

# 准备工作

---

- Download Gatling from <http://gatling.io/#/resources/download>
  - Unzip the downloaded bundle to a folder of your choice
- Github: <https://github.com/twqa/GatlingWorkShop>



# ThoughtWorks®

*QA Community Workshop*

---

# GATLING

---

*Qianqian Wang & Juewen Zhang & Chen Li*

# 性能测试的那些事儿

---

## 性能测试 VS 功能测试

## 前端性能测试 VS 后端性能测试

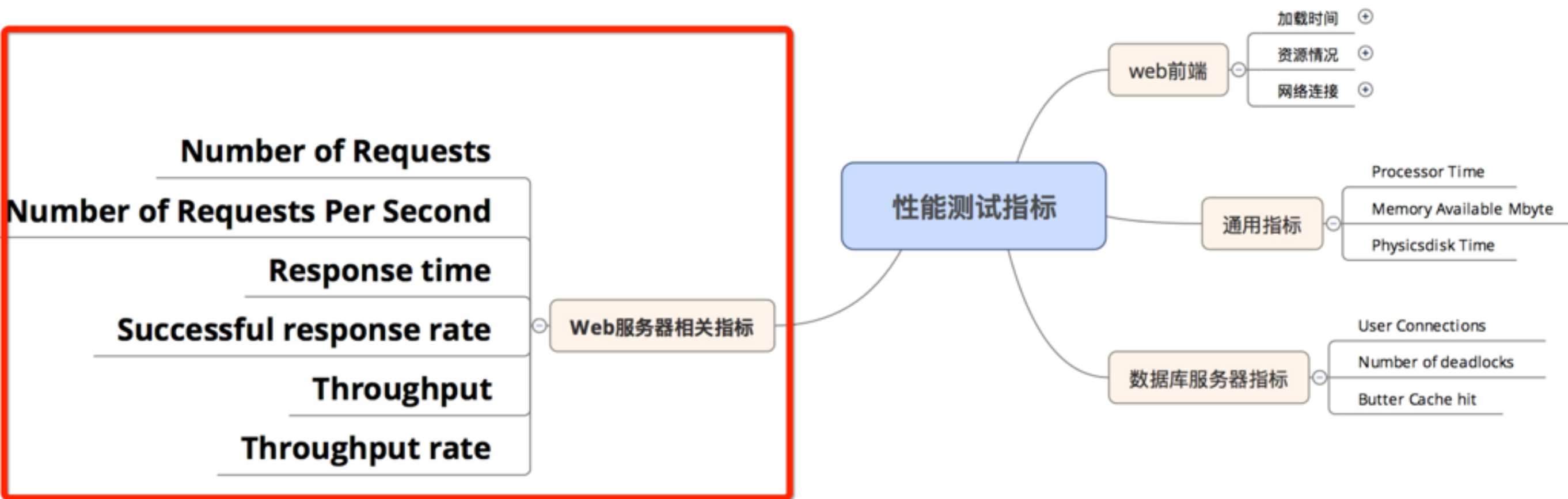


# 性能测试流程

---

1. 项目背景 & 验收条件
2. 测试场景制定
3. 测试环境 & 数据准备
4. 测试执行
5. 报告分析

# 性能测试的指标



# 性能测试工具选择

---

- 性能需求
- 购买成本
- 学习成本

LoadRunner

Jmeter

Gatling



# GATLING特点

---

- 支持DSL脚本，易于开发和维护，学习成本低
- 支持Akka Actors 和 Async IO，从而能达到很高的性能
- Session支持，也就是第一个请求的响应，能用于后续请求的参数。
- 支持实时生成Html动态轻量报表，从而使报表更易阅读和进行数据分析
- 支持录制和导入HAR（Http Archive）并生成测试脚本
- 支持Maven，Eclipse，IntelliJ等，以便于开发
- 支持Jenkins，以便于进行持续集成
- 开源免费，支持拓展

# GATLING 适用场景

---

- 测试需求经常改变，测试脚本需要经常维护；
- 测试环境的客户机性能不强，但又希望发挥硬件的极限性能；
- 能对测试脚本进行很好的版本管理，并通过CI进行持续的性能测试；
- 希望测试结果轻量易读；



