

05 - Load testing

"It works on my machine" (but not only)

AMT 2020

Olivier Liechti

```
package ch.heigvd.amt.mvc.simple.presentation;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException;

@WebServlet(name = "CounterServlet", urlPatterns = "/counter")
public class CounterServlet extends HttpServlet {

    private int counter = 0;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String command = request.getParameter("command"); if ("reset".equals(command)) {
            counter = 0;
        } else {
            counter = counter + 1;
        }
        response.getWriter().println("this page has been accessed " + counter + " time(s).");
    }
}
```

```
package ch.heigvd.amt.mvc.simple.presentation;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet; import j
import javax.servlet.http.HttpServletRequest; import

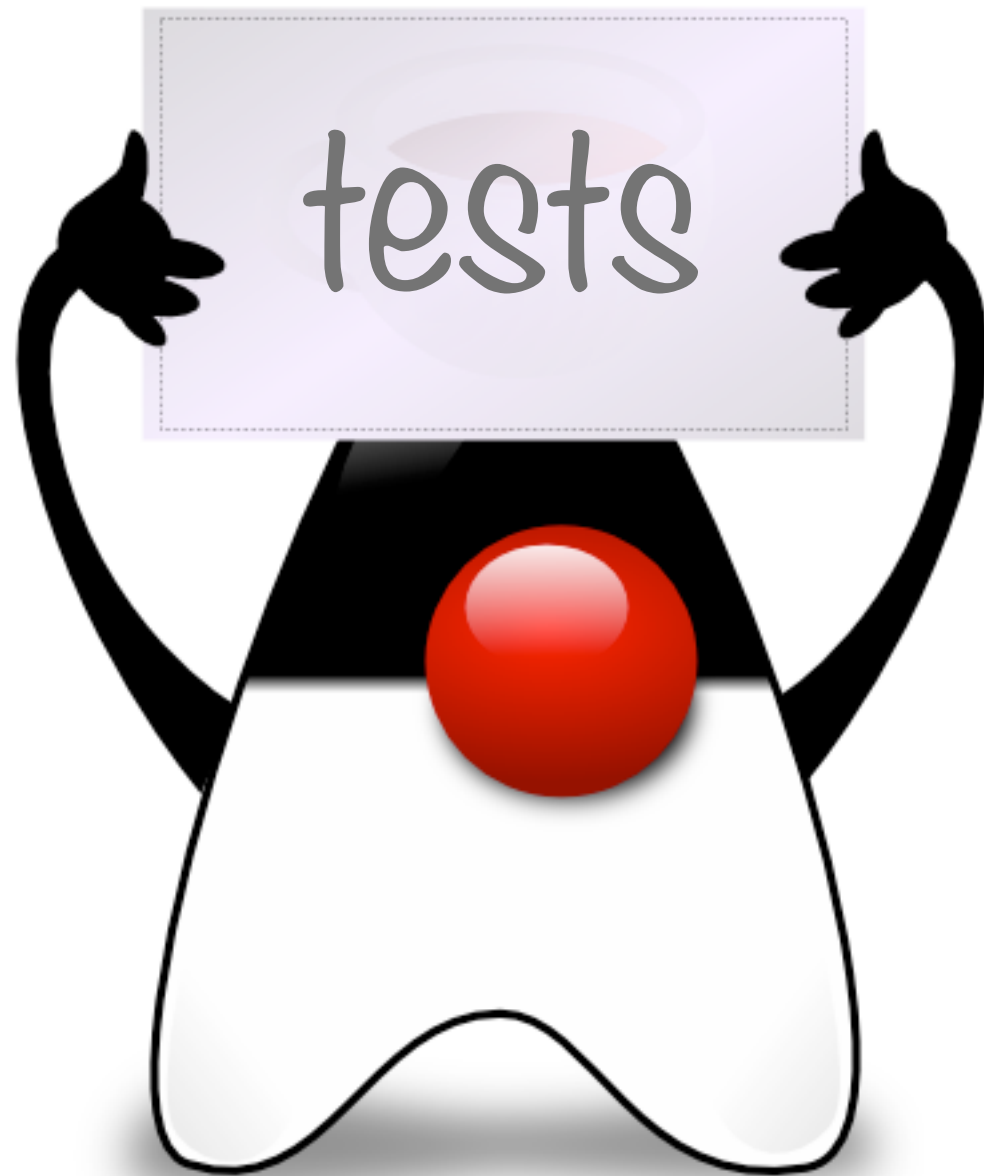
@WebServlet(name = "CounterServlet", urlPatterns = "
public class CounterServlet extends HttpServlet {

    private int counter = 0;

    protected void doGet(HttpServletRequest request, H
        String command = request.getParameter(s: "comman
            counter = 0;
        } else {
            counter = counter + 1;
        }
        response.getWriter().println("this page has been
    }
}
```



