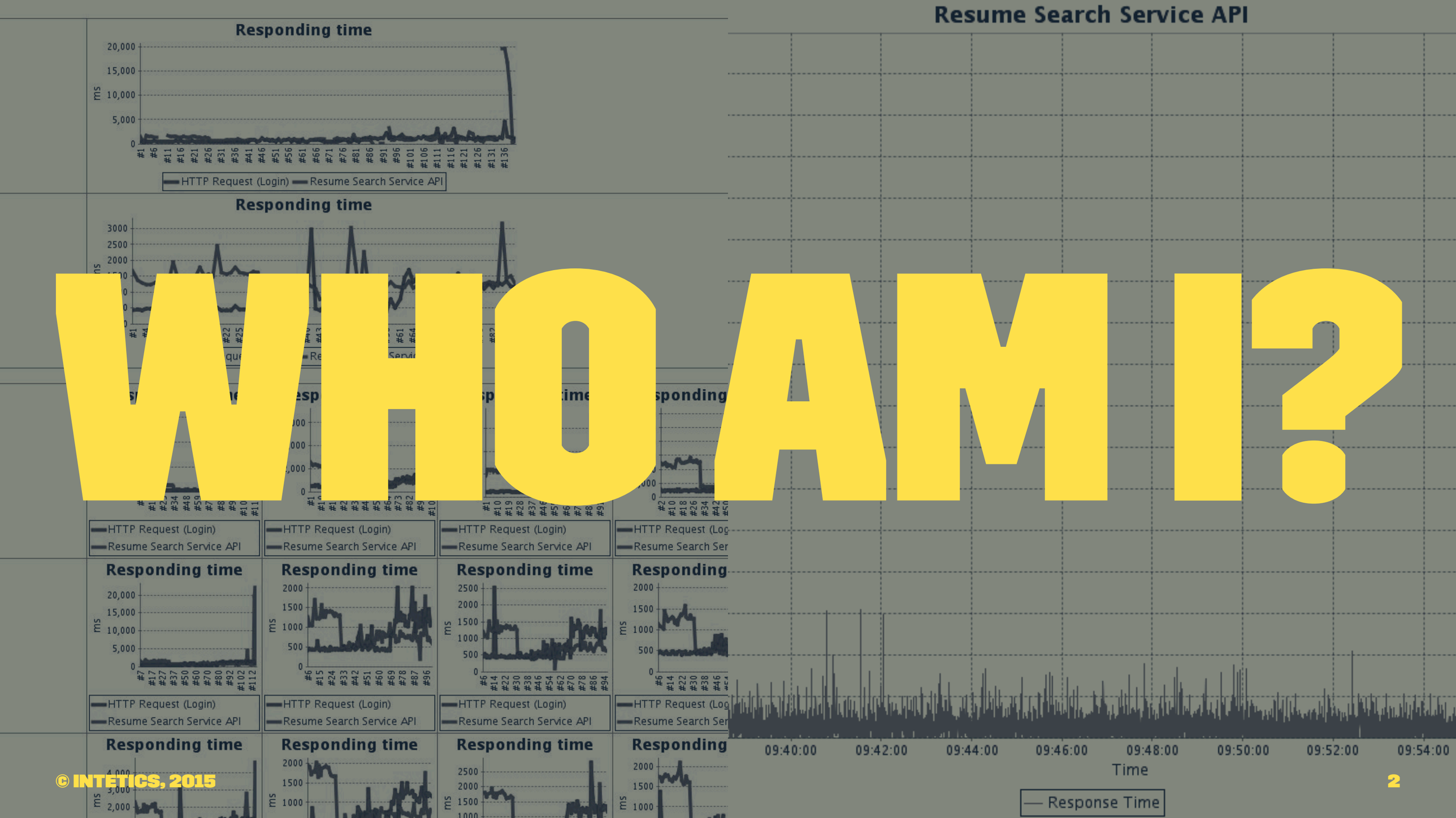
PERFORMANCE TESTING FOR EVERYONE

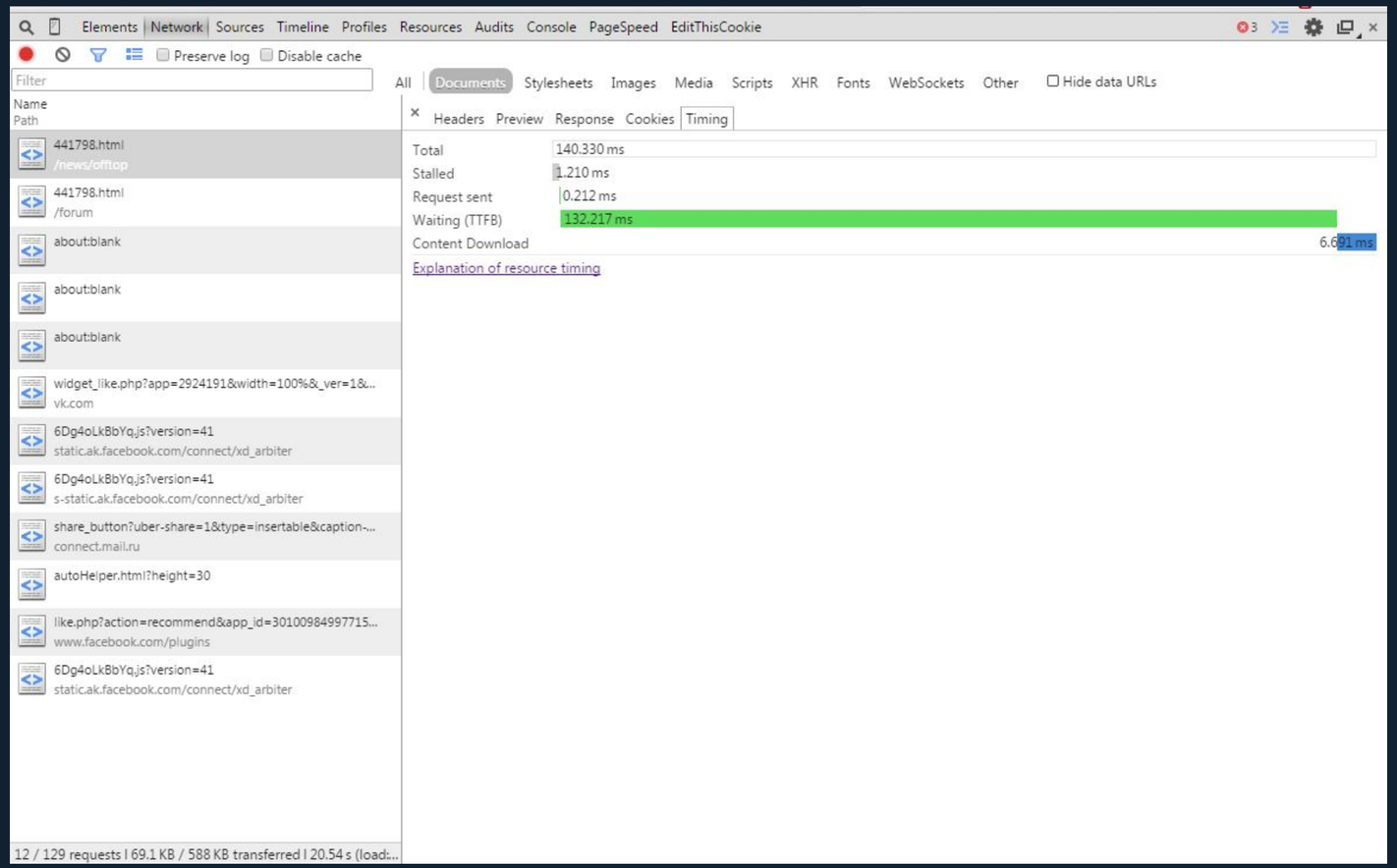


WHO AM I?

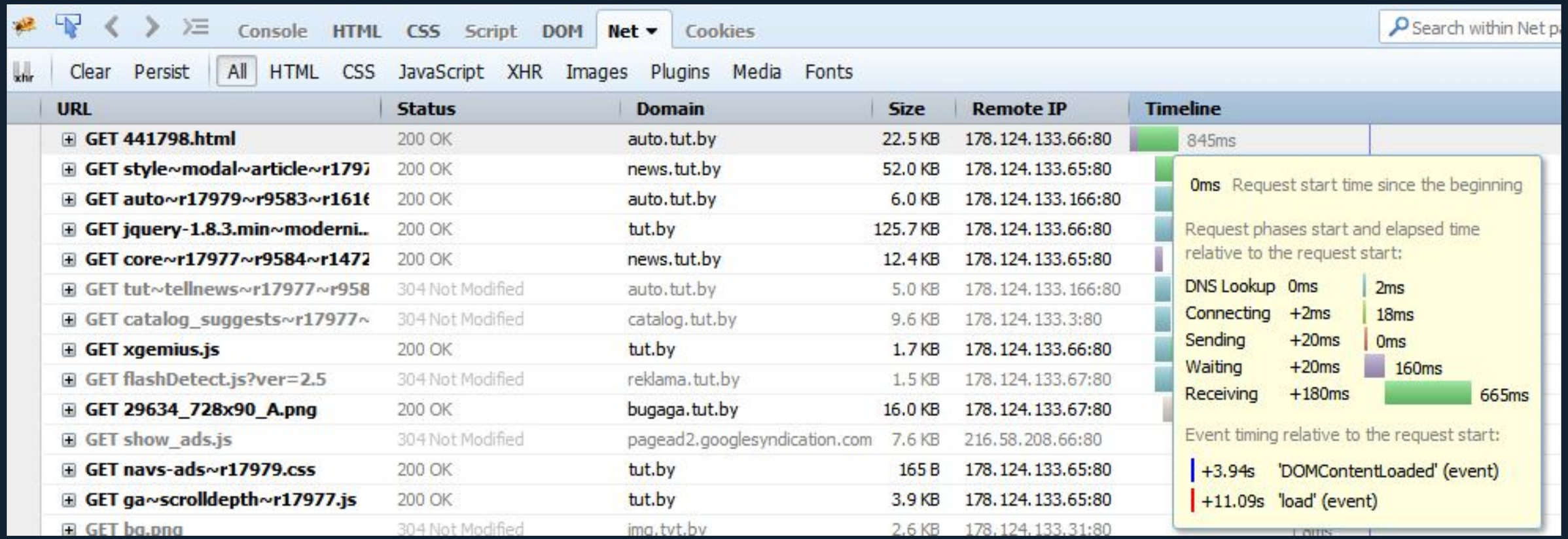
NAIVE APPROACH

F5

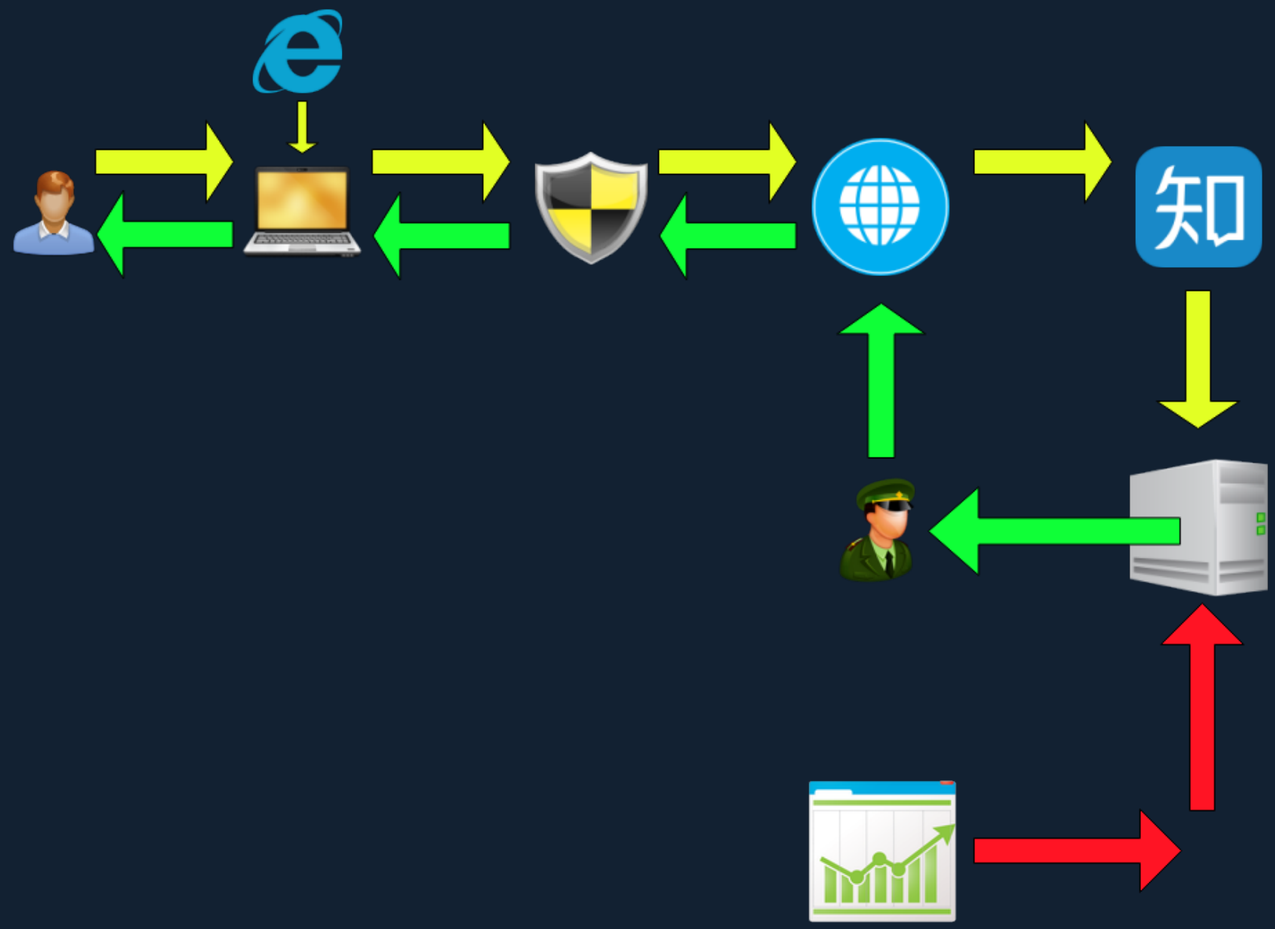
A BIT SMARTER APPROACH
CHROME



A BIT SMARTER APPROACH
FIREFOX / FIREBUG



FUNNY NETWORK DIAGRAM



RIGHT SOULTION

JMETER

AGENDA

» elements of a performance test

» what is a load

» testing RESTful services

» testing 'like a web browser'

