

HTTPS Test Using **Grinder-3.4**

Version: 1.0
Date: 04th May 2010

© 2009 TATA CONSULTANCY SERVICES

This is a controlled document. Unauthorised access, copying and replication are prohibited.

This document must not be copied in whole or in parts by any means, without the written authorisation of Tata Consultancy Services.

Author

Name	Company/Contact Information
SEF	SEF, Tata Consultancy Services Limited

Document History

Version	Date	Section(s) Changed	Change(s) relative to previous version	Changes by
1.0	May 04 th 2010	All	Initial	-

Table Of Contents

1 Introduction to Grinder-3.4.....	7
1.1 BACKGROUND OF THE DOCUMENT.....	7
1.2 WHAT IS THE GRINDER?.....	7
1.3 KEY FEATURES:.....	8
1.4 WHERE TO OBTAIN THE GRINDER ?.....	8
1.5 WHAT ELSE DO YOU NEED?.....	8
2 The Grinder-3.4.....	9
2.1 WHAT IS NEW IN GRINDER-3.4.....	9
2.2 CAPABILITIES OF GRINDER-3.4.....	10
2.3 STANDARDS OF GRINDER-3.4.....	10
2.4 THE GRINDER ARCHITECTURE.....	11
2.5 THE GRINDER ARCHITECTURE.....	11
2.6 SCRIPT.....	11
2.7 THE GRINDER PLUG-INS.....	12
2.8 TCP PROXY.....	12
3 Getting Started.....	13
3.1 THE GRINDER PROCESSES	13
3.2 THE OUTPUT.....	15
3.3 HOW TO START THE GRINDER-3.4.....	16
3.3.1 Grinder-3.4 Properties file.....	16
3.4 THE CONSOLE.....	24
3.4.1 Process Controls.....	24
3.4.2 Sample Controls.....	25
4 Application to be Tested.....	29
4.1 STEPS TO MAKE THE WEB APPLICATION ACCESSIBLE IN THE BROWSER.....	29
5 Functionality Testing of the Web Application.....	32
5.1 RECORDING HTTPS REQUESTS	36
5.1.1 The TCPProxy.....	36
5.1.2 How to start the TCPProxy?.....	36
5.1.3 Steps for recording HTTPS request.....	38
5.2 DESCRIPTION OF THE SCRIPT GENERATED.....	41
5.3 EDITING SCRIPT FOR CORRELATION AND TEXT VERIFICATION.....	48
5.3.1 CORRELATION FOR VARIABLE “LT”.....	50
5.3.2 Correlation for variable “ticket”.....	51
5.3.3 Writing response of each request to a File.....	52
5.4 EXECUTING THE SCRIPT.....	53
6 Results of the Test	55
6.1 STRENGTH AND WEAKNESS OF GRINDER-3.4.....	60

7 References.....	61
8 Appendix.....	62

1 Introduction to Grinder-3.4

1.1 Background of the Document

This document is prepared based on the experiences gained while using Grinder-3.4 for the performance testing of the HTTPS eMunicipality application of SEF. The document is intended to serve as a guide to using Grinder-3.4.

Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control, and test reporting functions.

Simply put, it is the process of automating the manual testing process currently in use, by the use of software. Hence, this definition goes further than simply using some Word Processor software.

If you decide to have test automation as a part of your testing project, then it's time for you to choose which test automation software to use. There are numerous test automation tools or software's available, and they may cost from zero to tens of thousands of dollars. Your choice will depend on the needs, available resources, budget of the testing project, and certainly, the project funding. If you are faced with testing project challenges like tight budget and schedule and want a quick and reliable test automation solution to your testing needs then Grinder is your right choice.

1.2 What is The Grinder?

The Grinder is a freely available load-generating and data collection tool with which to carry out the tests. However, the methodology is not tied to The Grinder. It is equally applicable to any load-generating tool of your choice. The Grinder is the primary performance measurement tool.

The Grinder's ease of use and adaptability to a wide range of J2EE performance problems. The Grinder is also difficult to beat on price – it's freely available under a BSD-style license.