SEEMS LEGIT



Introduction to JMeter

- Open source project, apache foundation
- <http://jmeter.apache.org/index.html>



*“The Apache JMeter™ desktop application is open source software, a 100% pure Java application designed to **load test functional behavior and measure performance.***

*It was originally designed for **testing Web Applications** but has since **expanded to other** test functions.”*

*“Apache JMeter may be used to **test performance** both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers and more).*

*It can be used to **simulate a heavy load** on a server, network or object to test its strength or to analyze overall performance under **different load types**. You can use it to make a **graphical analysis of performance** or to test your server/script/object behavior under heavy **concurrent** load.”*

Types of tests (1)

- **Functional tests**

- Is the system doing what it is supposed to do?
- Does its behavior comply with functional requirements (use cases)?
- Selenium is a tool for automating functional testing of web applications (<http://seleniumhq.org/>)

- **Performance, load and stress tests**

- What is the response time? What is the consumption of resources? Are there issues (e.g. concurrency issues) that happen under load?
- Relevant both for interactive and batch use cases.

What to install?

- **Main project**
 - http://jmeter.apache.org/download_jmeter.cgi
- **Add-ons**
 - <http://jmeter-plugins.org/>

Standard Set

Basic plugins for everyday needs. Does not require additional libs to run.

[Download](#) | [Installation](#) | [Package Contents](#)



Extras Set

Additional plugins for extended and complex testing. Does not require additional libs to run.

[Download](#) | [Installation](#) | [Package Contents](#)



Extras with Libs Set

Additional plugins that *do require* additional libs to run.

[Download](#) | [Installation](#) | [Package Contents](#)



WebDriver Set

Selenium/WebDriver testing ability.

[Download](#) | [Installation](#) | [Package Contents](#)



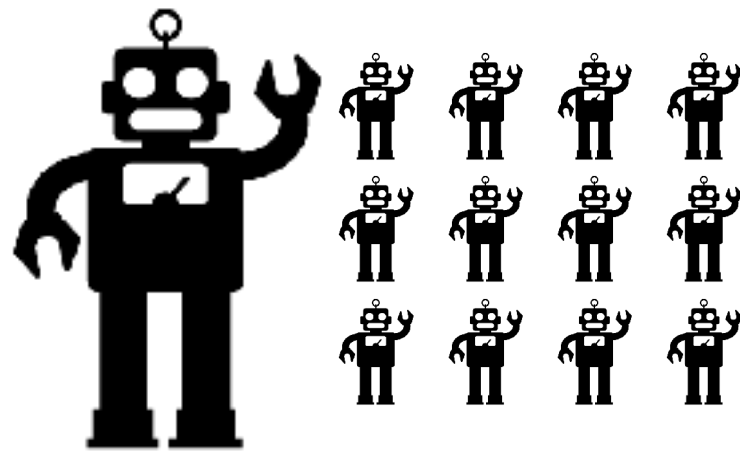
Hadoop Set

Hadoop/HBase testing plugins.