» passing variables to performance tests

» configuring Jenkins to run performance tests

AGENDA (EXPLAINED)

```
...
- testing RESTful services
 - authentication
 - sampling
 - variables VS properties
 - custom errors
 - logging
...
```

AGENDA (EXPLAINED)

```
...
- testing 'like a web browser'
 - authentication
 - session support
 - sampling
...
```

PREREQUISITS

» java 1.7

» maven 3

» git-scm or cygwin for Windows users

» jenkins

LET'S START[1]

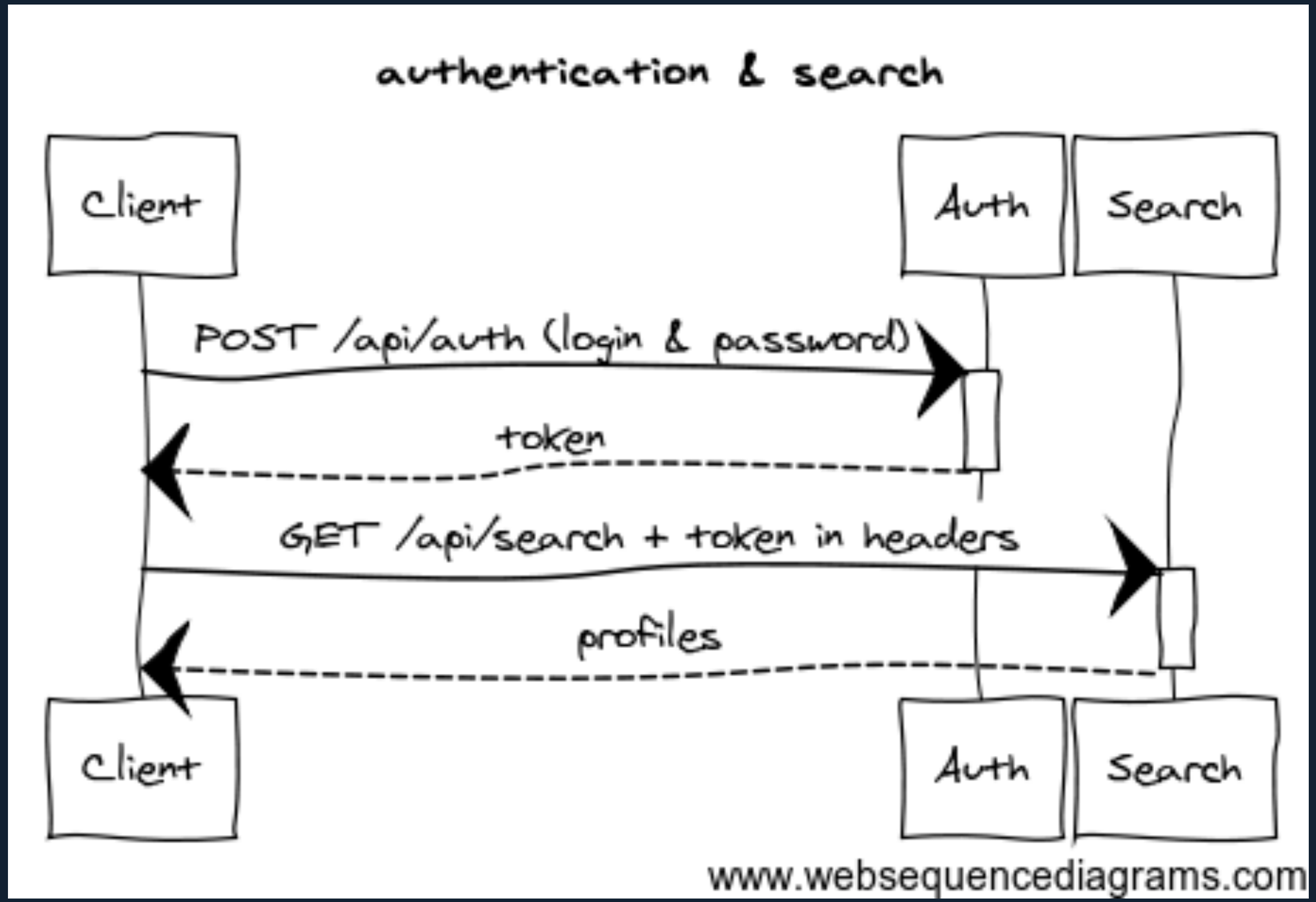
petapp (auth and search RESTful services)

```
petapp$ mvn spring-boot:run
```

petweb (web app with login form)

```
petweb$ mvn spring-boot:run -Dserver.port=9090
```

[1] showtime



AUTHENTICATION

```
$ curl localhost:8080/api/auth -d 'login=user&password=user' -X POST
```

RESPONSE

```
<AuthToken xmlns="">
    <token>ppr12c2u694b9hi2is7cqkalva</token>
    <expirationDate>1427717457176</expirationDate>
</AuthToken>
```

SEARCH REQUEST

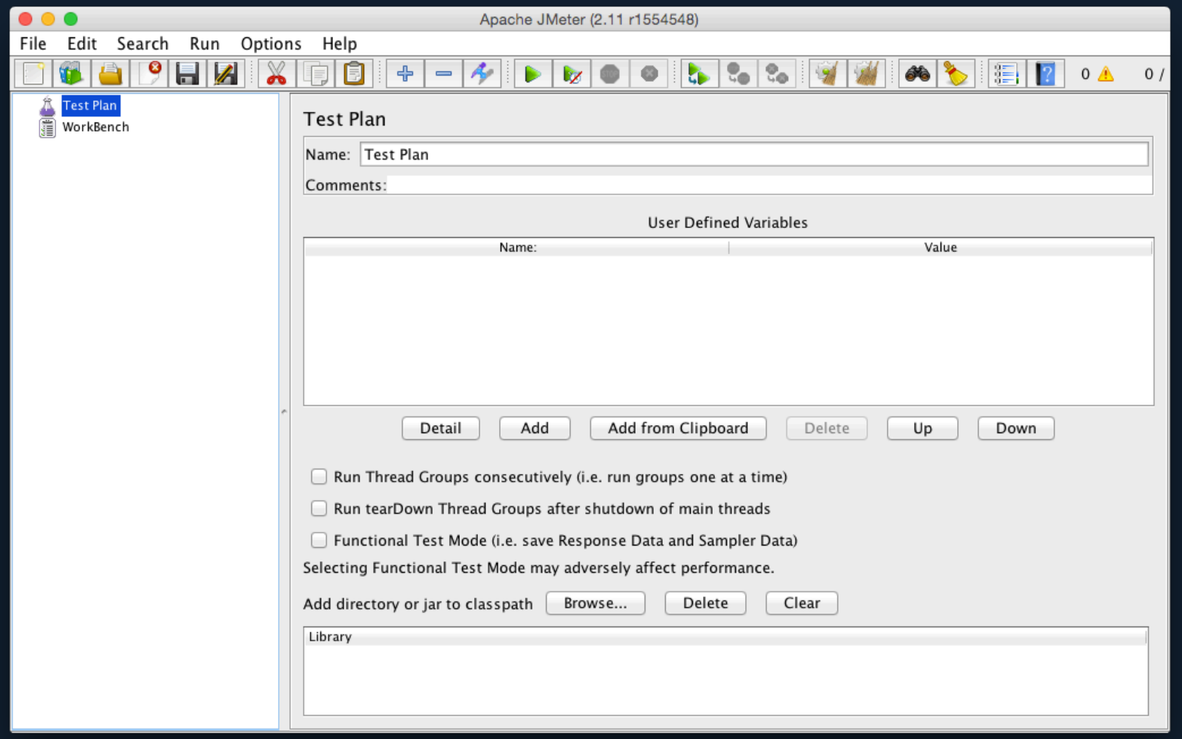
```
$ curl localhost:8080/api/search -H "token: pprl2c2u694b9hi2is7cqkalva"
```

SEARCH RESPONSE


```
{
  "page": 1,
  "items": [
    {
      "id": 6592,
      "title": "Oil & Gas Drilling Supervisor",
      "content": "B",
      "location": "C"
    },
...
  ]
}
```

JMETER GUI

```
tests$ mvn jmeter:gui
```



AGENDA

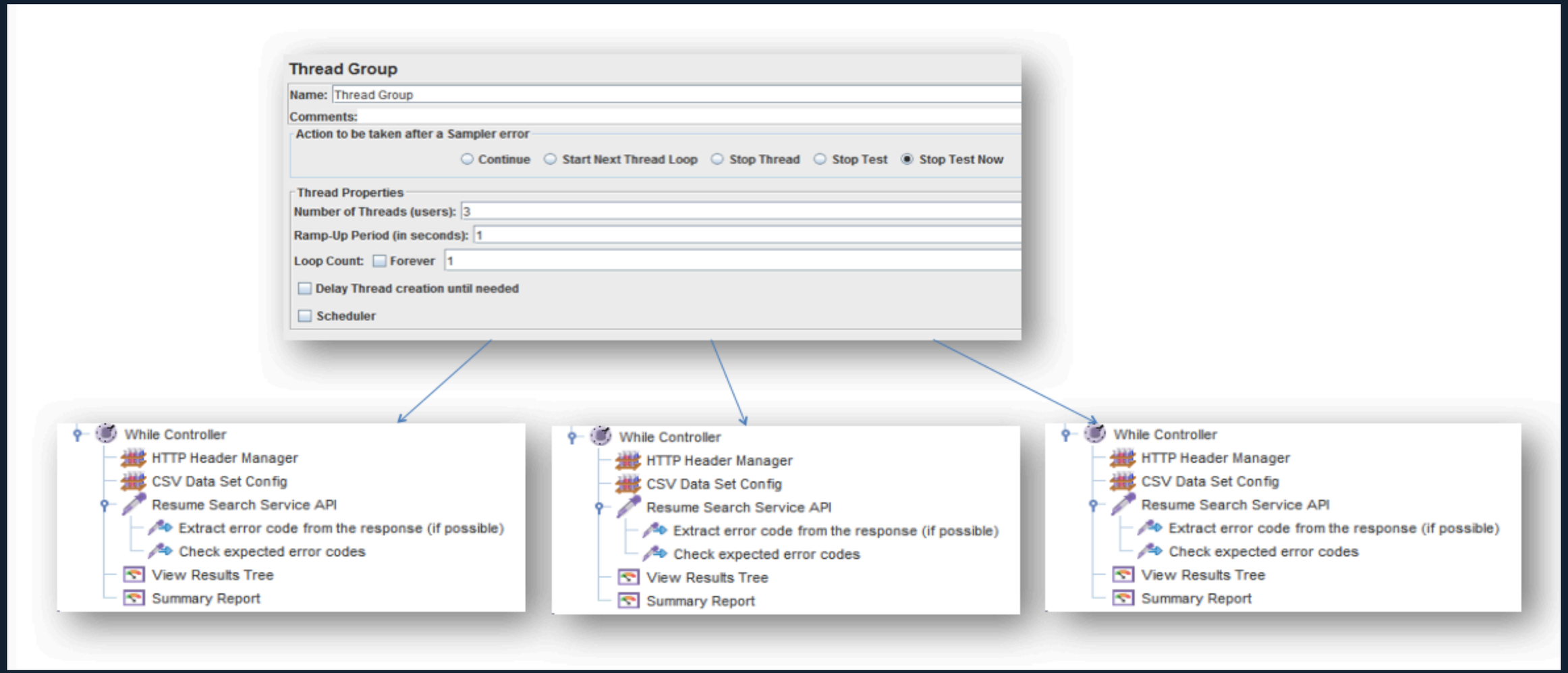
» elements of a performance test

...