The Grinder is a Java™ load testing framework that makes it easy to run a distributed test using many load injector machines. Test scripts make use of client code embodied in Java plug-ins. Most users of The Grinder do not write plug-ins themselves, instead they use one of the supplied plug-ins. The Grinder comes with a mature plug-in for testing HTTP services, as well as a tool which allows HTTP scripts to be automatically recorded.

Grinder only records the HTTP/HTTPS requests and responses and not the mouse clicks or keystrokes. In short, Grinder tool is free ware software used typically for Load and Performance Tests. However, the architectural design means it could be capable of much more.

1.3 Key Features:

Listed below are some of the key features of Grinder-3.4 performance testing tool with regard to load testing of Web applications:

- **Generic Approach** Load test anything that has a Java API. This includes common cases such as HTTP web servers, SOAP and REST web services, and application servers (CORBA, RMI, JMS, EJBs), as well as custom protocols.
- **Flexible Scripting** Tests are written in the powerful Jython scripting language.
- **Distributed Framework** A graphical console allows multiple load injectors to be monitored and controlled, and provides centralised script editing and distribution.
- **Mature HTTP Support** Automatic management of client connections and cookies. SSL. Proxy aware. Connection throttling. Sophisticated record and replay of the interaction between a browser and a web site.

Grinder is a free ware and is available for download at <http://grinder.sourceforge.net> site.

1.4 Where to Obtain The Grinder ?

The Grinder's home page is <http://grinder.sourceforge.net>. From here you should click the link for the Sourceforge summary page in order to download The Grinder distribution.

The Grinder 3 is distributed as two zip files which you should expand using unzip, WinZip (<http://www.winzip.com/>) or similar. Everything required to run The Grinder is in the zip file labelled grinder-3.4.zip. The remaining files that are needed to build The Grinder are distributed in the zip file labelled grinder-3.4-src.zip; these are mainly of interest to developers wanting to extend The Grinder tool.

We will use the binary code present in grinder-3.4.zip file in order to execute the tool and perform the load test.

1.5 What else do you need?

To run the Grinder-3.4 tool, we require Java2 Standard Edition 1.4 or later. We will be using JDK5.0 in our case, which can be downloaded from the following web address http://java.sun.com/javase/downloads/index_jdk5.jsp

As we will be testing a HTTPS web application, so optionally we may require JSSE (Java Secure Socket Extension) 1.0.3 which can be downloaded from the following web address <http://java.sun.com/products/archive/jsse/>

2 The Grinder-3.4

2.1 What is new in Grinder-3.4

- **Jython** : The most significant change that The Grinder 3 introduces is a Jython(<http://www.jython.org/>) scripting engine that is used to execute test scripts. Jython is a Java implementation of the popular Python language. Test scripts specify the tests to run. In The Grinder 2, tests were specified in the grinder.properties file. The grinder.properties file is still used to specify general control information (how to contact the console, how many worker processes to use, ..), as well as the name of the test script. Many other key features of The Grinder 2, such as the console and the process architecture, remain unchanged. It allows arbitrary branching and looping and makes test results directly available to the test script, allowing different test paths to be taken depending on the outcome of each test. The Grinder 3 can use the full power of Jython to create dynamic requests of arbitrary complexity.
- **Test any Java code** : The Grinder 3 allows any Java (or Jython) code to be encapsulated as a test. This practically removes the need to write custom plug-ins. Although plug-ins are no longer responsible for performing tests, they can still be useful to manage objects that the tests use. For example, the HTTP plug-in manages a pool of connections for each worker thread, and provides an HTTPRequest object that makes use of these connections.
- **Dynamic test scripting**: The Grinder 2 worker processes execute tests in the properties file sequentially in a fixed order, and there is limited support in some of the The Grinder 2 plug-ins for checking test results. The Grinder 3 allows arbitrary branching and looping and makes test results directly available to the test script, allowing different test paths to be taken depending on the outcome of each test. The Grinder 2 HTTP plug-in's string bean feature provides simple support for requests that contain dynamic data. The Grinder 3 can use the full power of Jython to create dynamic requests of arbitrary complexity.
- **Other Changes:**
 - The TCPSniffer has been renamed TCPProxy for correctness. The TCPProxy can be used with other HTTP and HTTPS proxies. Many new features and fixes have been added to the TCPProxy.
 - Console signals are now transmitted over a TCP socket connection. Multicast is no longer used, removing a frequent source of set up problems.
 - The interface that plug-ins must implement has changed significantly. Plug-ins written for The Grinder 2 will not work with The Grinder 3.
 - Many grinder.properties have been removed. The features formerly accessed through setting properties are now set by making calls to the plug-in from test scripts. Some of the remaining property names have been renamed.
 - HTTP tests can now be directed through an HTTP proxy.