# GATLING 关键词

---












- ❑ Virtual User
- ❑ Scenario
- ❑ Simulation
- ❑ Session
- ❑ Feeders
- ❑ Checks
- ❑ Assertion
- ❑ Reports



Checks vs Assertion

# GATLING 目录结构

---

 bin		gatling.bat
 conf		gatling.sh
 lib		recorder.bat
 LICENSE		recorder.sh
 results		
 target		
 user-files		

## 录制场景

---

1. 进入demo网站 <http://computer-database.gatling.io>
2. 搜索macbook
3. 打开搜索结果的第一条记录
4. 编辑这台电脑的任意信息

# GATLING RECORDER配置

- 启动Gatling Recorder - 运行\$GATLING\_HOME/bin/recorder.sh

**配置本地监听端口**

Recorder mode: HTTP Proxy

Network

Listening port\*: localhost HTTP/HTTPS 8888 HTTPS mode: Self-signed Certificate

Outgoing proxy: host: HTTP HTTPS Username Password

Simulation Information

Package: workshop Class Name\*: DemoScenario

☒ Follow Redirects? ☒ Infer html resources? ☒ Automatic Referers?

☒ Remove cache headers? ☐ Save & check response bodies?

Output

Output folder\*: /Users/jwezhang/Work/tools/gatling-charts-highcharts-bundle-2.2.2/user-files/simulations Browse

Encoding: Unicode (UTF-8)

Filters

Java regular expressions that matches the entire URI Strategy: Blacklist First

Whitelist

Blacklist

- \*.css
- \*.js
- \*.ico

+ - Clear + - Clear No static resources

Save preferences Start !

Gatling支持的3种  
HTTPS认证方式  
Self-signed certificate  
Provided KeyStore  
Certificate Authority

过滤掉不需要录制的请  
求，比如css文件等

# 监听端口配置 - FIREFOX

- Firefox > Preference > Advanced > Network > Connection (Settings...)

Configure Proxies to Access the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration:

HTTP Proxy:  Port:

☒ Use this proxy server for all protocols

SSL Proxy:  Port:

FTP Proxy:  Port:

SOCKS Host:  Port:

☐ SOCKS v4 ☒ SOCKS v5 ☐ Remote DNS

No Proxy for:

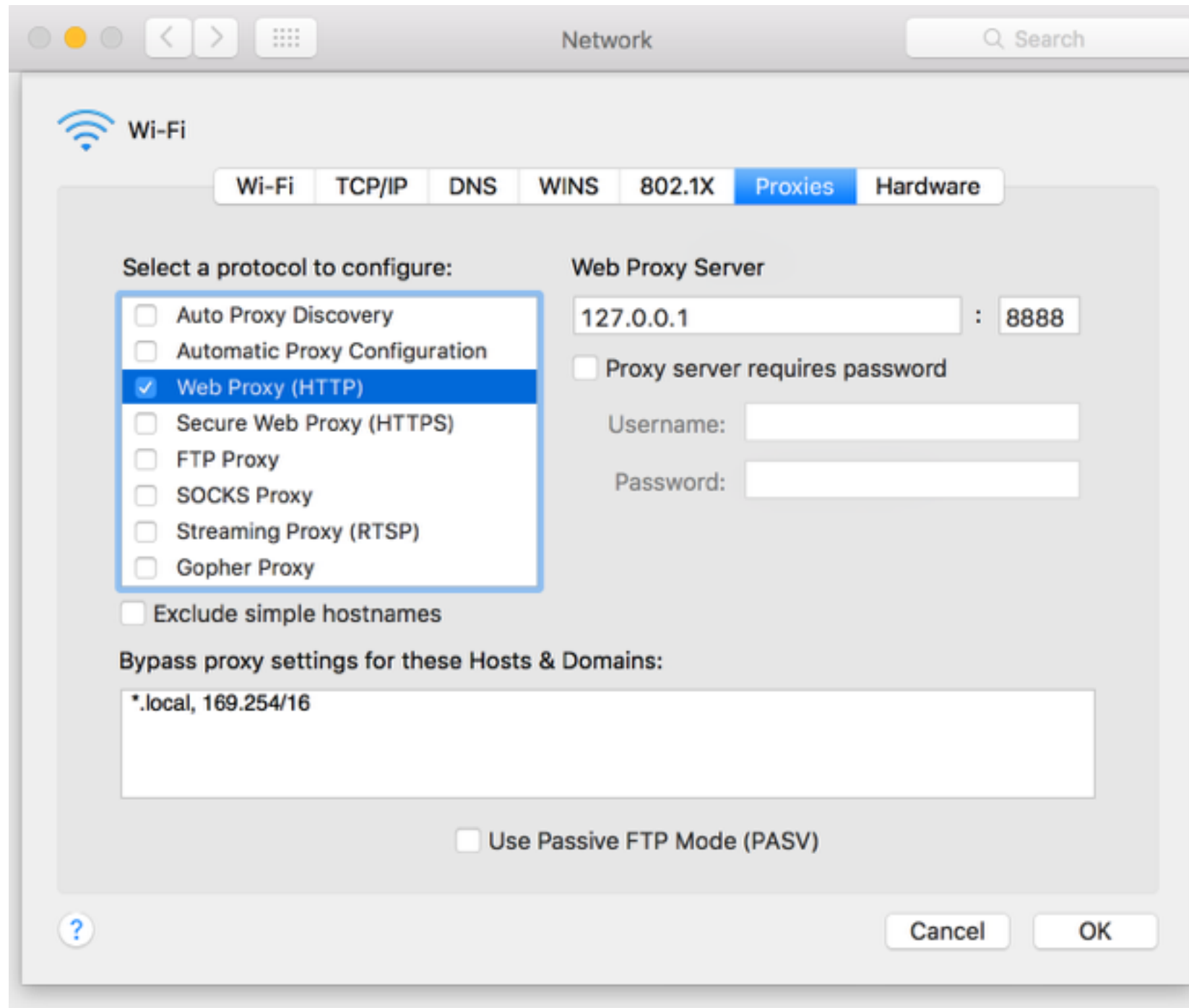
Example: .mozilla.org, .net.nz, 192.168.1.0/24