THREAD GROUP

“Thread group elements are the beginning points of any test plan. All controllers and samplers must be under a thread group. Other elements, e.g. Listeners, may be placed directly under the test plan, in which case they will apply to all the thread groups.”
Jmeter documentation

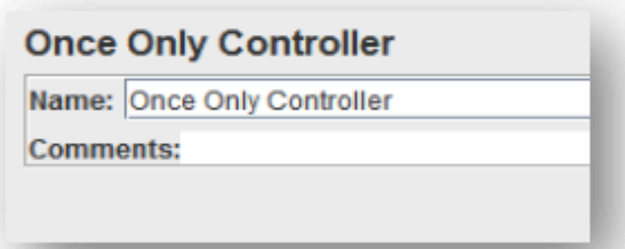
THREAD GROUP

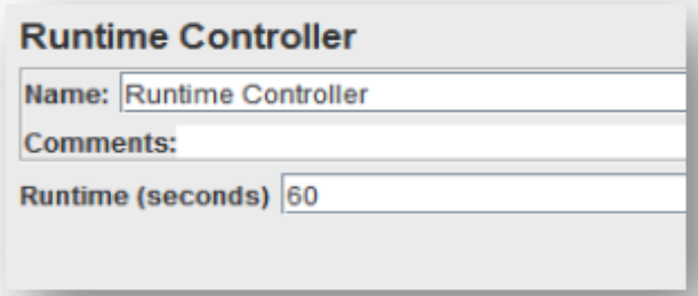


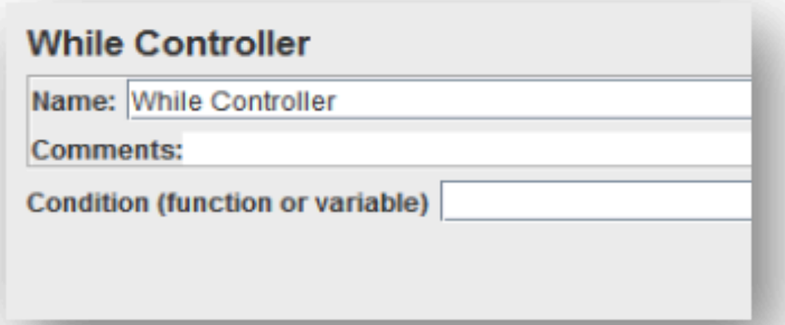
CONTROLLERS

“JMeter has two types of Controllers: Samplers and Logical Controllers. These drive the processing of a test. Samplers tell JMeter to send requests to a server. Logical Controllers let you customize the logic that JMeter uses to decide when to send requests.”
JMeter documentation

NECESSARY CONTROLLERS







AGENDA

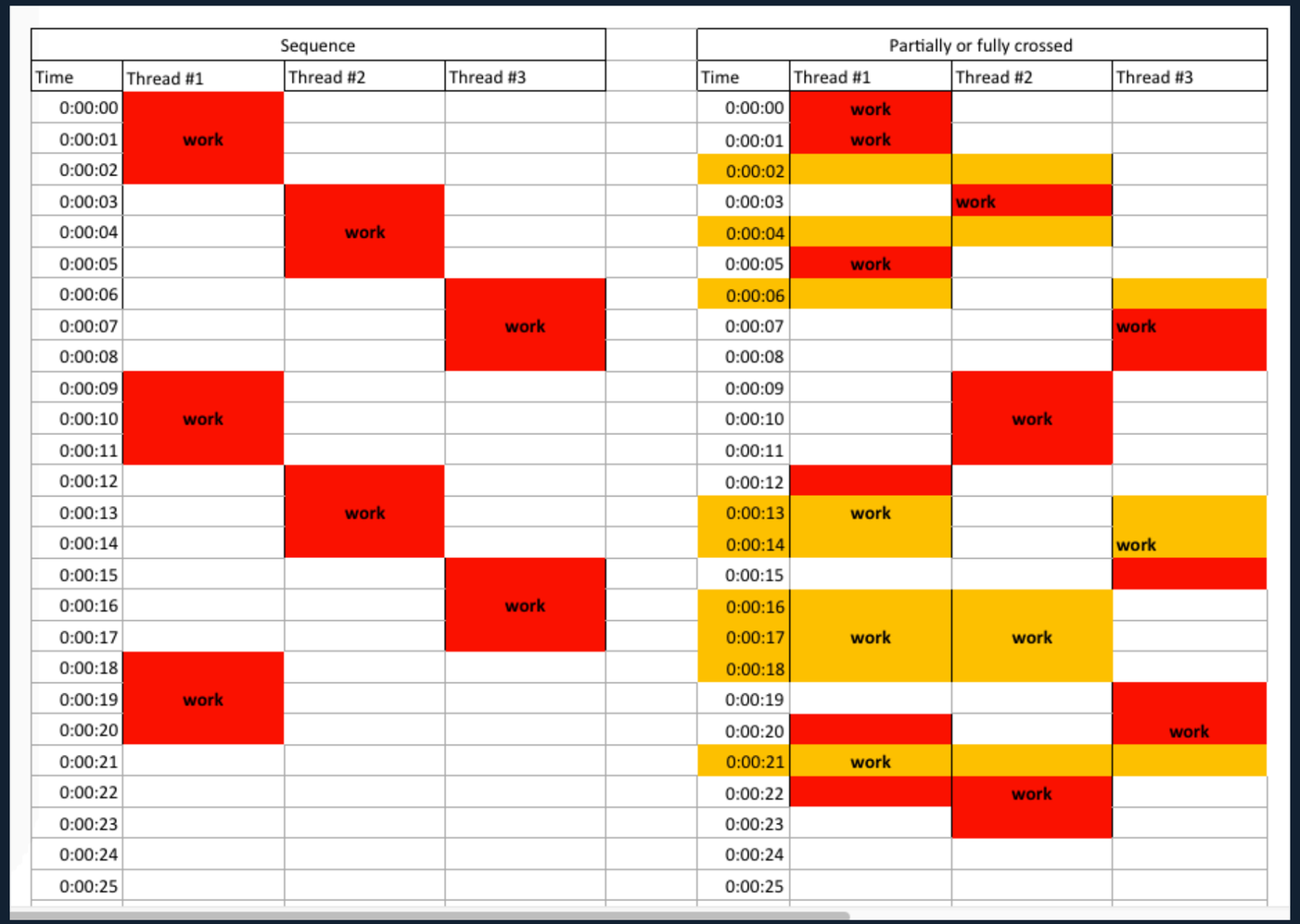
...

- what is a load

...

1 000 000 USERS

THREADS



Sequence			
Time	Thread #1	Thread #2	Thread #3
0:00:00	work		
0:00:01			
0:00:02			
0:00:03		work	
0:00:04			
0:00:05			
0:00:06			work
0:00:07			
0:00:08			
0:00:09	work		
0:00:10			
0:00:11			
0:00:12		work	
0:00:13			
0:00:14			
0:00:15			work
0:00:16			
0:00:17			
0:00:18	work		
0:00:19			
0:00:20			
0:00:21			
0:00:22			
0:00:23			
0:00:24			
0:00:25			

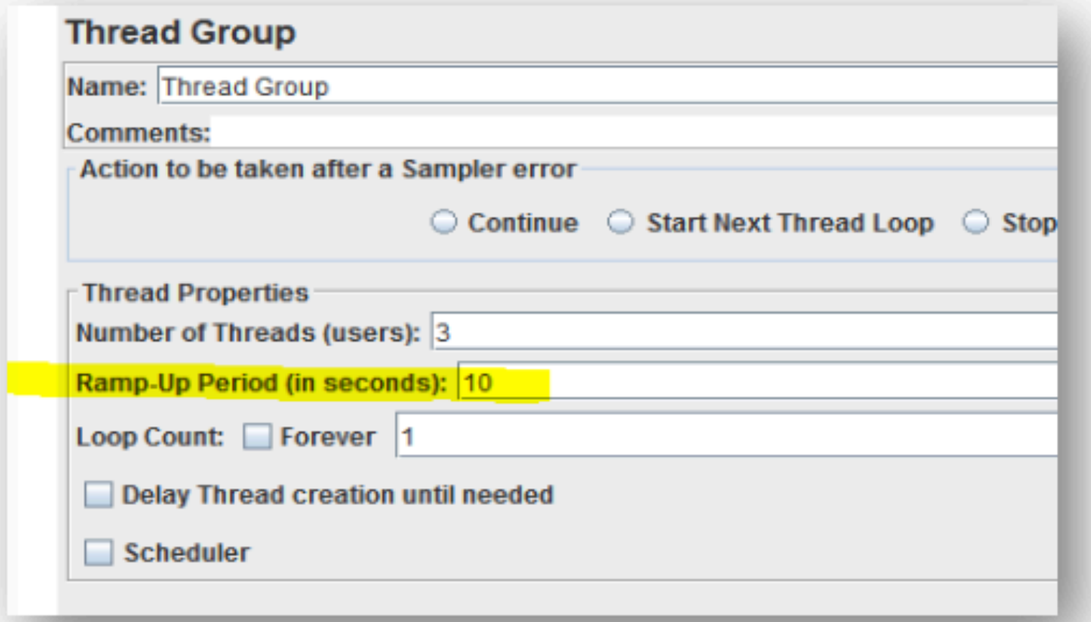
Partially or fully crossed			
Time	Thread #1	Thread #2	Thread #3
0:00:00	work		
0:00:01	work		
0:00:02			
0:00:03		work	
0:00:04			
0:00:05	work		
0:00:06			
0:00:07			work
0:00:08			
0:00:09		work	
0:00:10			
0:00:11			
0:00:12			
0:00:13	work		
0:00:14			work
0:00:15			
0:00:16		work	
0:00:17	work		
0:00:18			
0:00:19			work
0:00:20			
0:00:21	work		
0:00:22		work	
0:00:23			
0:00:24			
0:00:25			

RAMP-UP PERIOD

“Ramp-up Period - How long JMeter should take to get all the threads started. If there are 10 threads and a ramp-up time of 100 seconds, then each thread will begin 10 seconds after the previous thread started, for a total time of 100 seconds to get the test fully up to speed.”

JMeter documentation

RAMP-UP PERIOD



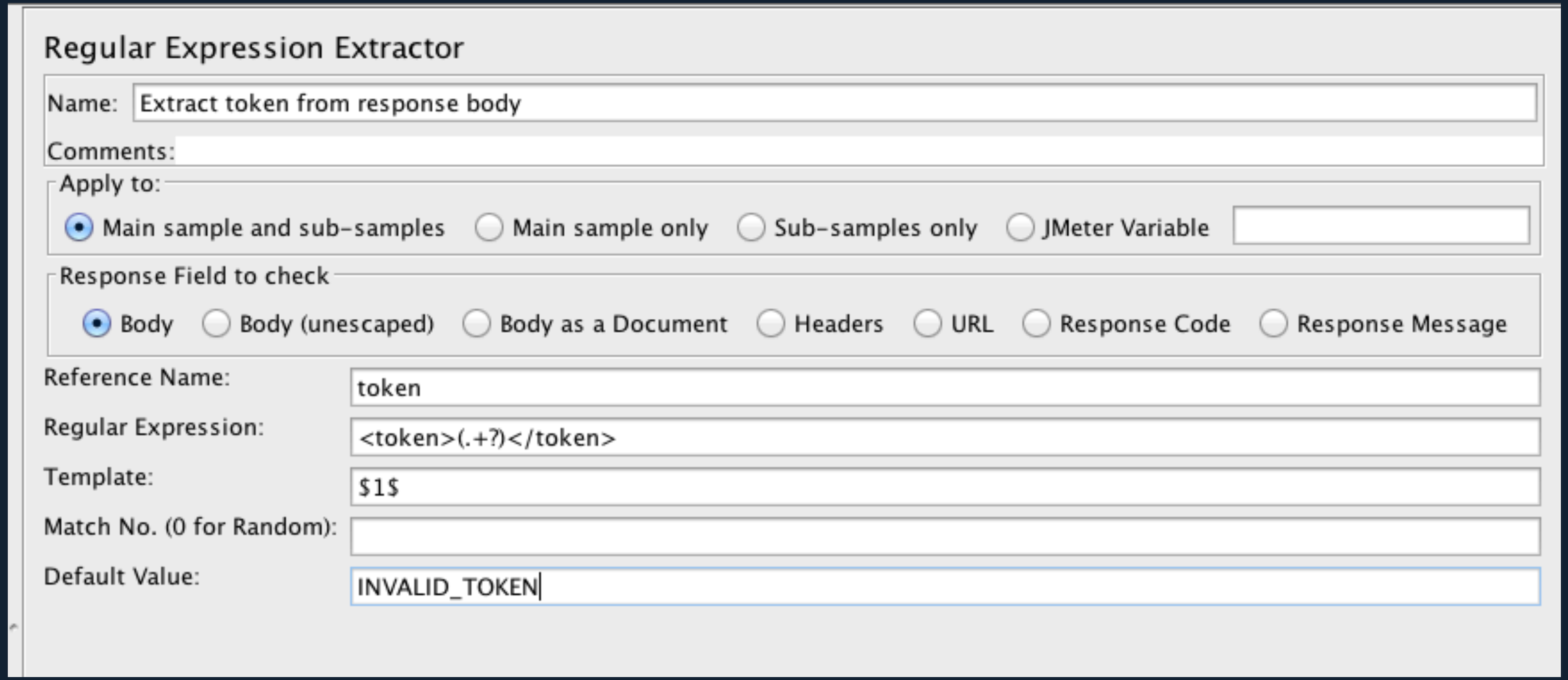
	Sequence			
	Time	Thread #1	Thread #2	Thread #3
Ramp-Up Period	0:00:00	work		
	0:00:01			
	0:00:02			
	0:00:03			
	0:00:04		work	
	0:00:05			
	0:00:06	work		
	0:00:07			
	0:00:08			
	0:00:09			work
	0:00:10	work		
	0:00:11		work	
	0:00:12			
	0:00:13			
	0:00:14			
	0:00:15			

AGENDA (EXTENDED)[1]