[Download](#) | [Installation](#) | [Package Contents](#)



JMeter Building Blocks



ThreadGroup



Test Plan



Listeners
(results & stats)



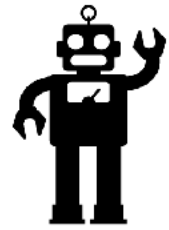
Samplers
(actions)



Logic Controllers
& Assertions

Concepts

- Test Plan
- ThreadGroup
- Samplers
- Logic Controllers
- Listeners
- Timers
- Assertions
- Configuration Elements
- Pre-Processor Elements
- Post-Processor Elements



*“Thread group elements are the **beginning points of any test plan**. All controllers and samplers must be under a thread group. [...]. As the name implies, the thread group element controls the number of threads JMeter will use to execute your test. The controls for a thread group allow you to:*

- Set the **number of threads**
- Set the **ramp-up** period
- Set the **number of times** to execute the test

*Each thread will execute the test plan in its entirety and completely independently of other test threads. **Multiple threads are used to simulate concurrent connections to your server application.**”*



*“Samplers tell JMeter to **send requests to a server and wait for a response**. They are processed in the order they appear in the tree. Controllers can be used to modify the number of repetitions of a sampler.*

- *FTP Request*
- ***HTTP Request***
- *JDBC Request*
- *Java object request*
- *LDAP Request*
- *SOAP/XML-RPC Request*
- *WebService (SOAP) Request*

*Each sampler has several **properties** you can set. You can further customize a sampler by adding one or more Configuration Elements to the Test Plan.”*



*“Logic Controllers let you customize **the logic that JMeter uses to decide when to send requests**. Logic Controllers can change the order of requests coming from their child elements. They can modify the requests themselves, cause JMeter to repeat requests, etc.”*

- *Loop Controller*
- *Once Only Controller*
- *Interleave Controller*
- *Random Controller*
- *Random Order Controller*
- *Throughput Controller*
- *Runtime Controller*
- *If Controller*
- *etc.*



*“Listeners provide **access to the information JMeter gathers about the test cases** while JMeter runs. The **Graph Results** listener plots the response times on a graph. The “**View Results Tree**” Listener shows details of sampler requests and responses, and can display basic HTML and XML representations of the response. Other listeners provide **summary or aggregation information**.*

*Additionally, listeners can **direct the data to a file** for later use.*

Listeners can be added anywhere in the test, including directly under the test plan. They will collect data only from elements at or below their level.”

*“By default, a JMeter thread sends requests without pausing between each request. We recommend that you specify a delay by adding one of the available timers to your Thread Group. If you do not add a delay, JMeter could **overwhelm your server** by making too many requests in a very short amount of time.*

The timer will cause JMeter to delay a certain amount of time before each sampler which is in its scope .

If you choose to add more than one timer to a Thread Group, JMeter takes the sum of the timers and pauses for that amount of time before executing the samplers to which the timers apply. Timers can be added as children of samplers or controllers in order to restrict the samplers to which they are applied.

*To provide a pause at a single place in a test plan, one can use the **Test Action Sampler.**”*

Assertions



*“Assertions allow you to **assert facts about responses received** from the server being tested.*

*Using an assertion, you can essentially **"test" that your application is returning the results you expect it to.***

*For instance, you can assert that the response to a query will **contain some particular text.** The text you specify can be a Perl-style regular expression, and you can indicate that the response is to contain the text, or that it should match the whole response.*

*You can add an assertion to any Sampler. For example, you can add an assertion to a HTTP Request that checks for the text, "</HTML>". JMeter will then check that the text is present in the HTTP response. If JMeter cannot find the text, then it will **mark this as a failed request.**”*

“A configuration element works closely with a Sampler. Although it does not send requests (except for HTTP Proxy Server), it can add to or modify requests.