- String beans and OK strings have been removed from the HTTP plug-in. String beans and OK strings are very limited in comparison to the flexibility now provided by Jython scripts.
- The HttpURLConnection implementation has been removed from the HTTP plug-in.
- Many other minor updates to HTTP testing functionality.
- HTTPS and SSL contexts can now be controlled on a per thread basis.
- The JUnit and Socket plug-ins have been removed. Their functionality can be achieved directly by using the appropriate Java objects in scripts.
- From the console you can edit test scripts and distribute them to the worker processes. It is no longer necessary to copy grinder.properties files around or to use a shared disk.

2.2 *Capabilities of Grinder-3.4*

- **Load Testing:** Load Testing determines if an application can support a specified load (for example, 500 concurrent users) with specified response times. Load Testing is used to create benchmarks.
- **Capacity Testing:** Capacity Testing determines the maximum load that an application can sustain before system failure.
- **Functional Testing:** Functional Testing proves the correct behaviour of an application.
- **Stress Testing:** Stress Testing is load testing over an extended period of time. Stress Testing determines if an application can meet specified goals for stability and reliability, under a specified load, for a specified time period.

2.3 *Standards of Grinder-3.4*

- **100% Pure Java:** The Grinder works on any hardware platform and any operating system that supports J2SE 1.4 and above.
- **Web Browsers:** The Grinder can simulate web browsers and other devices that use HTTP, and HTTPS.
- **Web Services:** The Grinder can be used to test Web Service interfaces using protocols such as SOAP and XML-RPC.
- **Database:** The Grinder can be used to test databases using JDBC.
- **Middleware:** The Grinder can be used to test RPC and MOM based systems using protocols such as IIOP, RMI/IIOP, RMI/JRMP, and JMS.

- **Other Internet protocols:** The Grinder can be used to test systems that utilise other protocols such as POP3, SMTP,FTP, and LDAP.

2.4 *The Grinder Architecture*

- **Goal:** Minimize system resource requirements while maximizing the number of test contexts ("virtual users").
- **Multi-threaded, multi-process:** Each test context runs in its own thread. The threads can be split over many processes depending on the requirements of the test and the capabilities of the load injection machine.
- **Distributed** The Grinder makes it easy to coordinate and monitor the activity of processes across a network of many load injection machines from a central console.
- **Scalable** The Grinder typically can support several hundred HTTP test contexts per load injection machine. (The number varies depending on the type of test client). More load injection machines can be added to generate bigger loads.

2.5 *The Grinder Architecture*

- **Graphical Interface** 100% Java Swing user interface.
- **Process coordination** Worker processes can be started, stopped and reset from one central console.
- **Process monitoring** Dynamic display of current worker processes and threads.
- **Internationalised and Localised** English, French, Spanish, and German translations are supplied. Users can add their own translations.
- **Script editing** Central editing and management of test scripts.

2.6 *Script*

- **Record real users** Scripts can be created by recording actions of a real user using the TCP Proxy. The script can then be customised by hand.
- **Powerful scripting in Python** Simple to use but powerful, fully object-oriented scripting.
- **Multiple scenarios** Arbitrary looping and branching allows the simulation of multiple scenarios. Simple scenarios can be composed into more complex scenarios. For example, you might allocate 10% of test contexts to a login scenario, 70% to searching, 10% to browsing, and 10% to buying; or you might have different workloads for specific times of a day.
- **Access to any Java API** Jython allows any Java-based API to be used directly from the test script.

- **Parameterization of input data:** Input data (e.g. URL parameters, form fields) can be dynamically generated. The source of the data can be anything including flat files, random generation, a database, or previously captured output.
- **Content Verification** Scripts have full access to test results. In the future, The Grinder will include support for enhanced parsing of common results such as HTML pages.

