	<u>^</u> 0	0/0	€	٤.
🔻 👗 JMeter Demo	User Defir	ed Variable	s									
🛛 🔀 User Defined Variables	oser bein											
🔻 🔯 JPetstore	Name:	Name: User Defined Variables										
DNS Cache Manager												
HTTP Cookie Manager	Comments:											
HTTP Cache Manager					User [	Defined Varial	oles					
🔀 HTTP Authorization Manager												
🔻 📒 Home page		Name:				Value			Descripti	on		
▶ 🧨 homepage	SLA_elapsedTime_threshold			10								
ReanShell Assertion	SLA_elapsed	A_elapsedTime_failures 0										
Size Assertion	SLA_elapsed	Time_maxErrors										
🕨 🥃 Login page												

\_\_\_\_\_

jpetstore_octoperfsite_with_as	sertions.jmx (/Users/smsf/jmeter/examples/jpetstore/jpetstore_octoperfsite_with_assertions.jmx) - Apache JMeter (5.3)
🖀 🗯 🚔 📰 📈 🗊 🙆 +	- 🍫 🕨 🔈 🌚 💩 👹 🎬 🦛 🍾 🚍 👔 🛛 00:00:00 🛕 0 0/0 🤂 💰
<ul> <li>Meter Demo</li> <li>User Defined Variables</li> <li>DNS Cache Manager</li> <li>HTTP Cookie Manager</li> <li>HTTP Cookie Manager</li> <li>HTTP Authorization Manager</li> <li>HTTP Authorization Manager</li> <li>Size Assertion</li> <li>Size Assertion</li> <li>Size Assertion</li> <li>Size Assertion</li> <li>Size Assertion</li> <li>Signin</li> <li>ViewCategory</li> <li>ViewProduct</li> <li>AddToCart</li> <li>Zogoff</li> <li>View Results Tree</li> <li>Jp@gc - Synthesis Report (filtered)</li> </ul>	<pre>- · · · · · · · · · · · · · · · · · · ·</pre>
Summary Report	<pre>13 14 15 16 17 18 19 19 19 19 19 10 10 10 10 10 10 10 10 10 10 10 10 10</pre>

#### BeanShell assertion code

```
debug();
long elapsed = SampleResult.getTime() ;
long threshold = Long.parseLong(vars.get("SLA_elapsedTime_threshold"));
if (elapsed > threshold) {
    int failureCount = Integer.parseInt(vars.get("SLA_elapsedTime_failures"));
    failureCount++;
    int maxErrors = Integer.parseInt(vars.get("SLA_elapsedTime_maxErrors"));
    if (failureCount >= maxErrors) {
            Failure = true;
            FailureMessage = "SF: failureCount" + " requests failed to finish in " + threshold + " ms";
        SampleResult.setSuccessful(false);
       SampleResult.setResponseMessage(failureCount + " requests failed to finish in " + threshold + " ms");
    } else {
        vars.put("SLA_elapsedTime_failures", String.valueOf(failureCount));
    };
    SampleResult.setResponseMessage("duration: "+elapsed+"; failureCount= "+failureCount);
}
```

After results are imported to Xray, we can see each sample result in the Test Run associated to the controller (i.e. HTTP Request sampler).

Calculator / Test Exe homepage	cution: CALC-6544 / Test: CALC-6537		Export Test as Text	Return to Test Execution		ł
Execution Status	FAIL			Assignee: Administra Executed By: Administra Tests - environments:	ator Versions or Revision	a - N -
Comment	Preview Comment	Execution Defects (0) Create Defect Create Sub-Task Add Defe	cts A Execution Evidence (0)		Add Evidence	~
Click to add cor	nment	No defects yet				
Execution	n Details					^
None						
Test Details						^
Test Type: Definition:	Generic jmeter.jpetstore.homepage					
Results						^
Context		Output		Duration	Status	
TestSuite .	JPetstore 1-6	SF: failureCount requests failed to finish in 10 ms		119.000 ms	FAIL	
TestSuite .	JPetstore 1-6	SF: failureCount requests failed to finish in 10 ms		115.000 ms	FAIL	
TestSuite .	JPetstore 1-6	SF: failureCount requests failed to finish in 10 ms		122.000 ms	FAIL	
				-		

## Room for improvement

- abstract the whole JMeter test plan as a Test
- use Robot Framework XML report instead of JUnit to provide more granular details
- provide the possibility of linking test(s) to an existing requirement in Xray
- implement SLAs on top of results

## References

- JMeter project
  JMeter Plugins Manager
  JMeter Plugins
  JMeter entities
  JMeter components reference
  Credentials Binding plugin
  Environment Injector plugin
  Jenkins Performance plugin
  Modified conversion utility "jmeter-junit-xml-report"
  How to attach files to an issue in Jira