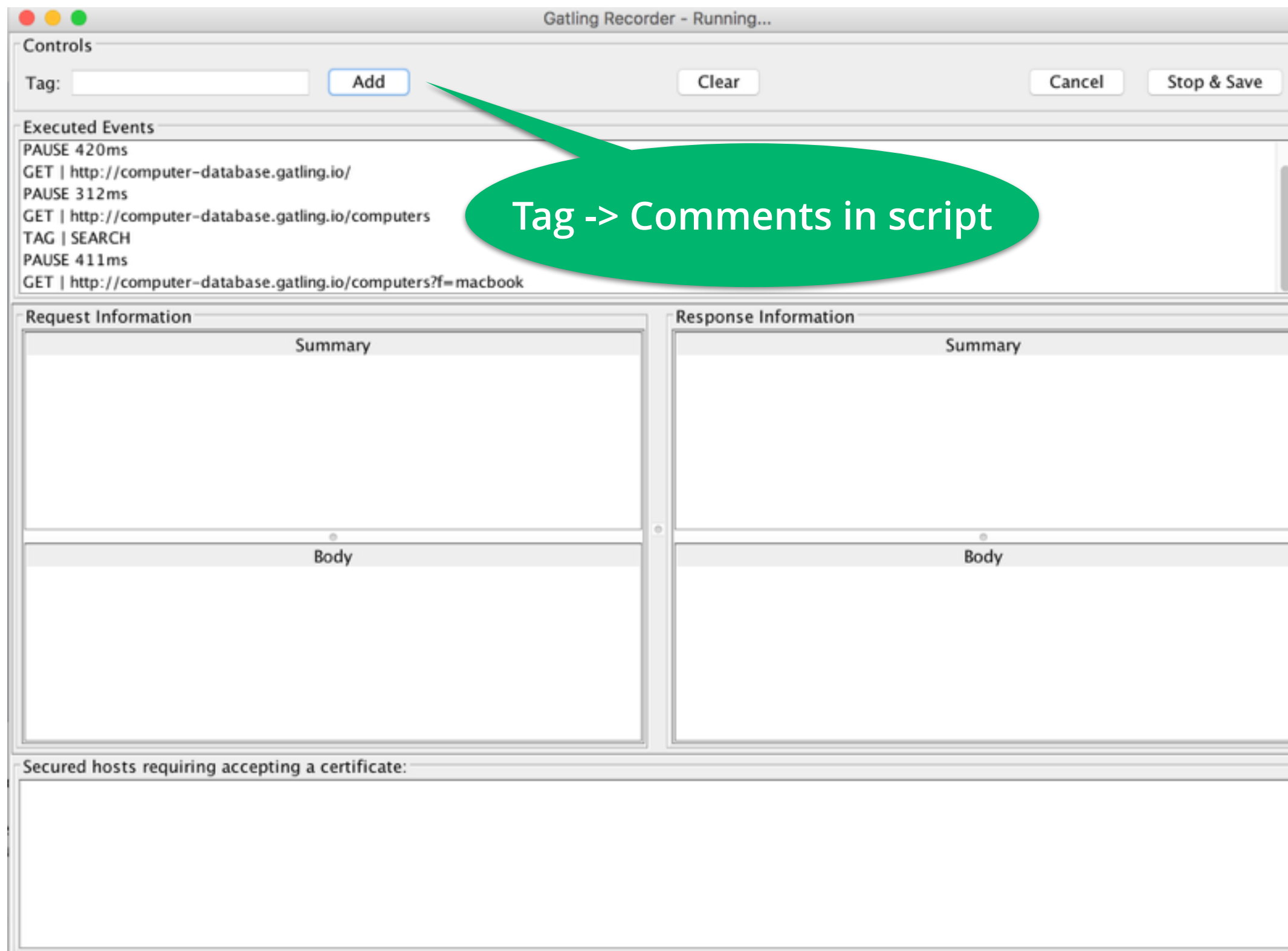
☐ Automatic proxy configuration URL:

☐ Do not prompt for authentication if password is saved

# 监听端口配置 - CHROME

- Chrome > Preference > Settings > Network (Change proxy settings...)





# SIMULATION 结构

```
package workshop

import scala.concurrent.duration._

import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.jdbc.Predef._

class DemoSimulation extends Simulation {

  val httpProtocol = http
    .baseUrl("http://computer-database.gatling.io")
    .inferHtmlResources(BlackList(""".*\.css""", """.*\.js""", """.*\.ico"""), WhiteList())
    .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
    .acceptEncodingHeader("gzip, deflate")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .userAgentHeader("Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:47.0) Gecko/20100101 Firefox/47.0")

  val scn = scenario("RecordedSimulation1")
    // OPEN
    .exec(http("request_0")
      .get("/"))
    .pause(2)
    // SEARCH
    .exec(http("request_1")
      .get("/computers?f=macbook"))
    .pause(2)
    // VIEW
    .exec(http("request_2")
      .get("/computers/89"))
    .pause(5)
    // EDIT
    .exec(http("request_3")
      .post("/computers/89")
      .formParam("name", "MacBook")
      .formParam("introduced", "2006-05-16")
      .formParam("discontinued", "")
      .formParam("company", "1"))

  setUp(scn.inject(atOnceUsers(1)).protocols(httpProtocol))
}
```

HTTP协议配置

场景定义

Simulation Setup



# SIMULATION 结构 - HTTP协议配置

被测系统的base URL

配置BlackList, WhiteList

```
val httpProtocol = http
    .baseUrl("http://computer-database.gatling.io")
    .inferHtmlResources(BlackList(""".*\.css""", """.*\.js""", """.*\.ico"""), WhiteList())
    .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
    .acceptEncodingHeader("gzip, deflate")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .userAgentHeader("Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:47.0) Gecko/20100101 Firefox/47.0")
```