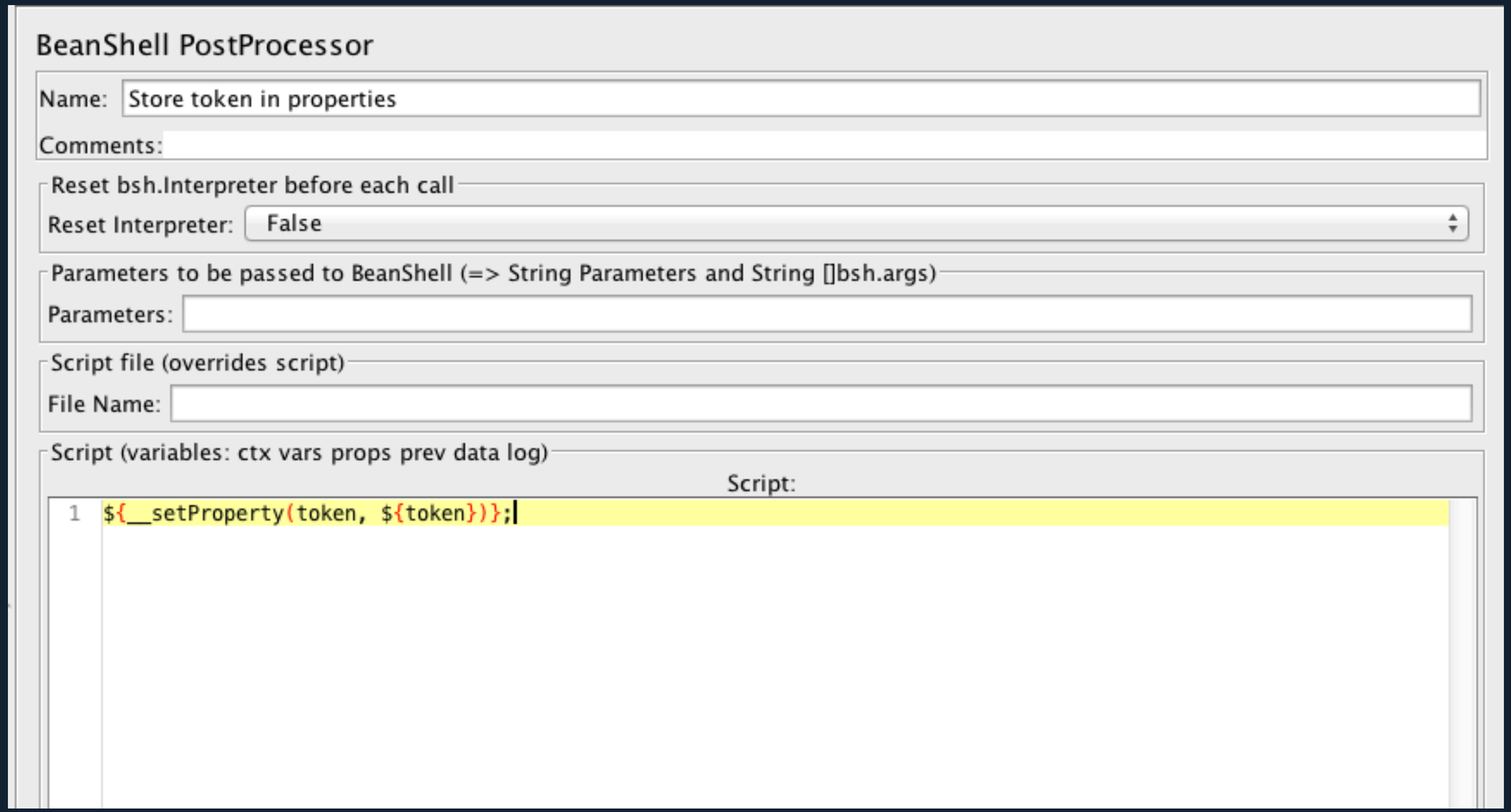
```
...
- testing RESTful services
 - authentication
 - sampling
 ...
```

[1] showtime

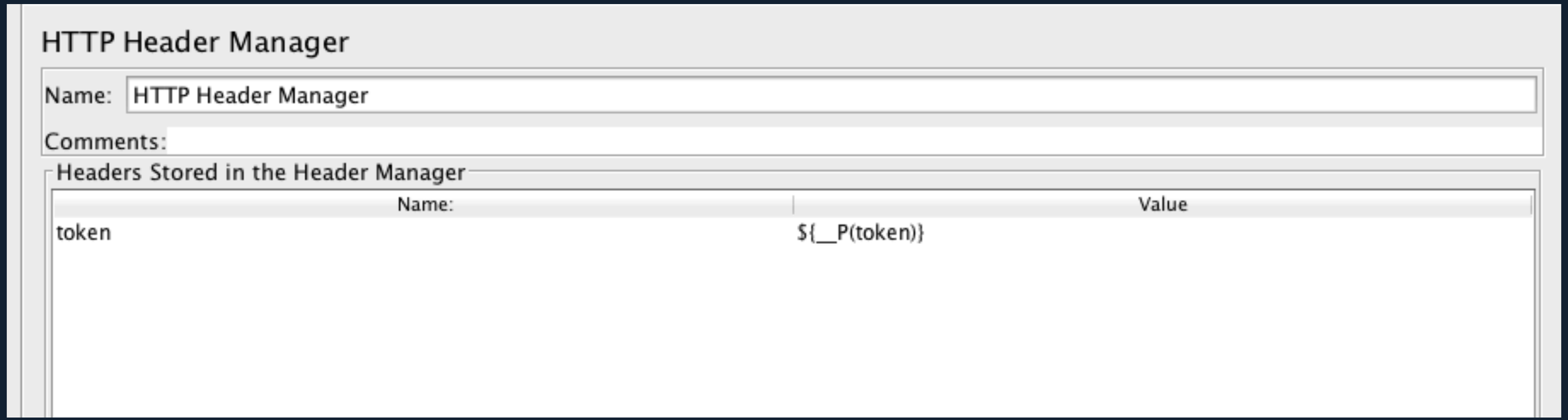
AUTHENTICATION



AUTHENTICATION

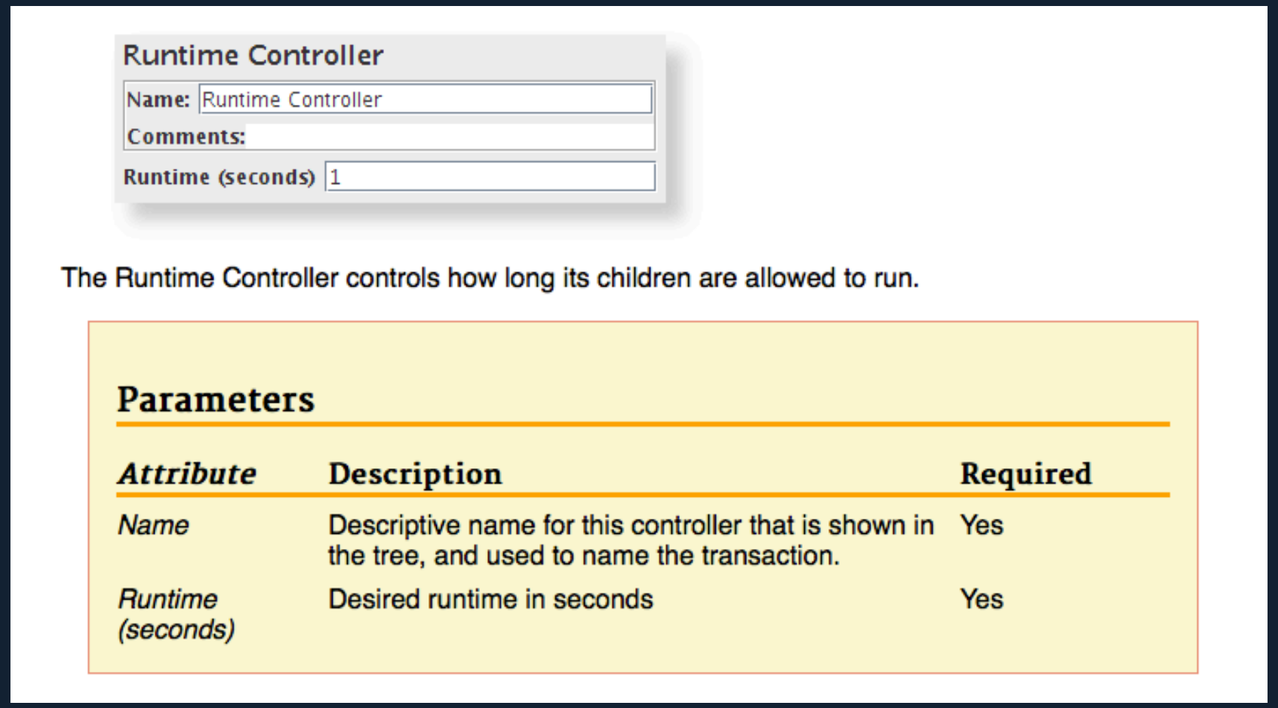


PASSING HEADERS



"The Header Manager lets you add or override HTTP request headers."

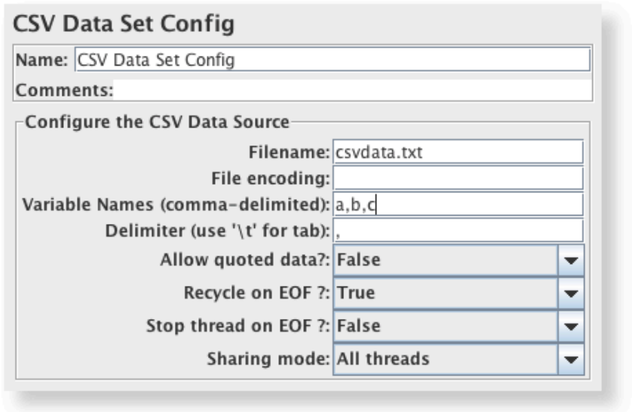
HOW TO RUN A TEST PLAN FOR A CERTAIN AMOUNT OF TIME



JMeter documentation

CSV DATA SET CONFIG

CSV Data Set Config



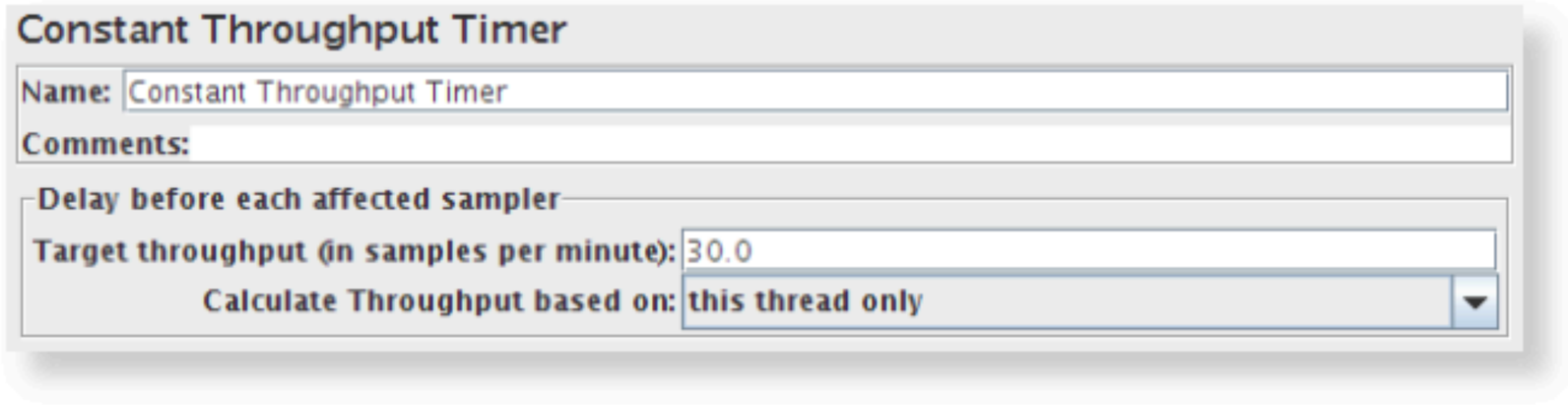
CSV Data Set Config is used to read lines from a file, and split them into variables. It is easier to use than the __CSVRead() and _StringFromFile() functions. It is well suited to handling large numbers of variables, and is also useful for tesing with "random" and unique values. Generating unique random values at run-time is expensive in terms of CPU and memory, so just create the data in advance of the test. If necessary, the "random" data from the file can be used in conjunction with a run-time parameter to create different sets of values from each run - e.g. using concatenation - which is much cheaper than generating everything at run-time.

Versions of JMeter after 2.3.1 allow values to be quoted; this allows the value to contain a delimiter. Previously it was necessary to choose a delimiter that was not used in any values. If "allow quoted data" is enabled, a value may be enclosed in double-quotes. These are removed. To include double-quotes within a quoted field, use two double-quotes. For example:

JMeter documentation

CONSTANT THROUGHPUT TIMER

Constant Throughput Timer



This timer introduces variable pauses, calculated to keep the total throughput (in terms of samples per minute) as close as possible to a give figure. Of course the throughput will be lower if the server is not capable of handling it, or if other timers or time-consuming test elements prevent it.

Jmeter documentation

CONSTANT THROUGHPUT TIMER

this thread only - each thread will try to maintain the target throughput. The overall throughput will be proportional to the number of active threads.

all active threads in current thread group - the target throughput is divided amongst all the active threads in the group. Each thread will delay as needed, based on when it last ran.

CONSTANT THROUGHPUT TIMER

all active threads - the target throughput is divided amongst all the active threads in all Thread Groups. Each thread will delay as needed, based on when it last ran. In this case, each other Thread Group will need a Constant Throughput timer with the same settings.