2.7 *The Grinder Plug-ins*

- **HTTP:** The Grinder has special support for HTTP that automatically handles cookie and connection management for test contexts.
- **Custom:** Users can write their own plug-ins to a documented interface; although this is rarely necessary due to the powerful scripting facilities.
- **HTTP 1.0, HTTP 1.1:** Support for both HTTP 1.0 and HTTP 1.1 is provided.
- **HTTPS :** The Grinder supports HTTP over SSL.
- **Cookies:** Full support for Cookies is provided.
- **Multi-part forms:** The Grinder supports multi-part forms.
- **Connection throttling:** Low bandwidth client connections can be simulated.

2.8 *TCP Proxy*

- **TCP proxy:** A TCP proxy utility is supplied that can be used to intercept system interaction at the protocol level. It is useful for recording scripts and as a debugging tool.
- **HTTP Proxy:** The TCP proxy can be configured as an HTTP/HTTPS proxy for easy integration with web browsers.
- **SSL Support:** The TCP proxy can simulate SSL sessions.
- **Filter-based architecture:** The TCP proxy has a pluggable filter architecture. Users can write their own filters.

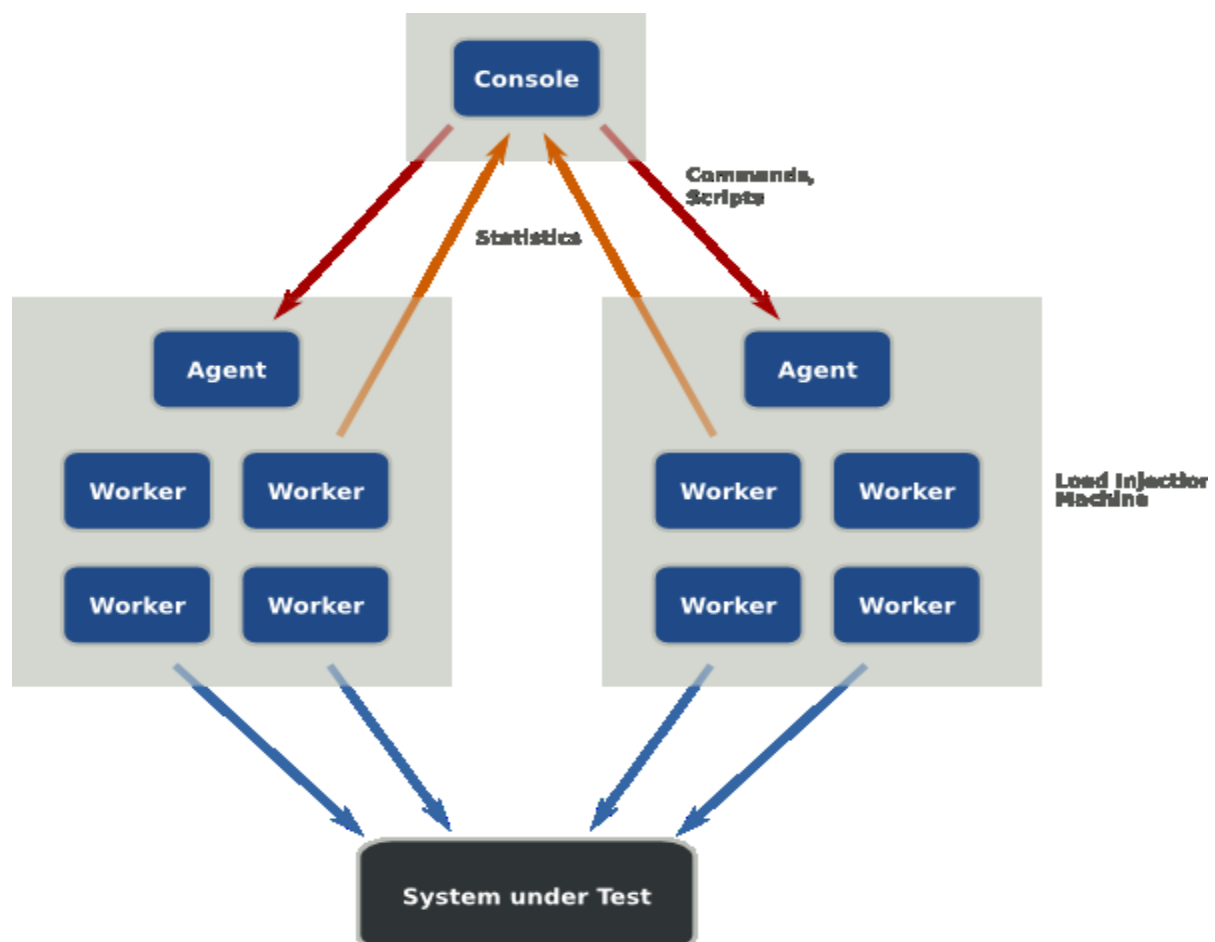
3 Getting Started

3.1 The Grinder Processes

The Grinder is a framework for running test scripts across a number of machines. The framework is comprised of three types of process (or program): **worker processes**, **agent processes**, and the **console**.

- **Worker processes**, which are created by Grinder agent processes and are responsible for performing the individual tests. Interpret Jython test scripts and perform tests using a number of worker threads.
- **Agent processes**, a single agent process runs on each test client machine and is responsible for managing worker processes on that machine.