*A configuration element is accessible from only inside the tree branch where you place the element. For example, if you place an **HTTP Cookie Manager** inside a Simple Logic Controller, the Cookie Manager will only be accessible to HTTP Request Controllers you place inside the Simple Logic Controller”*

- *HTTP Authorization Manager*
- *HTTP Cache Manager*
- *HTTP Cookie Manager*
- *HTTP Request Defaults*
- *HTTP Header Manager*

How to Create Test Scenarios?

- **Option 1 : manually**
 - Create a Test Plan
 - Add a Thread Group
 - Add HTTP samplers and specify HTTP request parameters
- **Option 2 : recording with JMeter configured as an HTTP proxy**
 - http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.pdf
 - Do manual adjustments

- **Use variables:**

- Very often, it is needed to use parts of an HTTP response into follow-up requests (e.g. session ids).
- http://jmeter.apache.org/usermanual/test_plan.html#properties

- **Use more than one machine:**

- With JMeter running on a single machine, it is common to “exhaust” the client before the server (especially when testing a “real” infrastructure with multiple nodes).
- For real performance tests, it is therefore recommended to use multiple machines for injecting load into the network. One way to do it is use use virtual machines in a cloud environment (e.g. on Amazon EC2).
- Multiple JMeter clients can be coordinated by a master and results can be collected and aggregated (http://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf)

To run the example

- **Install JMeter**
 - <http://jmeter.apache.org/>
- **Install the JMeter plugin manager**
 - <https://jmeter-plugins.org/install/Install/>
- Create a test plan, with at the minimum:
 - An **ultimate thread group** (plugin to install with the plugin manager)
 - An **HTTP request sampler**
 - A **constant timer**
 - **Listeners** to display the results (play with “summary report”, “response time graph”, “view results in tree”).

Exe

• Ins

• Ins

•

E GESTION
UD

3 Basic Graphs

5 Additional Graphs

BM-Sense Uploader

Command-Line Graph Plotting Tool

Composite Timeline Graph

Custom JMeter Functions

Custom Thread Groups

Distribution/Percentile Graphs

Dummy Sampler

FTP Protocol Support

Flexible File Writer

Graphs Generator Listener

HTTP Protocol Support

Inter-Thread Communication

JDBC Support

JMS Support

JMeter 'Monitors' (Deprecated)

JMeter Core

JSON Plugins

JUnit Support

Java Components

LDAP Protocol Support

Mail/SMTP Support

MongoDB Support

OS Process Support

PerfMon (Servers Performance Monitoring)

Plugins Manager

Selenium/WebDriver Support

TCP Protocol Support

Throughput Shaping Timer

Various Core Components

jpgc - Standard Set

Installed Plugins

Available Plugins

Upgrades

Custom Thread Groups

Vendor: *JMeter-Plugins.org*

Adds new Thread Groups:

Stepping Thread Group

Ultimate Thread Group

Concurrency Thread Group

Arrivals Thread Group

Free-Form Arrivals Thread Group

Documentation: <https://jmeter-plugins.org/wiki/ConcurrencyThreadGroup/>

jpgc - Concurrency Thread Group

Name: jpgc - Concurrency Thread Group

Comments:

le p on this plugin

v1.1.0

Action to be taken after a Sampler error:

Continue

Start Next Thread Loop

Stop Thread

Stop Test

Stop Test Now

Target Concurrency: 2

Ramp Up Time (sec): 60

Ramp-Up Steps Count: 3

Hold Target Rate Time (sec): 20

Concurrency Threads

Speed / Performance of Concurrency Threads

20

18

16

14

12

10

8

6

4

2

0

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

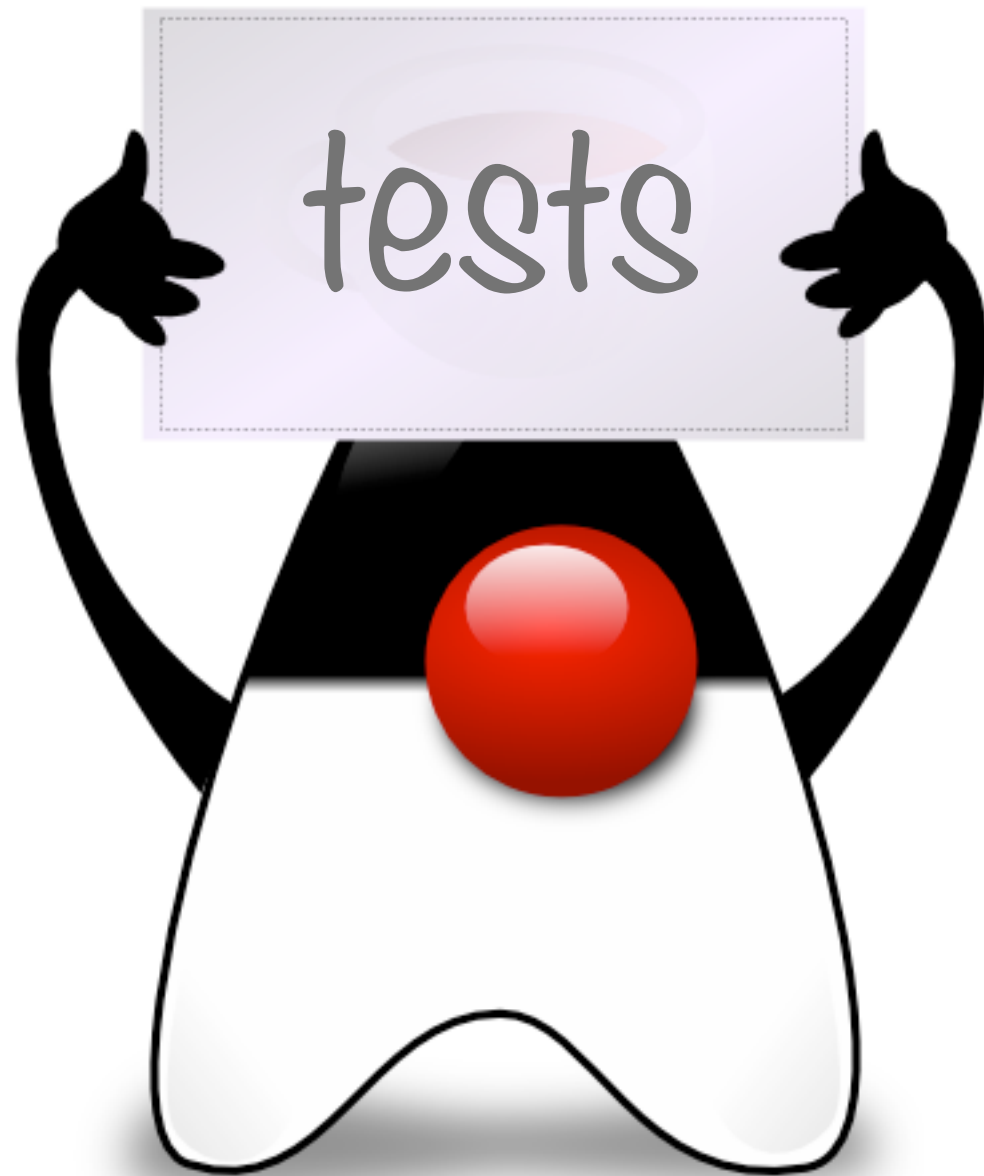
20

Version: 2.1

Review Changes

Uninstall plugin: jpgc-standard 2.0

Apply Changes and Restart JMeter



Applying JMeter to our use case


```
package ch.heigvd.amt.mvc.simple.presentation;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException;

@WebServlet(name = "CounterServlet", urlPatterns = "/counter")
public class CounterServlet extends HttpServlet {

    private int counter = 0;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String command = request.getParameter("command"); if ("reset".equals(command)) {
            counter = 0;
        } else {
            counter = counter + 1;
        }
        response.getWriter().println("this page has been accessed " + counter + " time(s).");
    }
}
```