CONSTANT THROUGHPUT TIMER

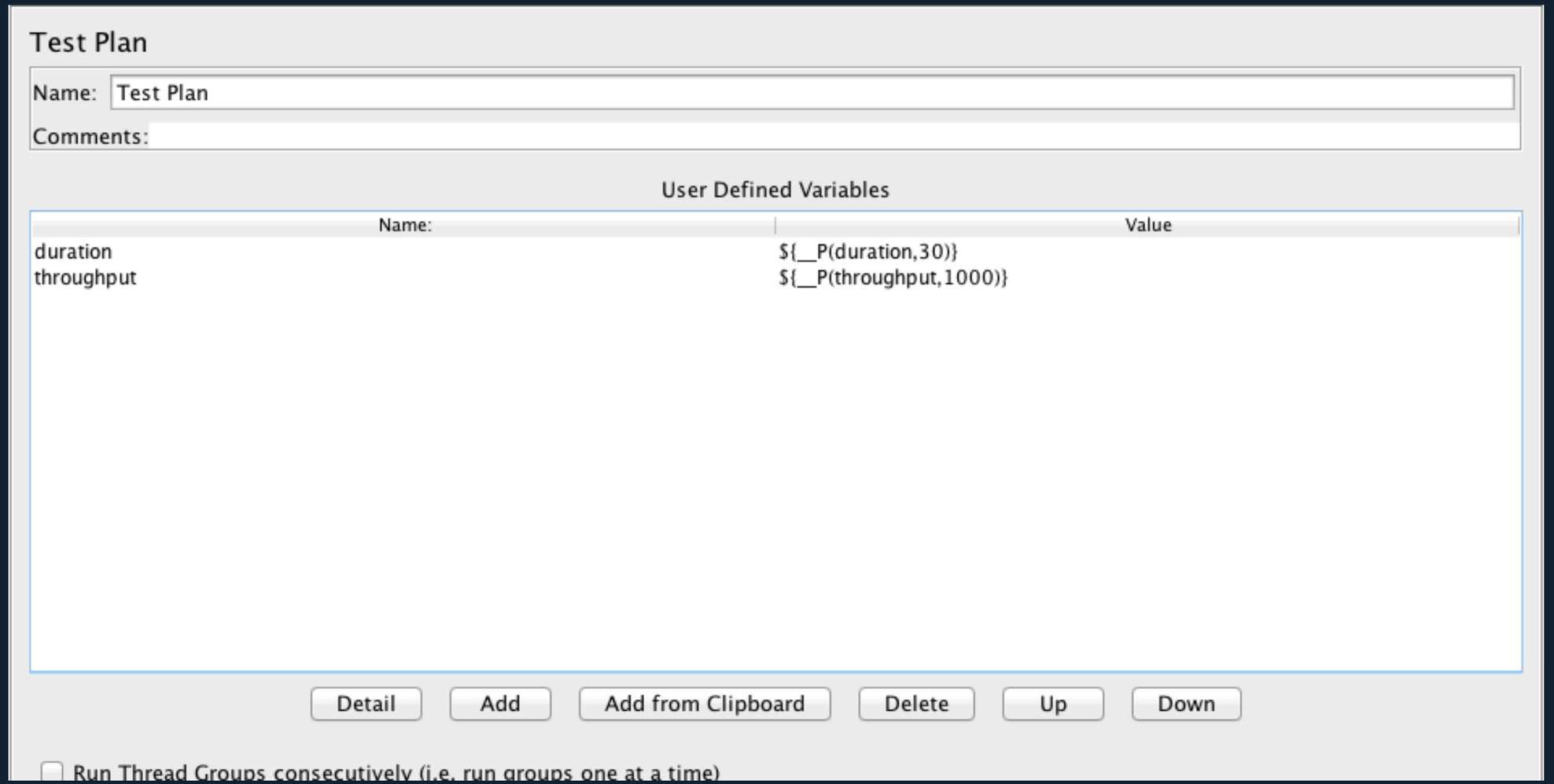
all active threads in current thread group (shared) - as above, but each thread is delayed based on when any thread in the group last ran.

all active threads (shared) - as above; each thread is delayed based on when any thread last ran.

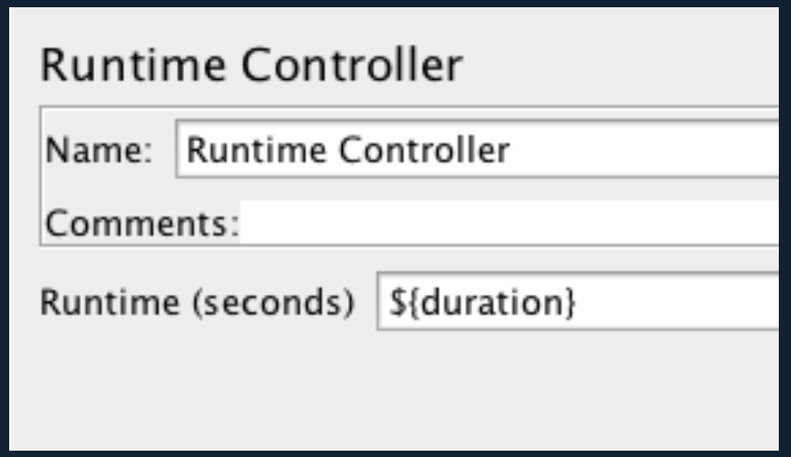
AGENDA (EXPLAINED)

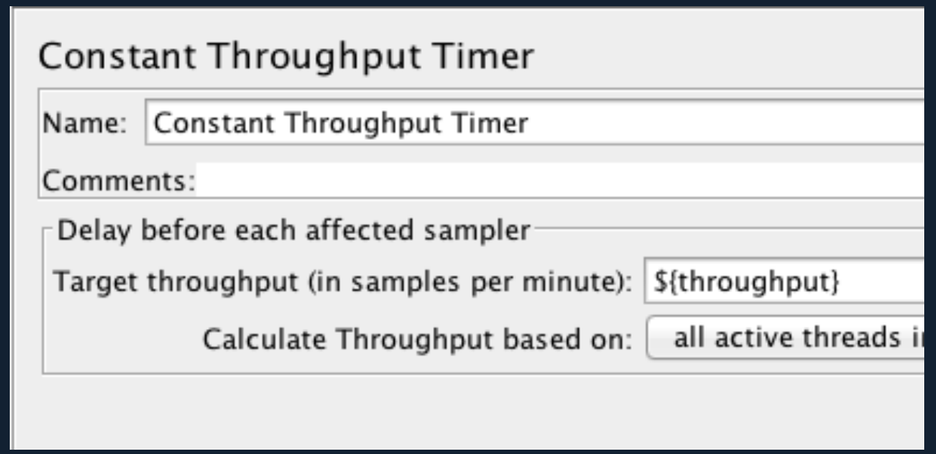
```
...
- testing RESTful services
  ...
  - variables VS properties
  ...
```

USER DEFINED VARIABLES



USER DEFINED VARIABLES





PASSING VARIABLES INTO TESTS
POM.XML

```
<!--
    User-defined variables
-->
<propertiesUser>
    <threads>${threads}</threads>
    <duration>${duration}</duration>
    <throughput>${throughput}</throughput>
</propertiesUser>
```

PASSING VARIABLES INTO TESTS

```
$ mvn verify \
-Dduration=60 \
-Dthroughput=2000 \
-Dthreads=200
etc...
```

AGENDA (EXPLAINED)

```
...
- testing RESTful services
  ...
  - custom errors
  - logging
```

PROCESSING CUSTOM ERRORS

```
if (prev.getResponseCode().equals("400") &&
    (   vars.get("error").equals("PAGE_IS_NOT_VALID")
    ||  vars.get("error").equals("LOCATION_ID_IS_NOT_VALID")
    )
) {
    prev.setResponseOK(); // It's expected error, we have to ignore it.
}
```

LOGGING

```
// logging into log file
log.info("Server returns status code 400 for the URL: "
        + prev.getUrlAsString()
        + ". Root cause: " + vars.get("error"));

// logging into console
System.out.println("Server returns status code 400 for the URL: "
        + prev.getUrlAsString()
        + ". Root cause: " + vars.get("error"));
```

ALL TOGETHER

```
if (prev.getResponseCode().equals("400") &&
    (   vars.get("error").equals("PAGE_IS_NOT_VALID")
    ||  vars.get("error").equals("LOCATION_ID_IS_NOT_VALID")
    )
) {
    log.info("Server returns status code 400 for the URL: "
            + prev.getUrlAsString()
            + ". Root cause: " + vars.get("error"));
    System.out.println("Server returns status code 400 for the URL: "
            + prev.getUrlAsString()
            + ". Root cause: " + vars.get("error"));
    prev.setResponseOK(); // It's expected error (business error), we have to ignore it.
} else if (prev.getResponseCode().equals("400") || prev.getResponseCode().equals("500")) {
    log.info("!!! Unknown server error for the URL: "
            + prev.getUrlAsString()
            + ". Root cause: " + vars.get("error"));
    System.out.println("!!! Unknown server error for the URL: "
            + prev.getUrlAsString()
            + ". Root cause: " + vars.get("error"));
}
```

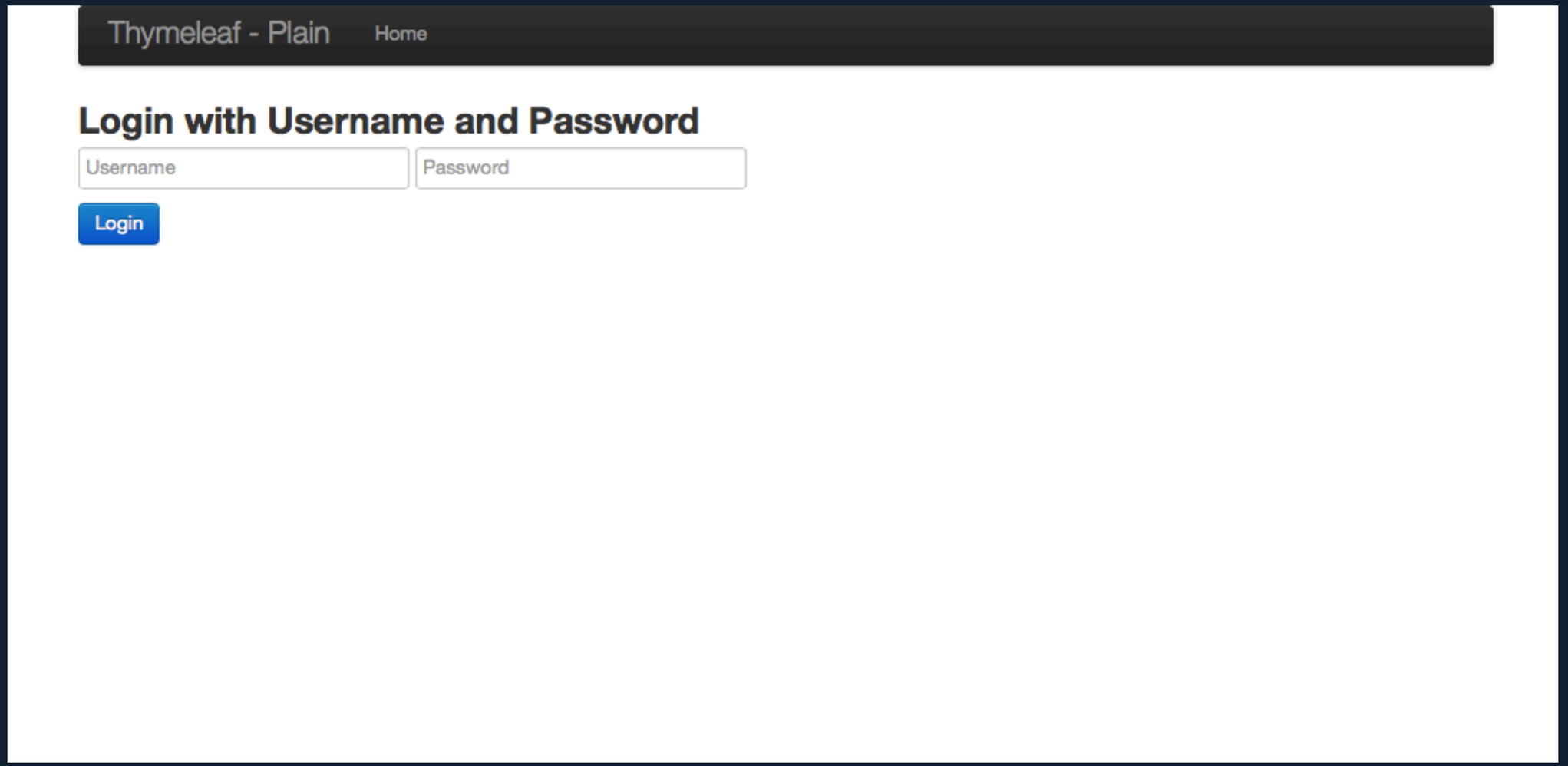
AGENDA (EXPLAINED)

```
...
- testing 'like a web browser'
 - authentication
 - session support
 - sampling
...
```

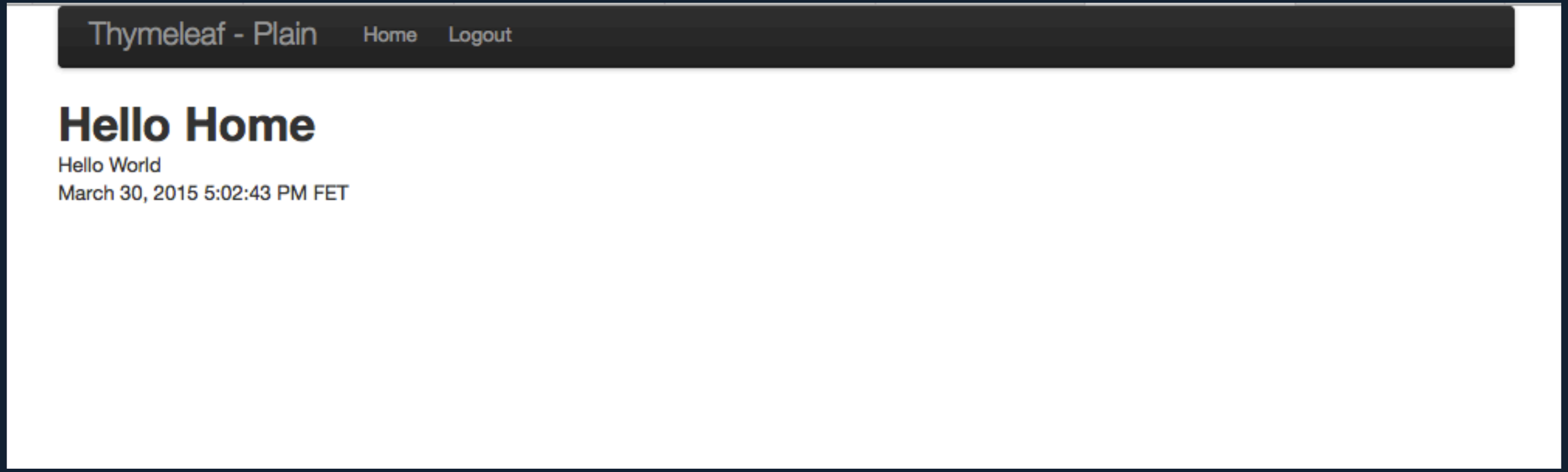
“JMeter is not a browser.”
Jmeter documentation

As far as web-services and remote services are concerned, JMeter looks like a browser (or rather, multiple browsers); however JMeter does not perform all the actions supported by browsers. In particular, JMeter does not execute the Javascript found in HTML pages. Nor does it render the HTML pages as a browser does (it's possible to view the response as HTML etc, but the timings are not included in any samples, and only one sample in one thread is ever viewed at a time).