- **The console**, which coordinates the other processes and collates statistics or results. Console also helps in editing and distribution of scripts.

As The Grinder is written in Java, each of these processes is a Java Virtual Machine (JVM).



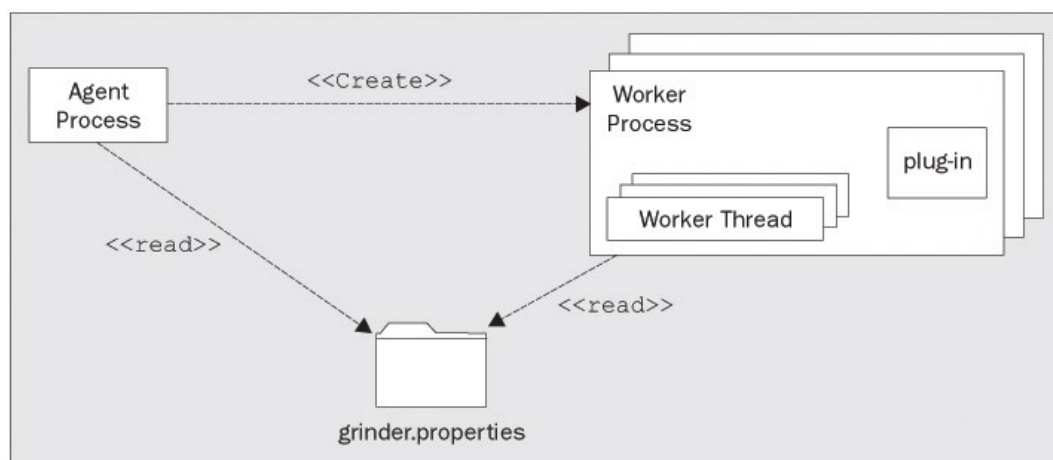
For heavy duty testing, you start an agent process on each of several load injector machines. The worker processes they launch can be controlled and monitored using the console. There is little reason to run more than one agent on each load injector, but you can if you wish.

A test is a unit of work against which statistics are recorded. Tests are uniquely defined by a test number and also have a description. Users specify which tests to run using aJython test script . If you wish your scripts can report many different actions (e.g. different web page requests) against the same test, The Grinder will aggregate the results.

The script is executed many times in a typical testing scenario. Each worker process has a number of worker threads, and each worker thread calls the script a number of times. A single execution of a test script is called a run.

You can write scripts for use with the Grinder by hand. There are a number of examples of how to do this in the Script.

If you are creating a script to test a web site or web application, you can use the TCPProxy to record a browser session as a script.



Each worker process sets up a network connection to the console to report statistics. Each agent process sets up a connection to the console to receive commands, which it passes on to its worker processes. The console listens for both types of connection on a particular address and port. By default, the console listens on port 6372 on all local network interfaces of the machine running the console.