通用的Headers

# SIMULATION 结构 - 场景定义

```
val scn = scenario("RecordedSimulation1")
```

定义Scenario

```
// OPEN  
.exec(http("OPEN")  
  .get("/"))  
.pause(2)
```

```
// SEARCH
```

```
.exec(http("SEARCH")  
  .get("/computers?f=macbook"))  
.pause(2)
```

.exec方法：执行一个动作，通常是发送一个请求 - get, post....

```
// VIEW
```

```
.exec(http("VIEW")  
  .get("/computers/89"))  
.pause(5)
```

.pause方法：暂停一段时间

```
// EDIT
```

```
.exec(http("EDIT")  
  .post("/computers/89")  
  .formParam("name", "MacBook")  
  .formParam("introduced", "2006-05-16")  
  .formParam("discontinued", "")  
  .formParam("company", "1"))
```

# SIMULATION 结构 - 模拟场景构建

```
setUp(scen.inject(atOnceUsers(1))).protocols(httpProtocol)
```

注入定义

协议定义

## ■ 常用注入方法

<code>nothingFor(duration)</code>	暂停一段时间
<code>atOnceUsers(N)</code>	一次注入的虚拟用户
<code>rampUsers(N) over (seconds)</code>	在指定时间内，设置一定数量逐步注入的虚拟用户；
<code>constantUsersPerSec(rate) during(duration)</code>	定义一个在每秒钟恒定的并发用户数，持续指定的时间；
<code>constantUsersPerSec(rate) during(duration) randomized</code>	定义一个在每秒钟围绕指定并发数随机增减的并发，持续指定时间；
<code>rampUsersPerSec(rate1) to (rate2) during(duration)</code>	定义一个并发数区间，运行指定时间，并发增长的周期是一个规律的值；
<code>rampUsersPerSec(rate1) to(rate2) during(duration) randomized</code>	定义一个并发数区间，运行指定时间，并发增长的周期是一个随机的值；
<code>heavisideUsers(nbUsers) over(duration)</code>	定义一个持续的并发，围绕和海维赛德函数平滑逼近的增长量，持续指定时间（译者解释下海维赛德函数，
<code>splitUsers(nbUsers) into(injectionStep) separatedBy(duration)</code>	定义一个周期，执行injectionStep里面的注入，将nbUsers的请求平均分配；
<code>splitUsers(nbUsers) into(injectionStep1) separatedBy(injectionStep2)</code>	使用injectionStep2的注入作为周期，分隔injectionStep1的注入，直到用户数达到nbUsers；

# INJECT 例子

## • .inject( )方法

```
setUp(  
    scn.inject(  
        nothingFor(4 seconds),  
        atOnceUsers(10),  
        rampUsers(10) over(5 seconds),  
        constantUsersPerSec(20) during(15 seconds),  
        constantUsersPerSec(20) during(15 seconds) randomized,  
        rampUsersPerSec(10) to 20 during(10 minutes),  
        rampUsersPerSec(10) to 20 during(10 minutes) randomized,  
        splitUsers(1000) into(rampUsers(10) over(10 seconds)) separatedBy(10 seconds),  
        splitUsers(1000) into(rampUsers(10) over(10 seconds)) separatedBy atOnceUsers(30),  
        heavisideUsers(1000) over(20 seconds)  
    ).protocols(httpConf)  
)
```

暂停4秒

一次注入10个用户

在5秒内，逐步注入10个用户

每秒注入20个用户，持续15秒

# SIMULATION 结构 - HEADERS 定义

---

- 通用的请求在httpProtocol里定义
- 单个请求需要的header用一下方式定义
  - `val headers_01 = Map("Content-Type" -> ""application/x-www-form-urlencoded"")`
  - `.headers( )`方法
    - `http("request_01")`
    - `.post("/computers")`
    - `.headers(headers_01)`

# SIMULATION 结构 - HOOKS

---

- Gatling提供两种hooks:

before: 在simulation script前执行

after: 在simulation script后执行

```
before {  
    println("Simulation is about to start!")  
}  
  
after {  
    println("Simulation is finished!")  
}
```