WEB APP (LOGIN FORM)



WEB APP (HOME)

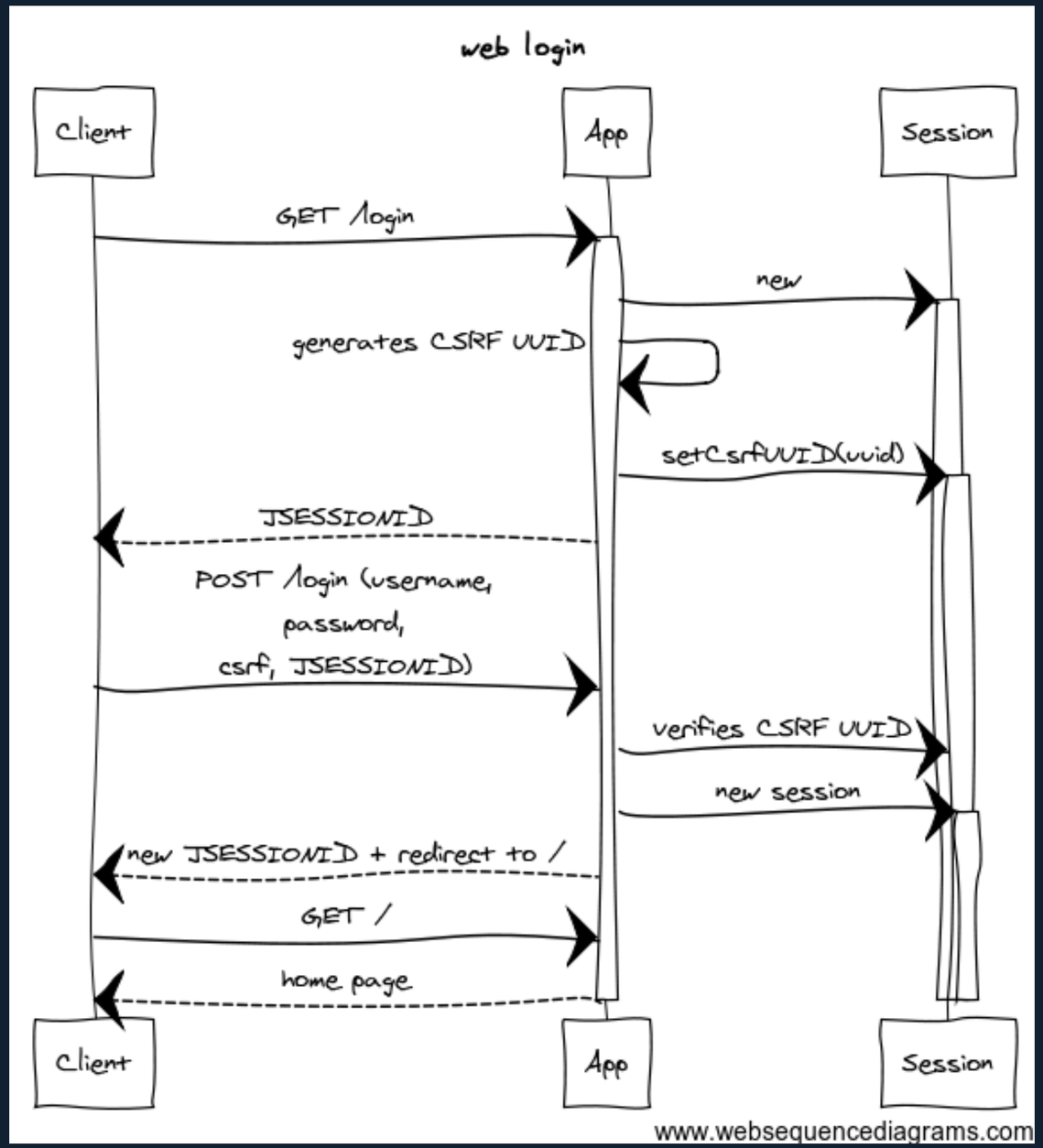


CSRF

“Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious Web site, email, blog, instant message, or program causes a user's Web browser to perform an unwanted action on a trusted site for which the user is currently authenticated.”
OWASP.ORG

SESSION FIXATION

“Session Fixation is an attack that permits an attacker to hijack a valid user session. The attack explores a limitation in the way the web application manages the session ID, more specifically the vulnerable web application.”
OWASP.ORG



GETTING WEB APP LOGIN FORM

```
$ curl localhost:9090/login -v
```

RESPONSE

```
> Set-Cookie: JSESSIONID=22EF4454BF1268649B200A70A60F13B3; Path=/; HttpOnly

<form name="form" action="/login" method="POST">
    <fieldset>
        <input type="text" name="username" value="" placeholder="Username" />
        <input type="password" name="password" placeholder="Password" />
    </fieldset>
    <input type="submit" id="login" value="Login" class="btn btn-primary" />
    <input type="hidden" name="_csrf" value="2773bb75-a090-4a6a-8ad3-966c94fb7092" />
</form>
```

LOGIN

```
$ curl -v -X POST localhost:9090/login \
-d 'username=user&password=user&_csrf=69035629-afd6-4a49-8396-c3034704ec73' \
-H 'Cookie: JSESSIONID=22EF4454BF1268649B200A70A60F13B3'
```

RESPONSE

```
> Set-Cookie: JSESSIONID=C9778DFFBD1A6CD5DDF3BC77B86CA87B; Path=/; HttpOnly
> Location: http://localhost:9090/
```

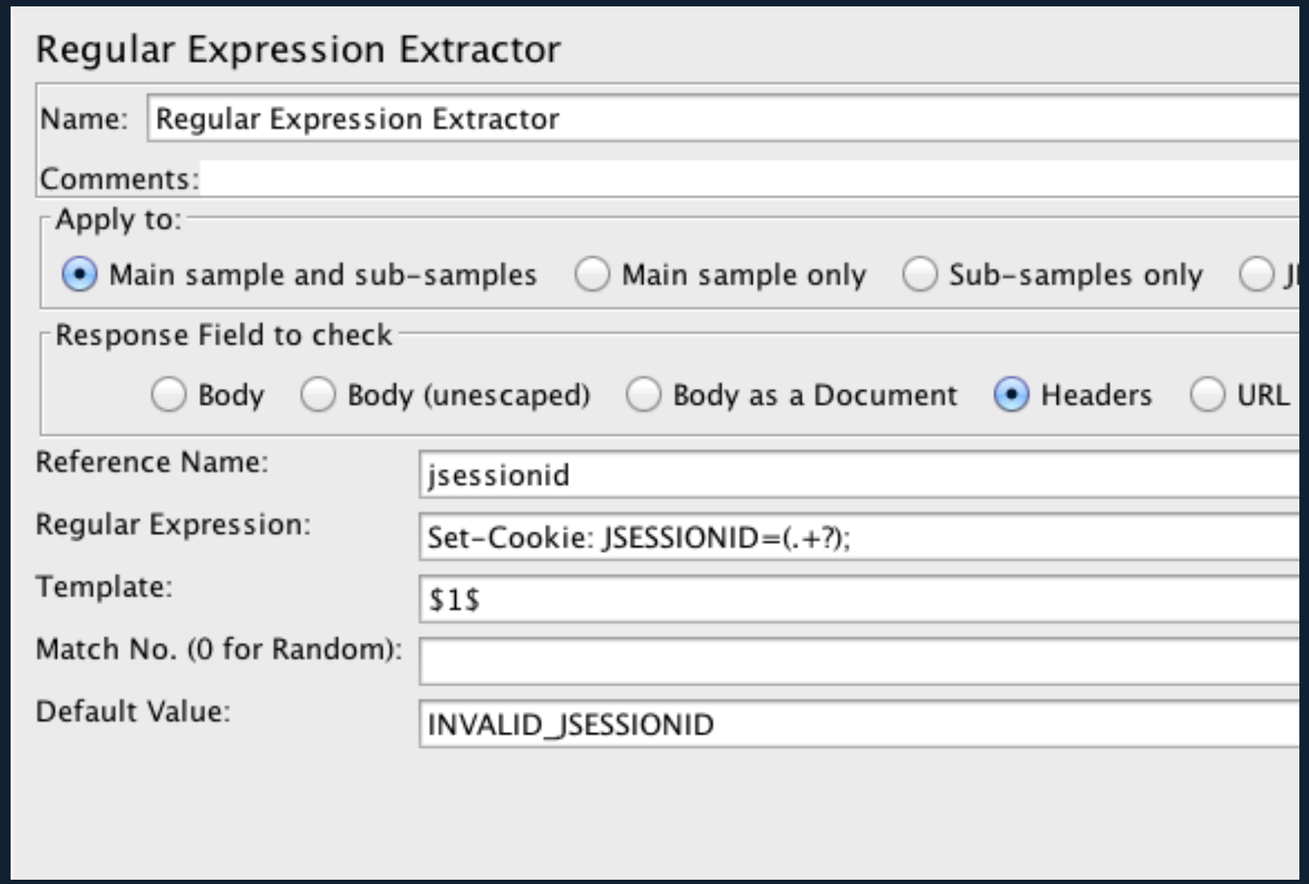
CHECKING SECURED PAGE

```
$ curl -v localhost:9090 -H 'Cookie: JSESSIONID=C9778DFFBD1A6CD5DDF3BC77B86CA87B'
```

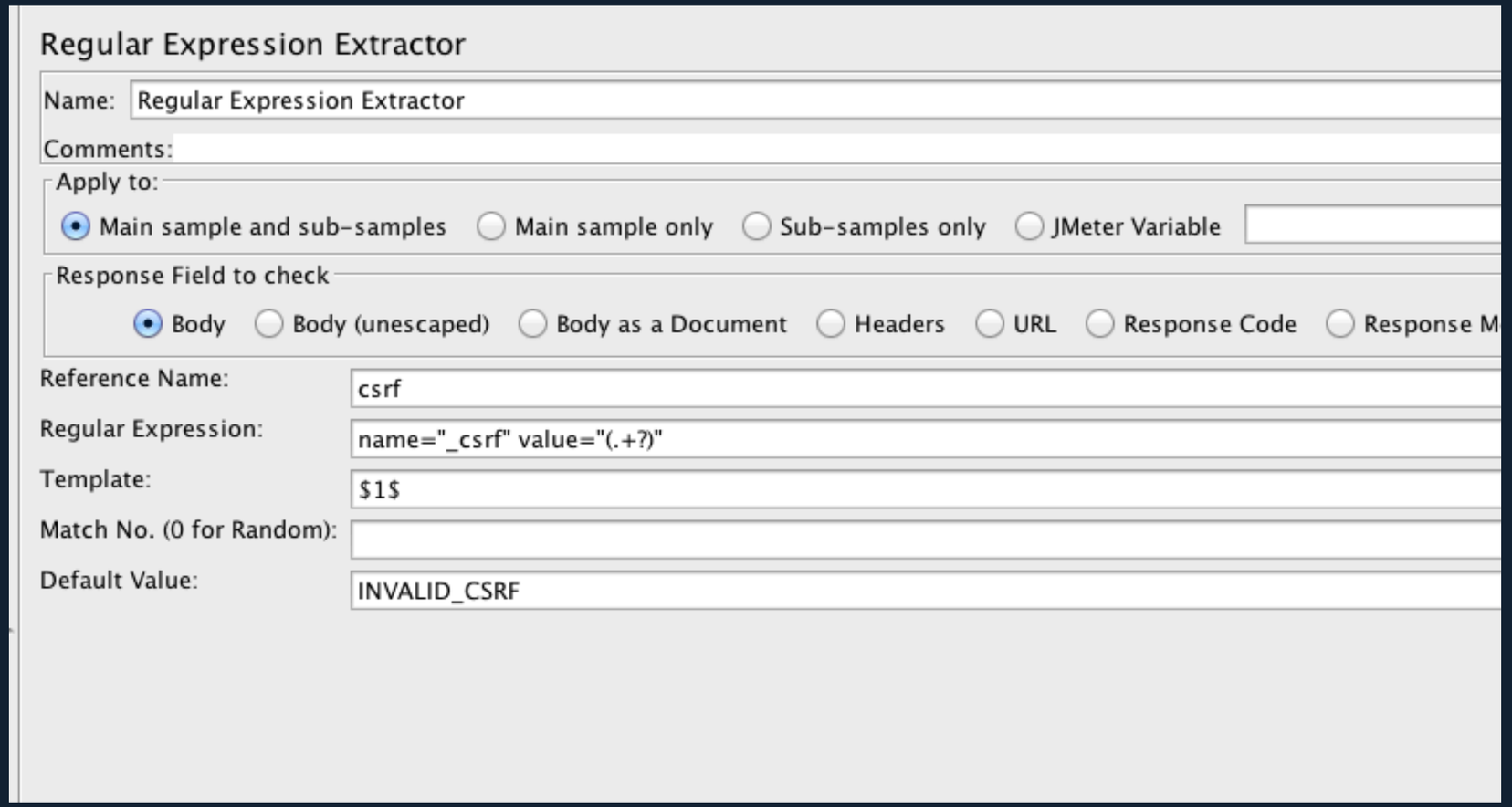
RESPONSE

```
<h1>Hello Home</h1>
<div>Hello World</div>
```

JMETER
EXTRACTING JSESSIONID



JMETER
EXTRACTING CSRF TOKEN



FINALLY, ALL TESTS[1]