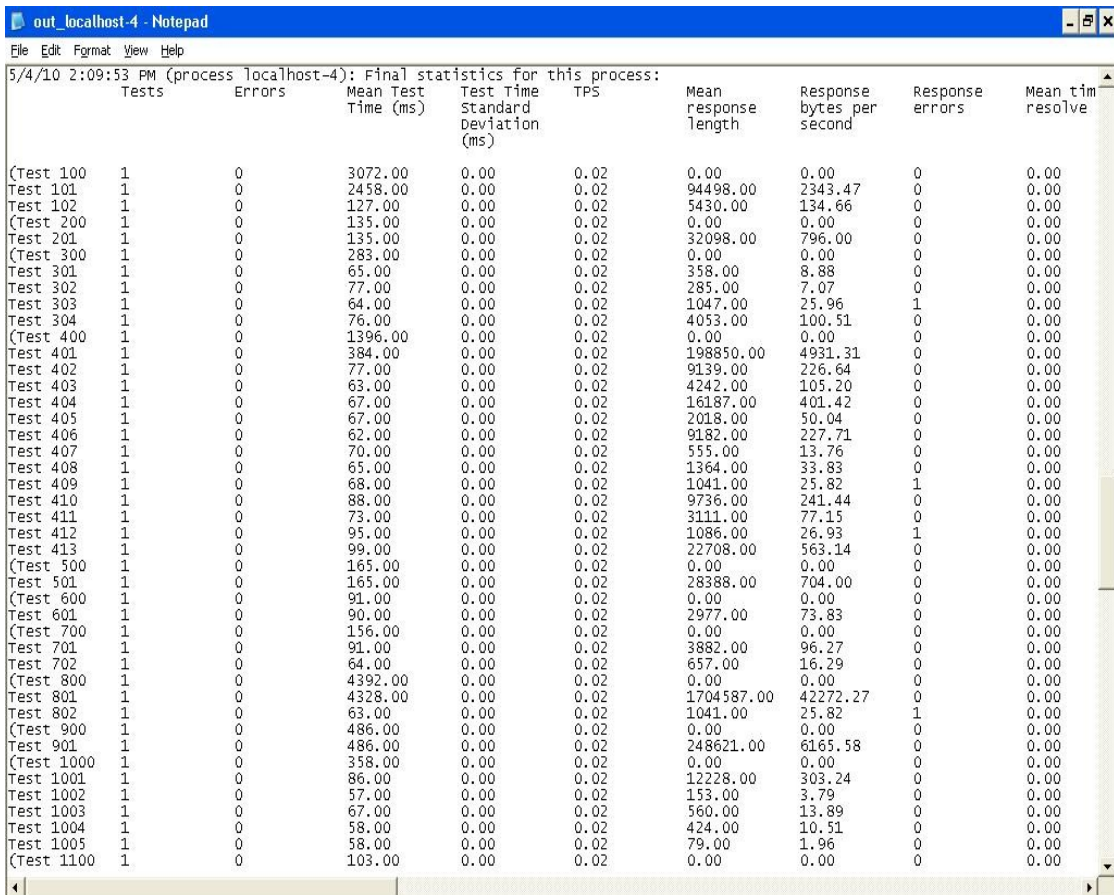
If an agent process fails to connect to the console, or the grinder.useConsole property is false, the agent will continue independently without the console and automatically will start its worker processes. The worker processes will run to completion and not report to the console. This can be useful when you want to quickly try out a test script without bothering to start the console.

3.2 The Output

Each worker process writes logging information to a file called out-host-n.log, where host is the machine host name and n is the worker process number. Errors are written to error-host-n.log. If no errors occur, an error file will not be created.

Data about individual test invocations is written into a file called data-host-n.log that can be imported into a spreadsheet tool such as Microsoft Excel™ for further analysis. The data file is the only place where information about individual tests is recorded; the console displays only aggregate information.