# 运行

- Run录制的Script - 运行\$GATLING\_HOME/bin/gatling.sh

```
→ bin ./gatling.sh
GATLING_HOME is set to /Users/jwezhang/Work/tools/gatling-charts-highcharts-bundle-2.2.2
Choose a simulation number:
  [0] RecordedSimulation
  [1] computerdatabase.BasicSimulation
  [2] computerdatabase.advanced.AdvancedSimulationStep01
  [3] computerdatabase.advanced.AdvancedSimulationStep02
  [4] computerdatabase.advanced.AdvancedSimulationStep03
  [5] computerdatabase.advanced.AdvancedSimulationStep04
  [6] computerdatabase.advanced.AdvancedSimulationStep05
  [7] gatlingdemo.Scenario1
  [8] github.GitHubScenario
  [9] workshop.DemoScenario
 [10] workshop.DemoScenario1
 [11] workshop.RecordedSimulation
```

- 选择需要执行的Scenario

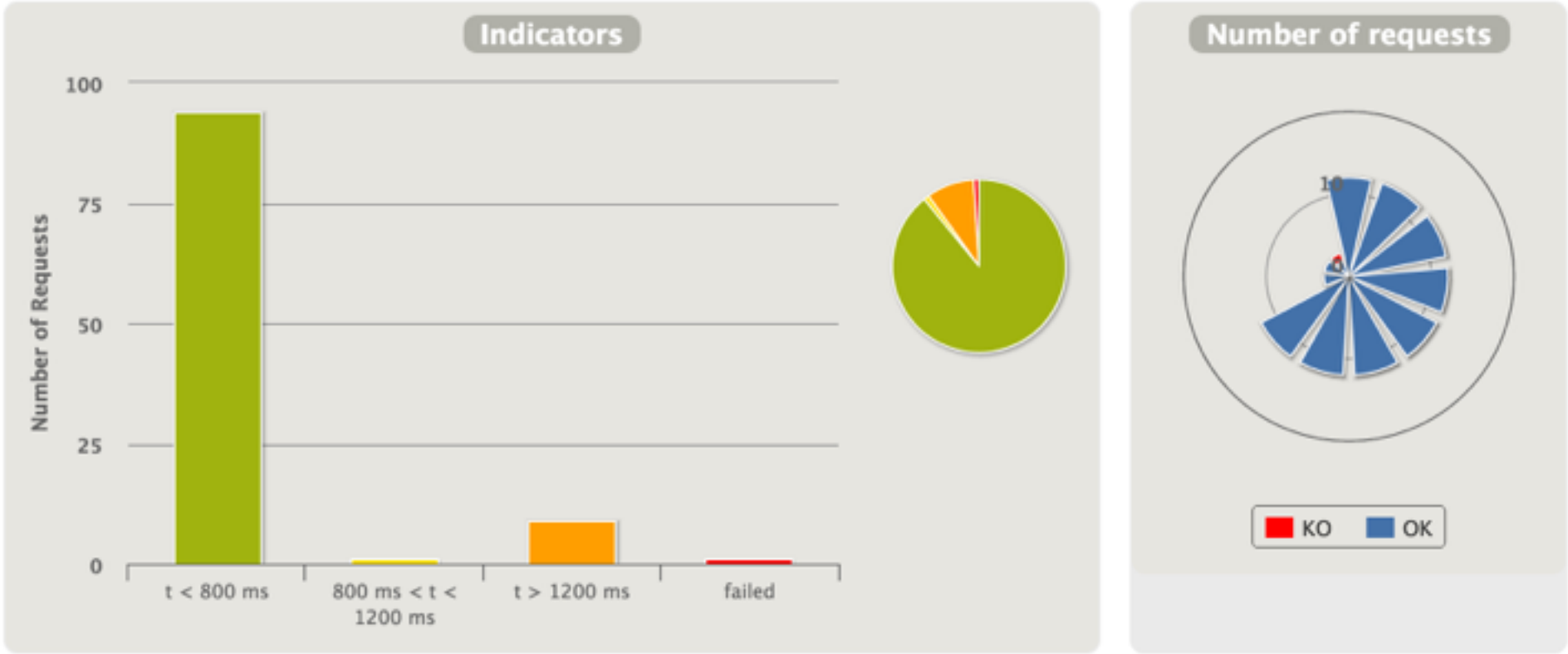
Tips - 参数 <http://gatling.io/docs/2.2.2/http/recorder.html>