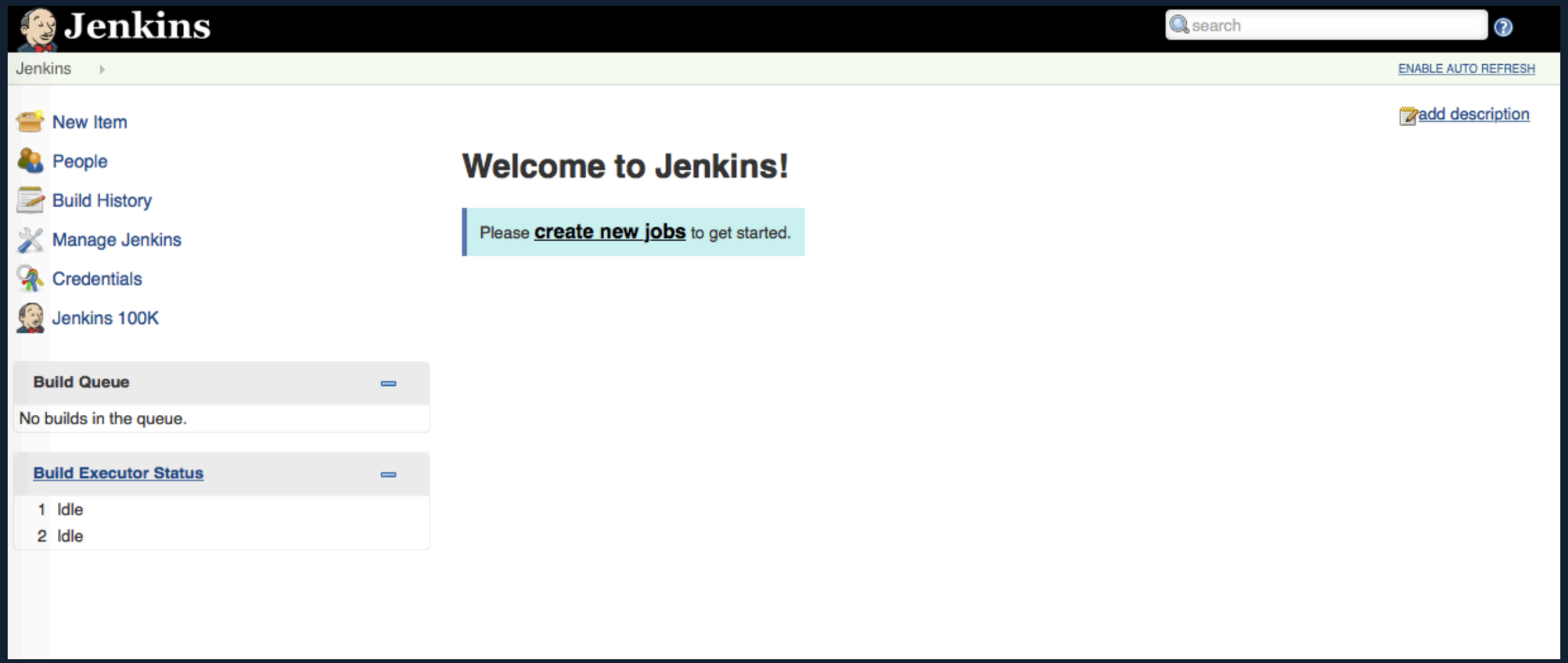
```
tests$ mvn jmeter:verify
```

it will execute all performance tests

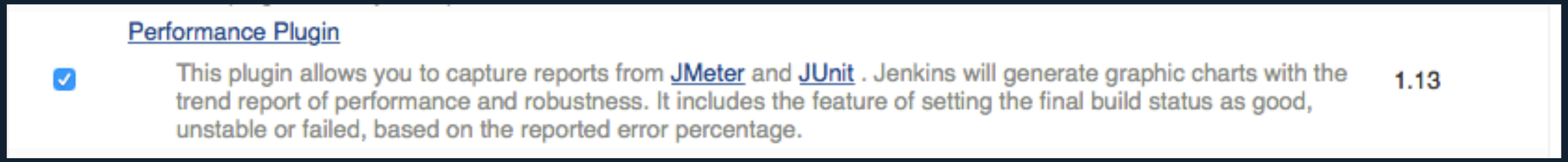
[1] showtime

JENKINS START

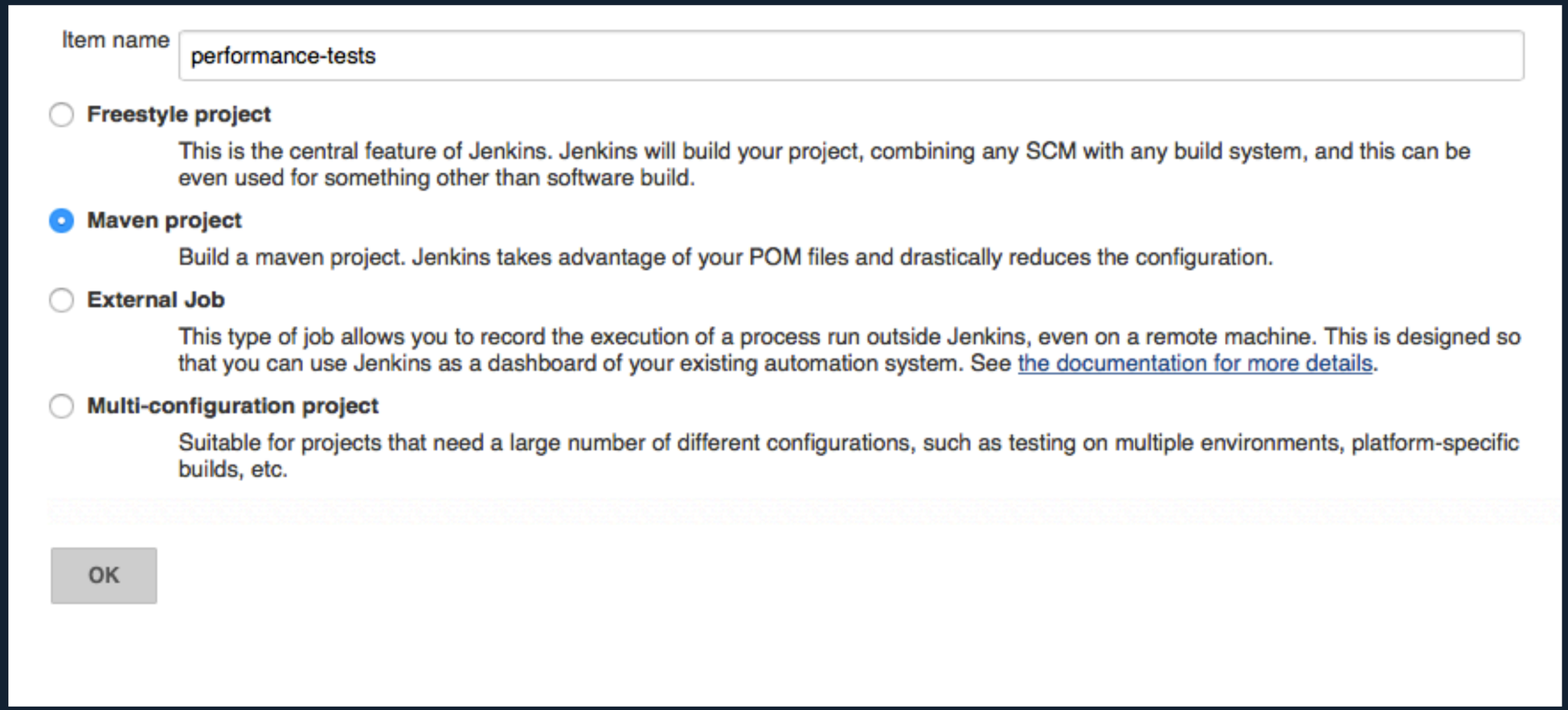
```
tomcat$ ./bin/catalina.sh start
```



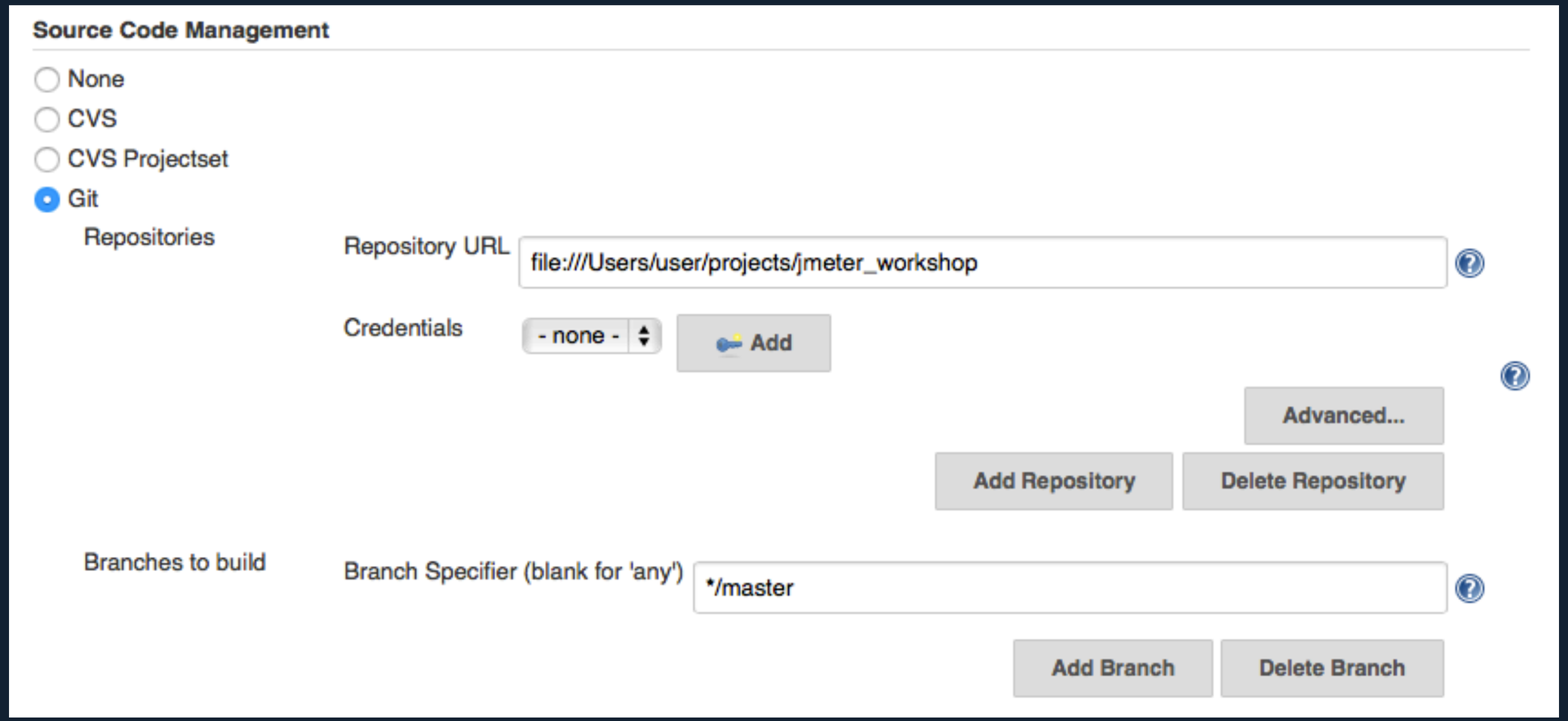
CONFIGURING JENKINS
PERFORMANCE PLUGIN



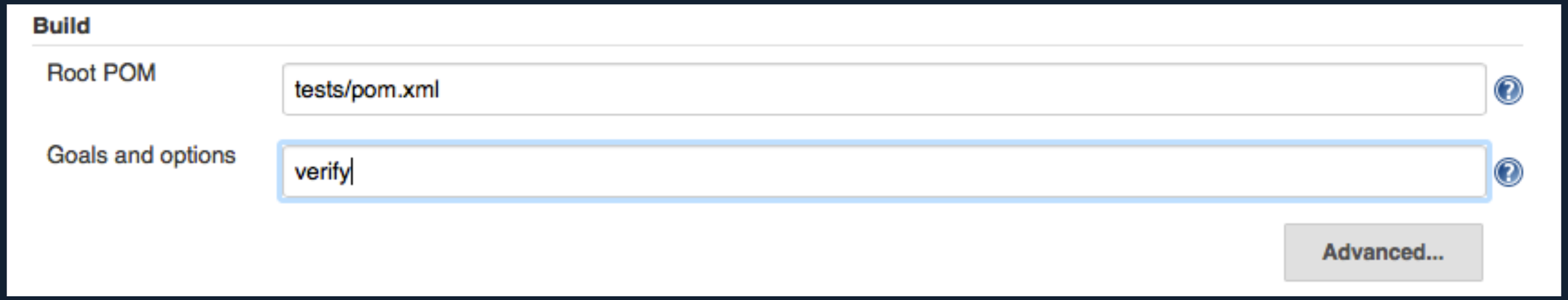
CONFIGURING JENKINS
NEW JOB



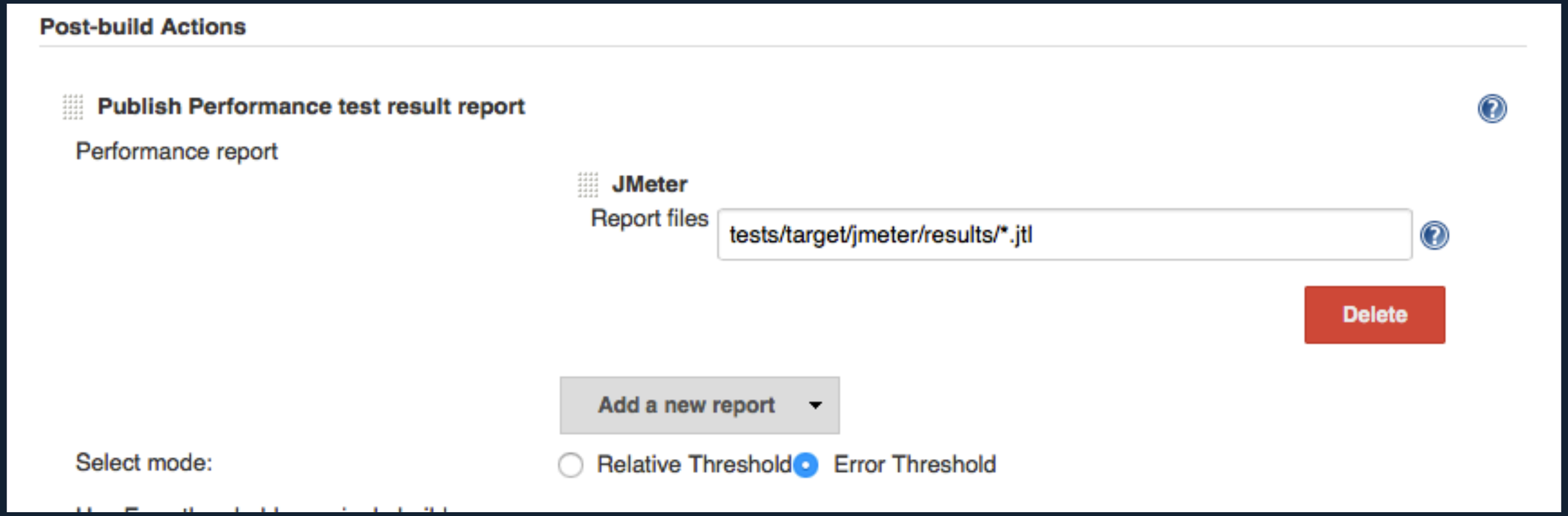
CONFIGURING JENKINS
REPOSITORY



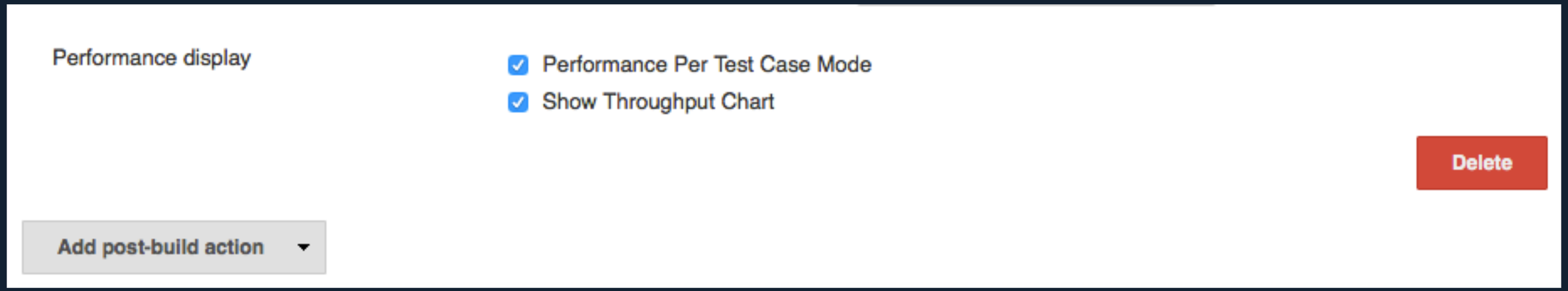
CONFIGURING JENKINS
BUILD



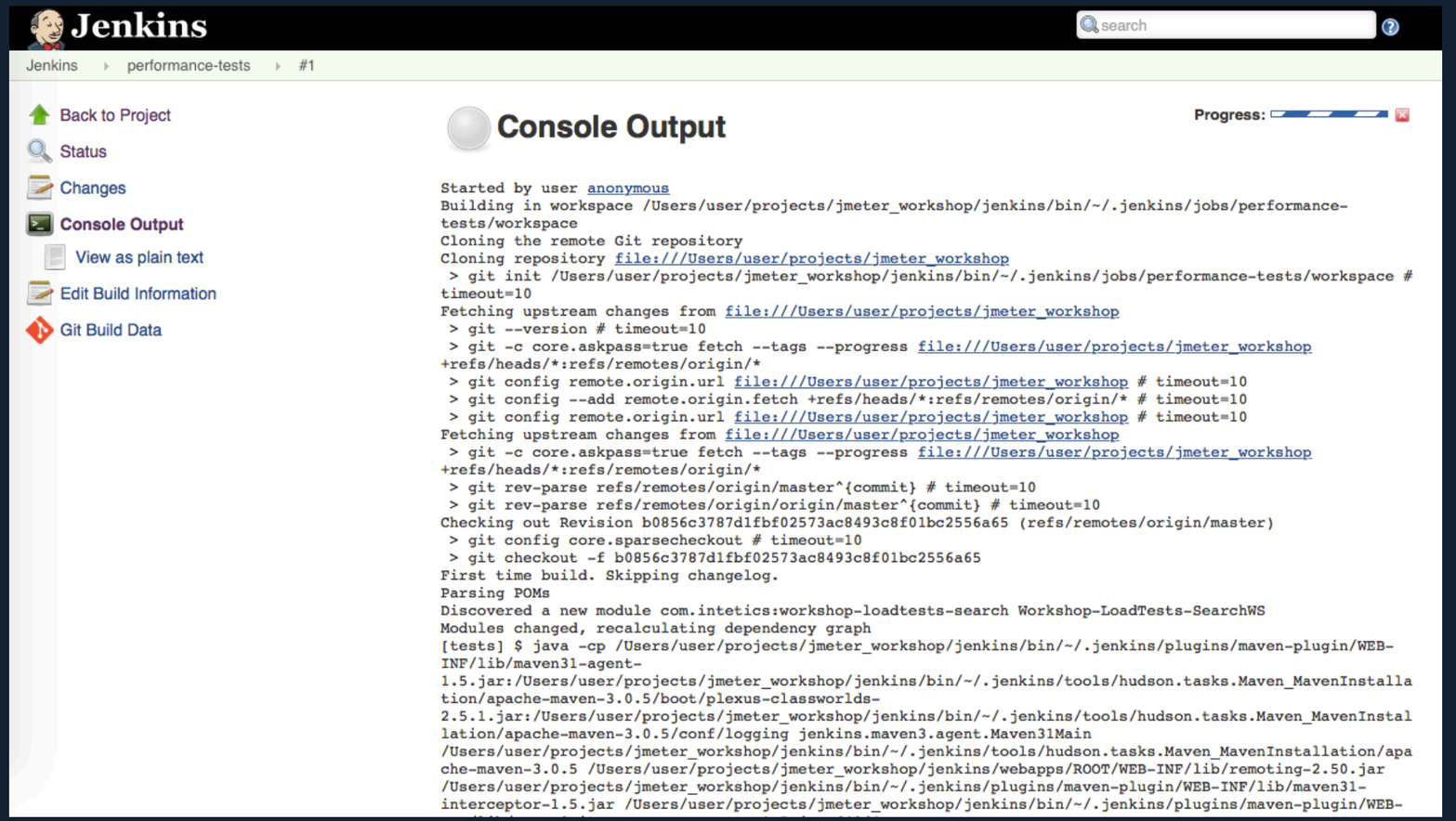
CONFIGURING JENKINS
PUBLISH REPORT



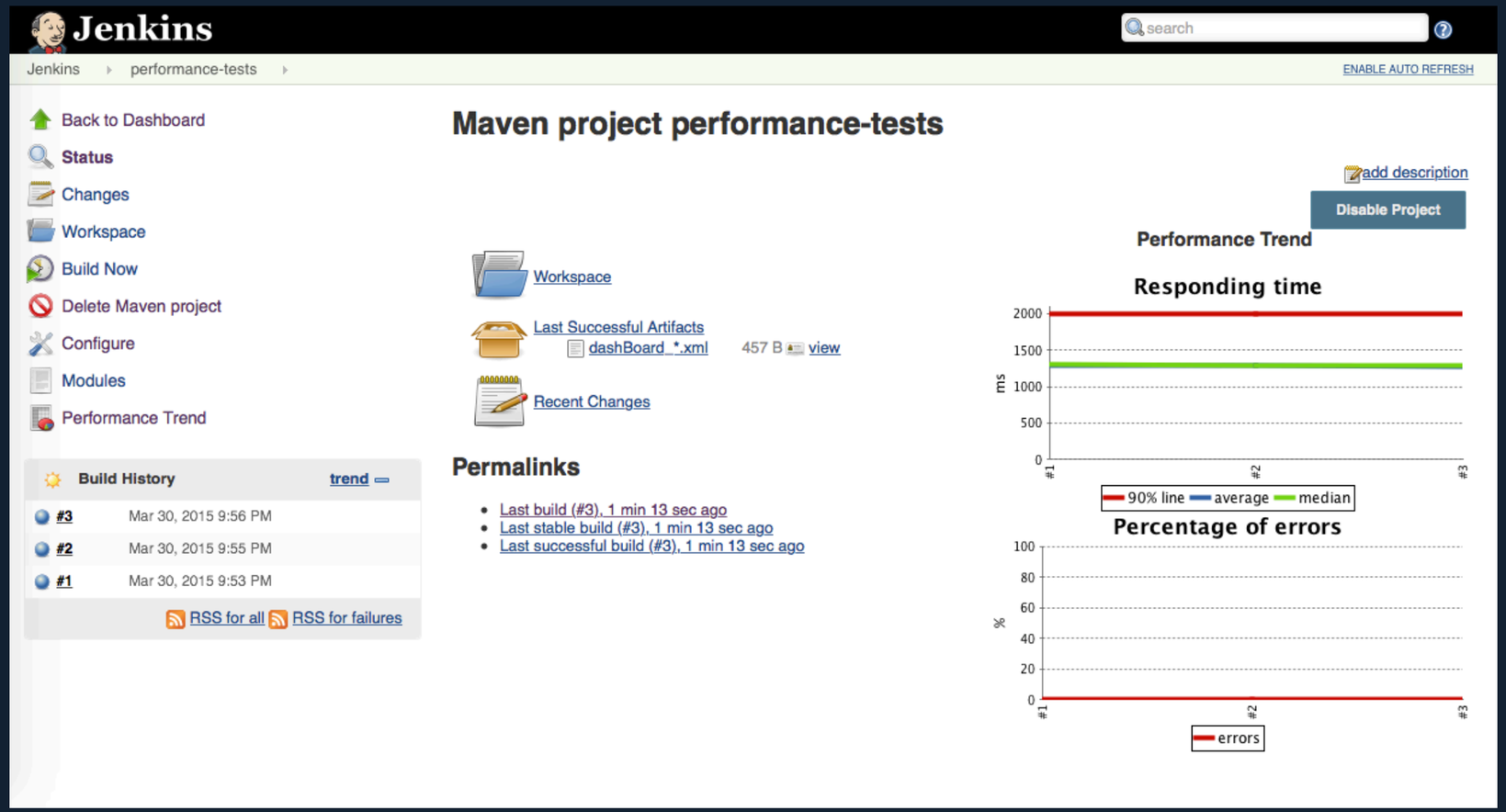
CONFIGURING JENKINS
PUBLISH REPORT



JENKINS
RUN ALL PERFORMANCE TESTS

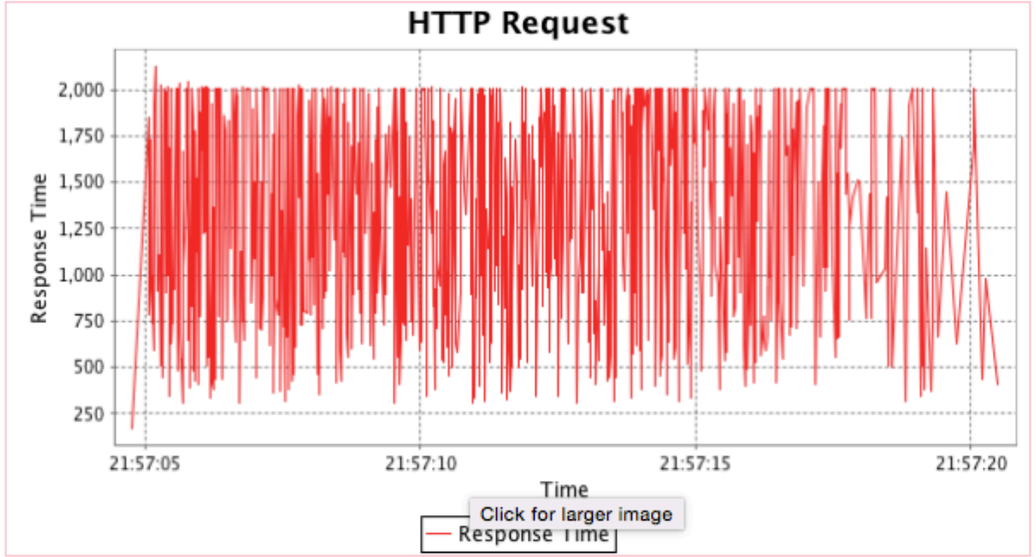


JENKINS REPORTING



JENKINS
REPORTING

URI: HTTP Request



URI: HTTP Request

URI	Samples	Samples diff	Average (ms)	Average diff (ms)	Median (ms)	Median diff (ms)	Line90 (ms)	Minimum (ms)	Maximum (ms)
	1001	0	1275	0	1286	0	2007	172	2123

LINKS

» Automating JMeter tests with Maven and Jenkins

» JMeter Maven Plugin

» JMeter Maven Plugin online documentation

» About Constant Throughput Timer on Stackoverflow

» How to run a test plan for a certain amount of time (very strange approach)

» JMeter functions (${__P(...)})

LINKS

» Passing global parameters to a jmx file

» How to use a CSV file with JMeter

» Difference between sharing modes “All threads” and “Current thread group” in Jmeter

» How to use BeanShell

» JMeter how to NOT fail 500 Internal Server Errors

» Howto debug JMeter scripts