The final statistics summary (in the out-* files of each process) looks something like this:



Tests	Errors	Mean Test Time (ms)	Test Time Standard Deviation (ms)	TPS	Mean response length	Response bytes per second	Response errors	Mean time resolve
(Test 100	1	0	3072.00	0.00	0.02	0.00	0.00	0.00
Test 101	1	0	2458.00	0.00	0.02	94498.00	2343.47	0.00
Test 102	1	0	127.00	0.00	0.02	5430.00	134.66	0.00
(Test 200	1	0	135.00	0.00	0.02	0.00	0.00	0.00
Test 201	1	0	135.00	0.00	0.02	32098.00	796.00	0.00
(Test 300	1	0	283.00	0.00	0.02	0.00	0.00	0.00
Test 301	1	0	65.00	0.00	0.02	358.00	8.88	0.00
Test 302	1	0	77.00	0.00	0.02	285.00	7.07	0.00
Test 303	1	0	64.00	0.00	0.02	1047.00	25.96	0.00
Test 304	1	0	76.00	0.00	0.02	4053.00	100.51	0.00
(Test 400	1	0	1396.00	0.00	0.02	0.00	0.00	0.00
Test 401	1	0	384.00	0.00	0.02	198850.00	4931.31	0.00
Test 402	1	0	77.00	0.00	0.02	9139.00	226.64	0.00
Test 403	1	0	63.00	0.00	0.02	4242.00	105.20	0.00
Test 404	1	0	67.00	0.00	0.02	16187.00	401.42	0.00
Test 405	1	0	67.00	0.00	0.02	2018.00	50.04	0.00
Test 406	1	0	62.00	0.00	0.02	9182.00	227.71	0.00
Test 407	1	0	70.00	0.00	0.02	555.00	13.76	0.00
Test 408	1	0	65.00	0.00	0.02	1364.00	33.83	0.00
Test 409	1	0	68.00	0.00	0.02	1041.00	25.82	0.00
Test 410	1	0	88.00	0.00	0.02	9736.00	241.44	0.00
Test 411	1	0	73.00	0.00	0.02	3111.00	77.15	0.00
Test 412	1	0	95.00	0.00	0.02	1086.00	26.93	0.00
Test 413	1	0	99.00	0.00	0.02	22708.00	563.14	0.00
(Test 500	1	0	165.00	0.00	0.02	0.00	0.00	0.00
Test 501	1	0	165.00	0.00	0.02	28388.00	704.00	0.00
(Test 600	1	0	91.00	0.00	0.02	0.00	0.00	0.00
Test 601	1	0	90.00	0.00	0.02	2977.00	73.83	0.00
(Test 700	1	0	156.00	0.00	0.02	0.00	0.00	0.00
Test 701	1	0	91.00	0.00	0.02	3882.00	96.27	0.00
Test 702	1	0	64.00	0.00	0.02	657.00	16.29	0.00
(Test 800	1	0	4392.00	0.00	0.02	0.00	0.00	0.00
Test 801	1	0	4328.00	0.00	0.02	1704587.00	42272.27	0.00
Test 802	1	0	63.00	0.00	0.02	1041.00	25.82	0.00
(Test 900	1	0	486.00	0.00	0.02	0.00	0.00	0.00
Test 901	1	0	486.00	0.00	0.02	248621.00	6165.58	0.00
(Test 1000	1	0	358.00	0.00	0.02	0.00	0.00	0.00
Test 1001	1	0	86.00	0.00	0.02	12228.00	303.24	0.00
Test 1002	1	0	57.00	0.00	0.02	153.00	3.79	0.00
Test 1003	1	0	67.00	0.00	0.02	560.00	13.89	0.00
Test 1004	1	0	58.00	0.00	0.02	424.00	10.51	0.00
Test 1005	1	0	58.00	0.00	0.02	79.00	1.96	0.00
(Test 1100	1	0	103.00	0.00	0.02	0.00	0.00	0.00

The console has a dynamic display of similar information collected from all the worker processes. Plug-ins and advanced test scripts can provide additional statistics; for example, the HTTP plug-in adds a statistic for the content length of the response body.

Each test has one of two possible outcomes:

1. Success. The number of Successful Tests for that test is incremented. The time taken to perform the test is added to the Total.
2. Error. The execution of a test raised an exception. The number of Errors for the test is incremented. The time taken is discarded.

The Total, Mean, and Standard Deviation figures are calculated based only on successful tests.