- -s <className> or -simulation <className>, 指定运行某个Scenario  
e.g. \$GATLING\_HOME/bin/gatling.sh -s workshop.DemoScenario



- ❑ Indicator
- ❑ Active users over time
- ❑ Response time distribution
- ❑ Response time percentiles over time
- ❑ Requests per second over time
- ❑ Responses per second over time
- ❑ Request/group specific charts

> Global Information



STATISTICS

Expand all groups | Collapse all groups

Requests ^	Executions				Response Time (ms)								
	Total ↕	OK ↕	KO ↕	% KO ↕	Req/s ↕	Min ↕	50th pct ↕	75th pct ↕	95th pct ↕	99th pct ↕	Max ↕	Mean ↕	Std Dev ↕
Global Information	105	104	1	1%	3.889	597	622	657	1950	2255	2295	754	405
Home	12	12	0	0%	0.444	610	618	623	644	645	646	622	10
Home Redirect 1	12	12	0	0%	0.444	604	627	657	664	666	667	633	23
Search	12	12	0	0%	0.444	597	620	676	2035	2207	2250	864	539
Select	12	12	0	0%	0.444	604	635	659	1531	2142	2295	790	460
Page 0	12	12	0	0%	0.444	604	620	650	665	667	668	630	21
Page 1	12	12	0	0%	0.444	606	637	662	1332	1984	2147	762	418
Page 2	12	12	0	0%	0.444	600	625	647	1190	1699	1827	726	332
Page 3	12	12	0	0%	0.444	606	631	906	2255	2255	2256	983	632
Form	3	3	0	0%	0.111	607	621	1297	1837	1945	1973	1067	640
Post	3	3	0	0%	0.111	615	650	701	741	749	752	672	58
Post Redirect 1	3	2	1	33%	0.111	608	622	634	644	646	647	625	16





# PRACTICE

<https://github.com/twqa/GatlingWorkShop>



# 常用性能测试工具对比

Feature	The Grinder	Gatling	Tsung	JMeter
OS	Any	Any	Linux/Unix	Any
GUI	Console Only	Recorder Only	No	Full
Test Recorder	TCP (including HTTP)	HTTP	HTTP, Postgres	HTTP
Test Language	Python, Clojure	Scala	XML	XML
Extension Language	Python, Clojure	Scala	Erlang	Java, Beanshell, Javascript, Jexl
Load Reports	Console	HTML	HTML	CSV, XML, Embedded Tables, Graphs, Plugins
Protocols	JM	HTTP JDBC JMS	AMQP MQTT LDAP	JDBC, SOAP, LDAP, TCP, JMS, SMTP, POP3, IMAP
Host monitoring	No	No	Yes	Yes with PerfMon plugin
Limitations	Reports are very plain and brief	Does not scal	Tested and supported only on Linux systems.	Bundled reporting isn't easy to interpret

- CI Integration - <https://wiki.jenkins-ci.org/display/JENKINS/Gatling+Plugin>
- gatling-sbt
  - A simple project showing how to configure and use Gatling's SBT plugin to run Gatling simulations
  - [http://gatling.io/docs/2.2.2/extensions/sbt\\_plugin.html](http://gatling.io/docs/2.2.2/extensions/sbt_plugin.html)
- HTTPS
  - 添加gatling自己的证书
  - openssl 自己创建CA
- Login Token问题

## 参考资料

---

- 性能测试 - 虫师的博客 <http://www.cnblogs.com/fnng/archive/2012/08/17/2644878.html>
- Gatling入门 - Gatling官网 <http://gatling.io/>
- 实践相关 - Testerhome.com, gatling google group

# THANK YOU

---

*For questions or suggestions:*

*<https://jinshuju.net/f/vszO2e>*



---

ThoughtWorks®