3.3 How to start the Grinder-3.4

Grinder tool can be started by following the given steps:

Step 1: Create a `grinder.properties` file in the grinder root directory where all the grinder files and their respective folders are present. This file specifies general control information (how the worker processes should contact the console, how many worker processes to use, ..), as well as the name of the Jython script that will be used to run the tests.

3.3.1 Grinder-3.4 Properties file

The Grinder worker and agent processes are controlled by setting properties in the `grinder.properties` file.

All properties have default values. If you start The Grinder agent process without a `grinder.properties` file it will communicate with the console using default addresses, use one worker process, one thread, and make one run through the test script found in the file `grinder.py`. This is not much use, so we have to write a specific `grinder.properties` file for each test scenario.

Table lists the properties understood by The Grinder engine includes:

Property	Description	Default
<code>grinder.processes</code>	The number of worker processes the agent should Start.	1
<code>grinder.threads</code>	The number of worker threads that each worker process Spawns.	1
<code>grinder.runs</code>	The number of runs of the test script each thread performs. 0 means "run forever", and should be used when you are using the console to control your test runs.	1
<code>grinder.processIncrement</code>	Used in conjunction with <code>grinder.processIncrement</code> , this property sets the interval in milliseconds at which the agent starts new worker processes.	60000
<code>grinder.duration</code> <code>grinder.script</code>	The maximum length of time in milliseconds that each worker process should run for. <code>grinder.duration</code> can be specified in conjunction with <code>grinder.runs</code> , in which case the worker processes will terminate if either the duration time or the number of runs is Exceeded.	Runs Forever
	The file name of the Jython script to run	<code>grinder.py</code>

<code>grinder.logDirectory</code>	Directory to write log files to. Created if it doesn't already Exist.	The local directory.
<code>grinder.numberOfOldLogs</code>	The number of archived logs from previous runs that should be kept.	1
<code>grinder.hostID</code>	Override the "host" string used in log filenames and logs.	The host name.
<code>grinder.consolePort</code>	The IP port that the agent and worker processes use to contact the console.	6372
<code>grinder.useConsole</code>	Set to <code>false</code> to set the agent and worker processes not to use the console.	TRUE
<code>grinder.reportToConsole</code>	The period at which each process sends updates to the console. This also controls the frequency at which the data files are flushed.	500 ms
<code>grinder.logProcessStream</code>	Set to <code>false</code> to disable the logging of output and error streams for worker processes. You might want to use this to reduce the overhead of running a client thread.	TRUE
<code>grinder.debug.singleprocess</code>	Set to <code>true</code> , the agent process spawns engines in threads rather than processes, using special class loaders to isolate the engines. This allows the engine to be easily run in a debugger. This is primarily a tool for debugging The Grinder engine, but it might also be useful to advanced users.	FALSE

A Sample `Grinder.properties` file used to perform Load test on eMunicipality application is shown below:

```
# Represent comment statement

# Sample grinder.properties
#
#
# The properties can be specified in three ways.
# - In the console. A properties file in the distribution directory
#   can be selected in the console.
# - As a Java system property in the command line used to start the
#   agent. (e.g. java -Dgrinder.threads=20 net.grinder.Grinder).
# - In an agent properties file. A local properties file named
#   "grinder.properties" can be placed in the working directory of
#   each agent, or a different file name passed as an agent command
#   line argument.
#
```

```

# Properties present in a console selected file take precedence over
# agent command line properties, which in turn override those in
# an agent properties file.
#
# Any line which starts with a ; (semi-colon) or a # (hash) is a
# comment and is ignored. In this example we will use a # for
# commentary and a ; for parts of the config file that you may wish
to
# enable
#
# Please refer to
# http://net.grinder.sourceforge.net/g3/properties.html for further
# documentation.

#
# Commonly used properties
#

# The file name of the script to run.
#
# Relative paths are evaluated from the directory containing the
# properties file. The default is "grinder.py".
grinder.script = rajesh1.py

# The number of worker processes each agent should start. The default
# is 1.
grinder.processes = 1

# The number of worker threads each worker process should start. The
# default is 1.
grinder.threads = 1

# The number of runs each worker process will perform. When using the
# console this is usually set to 0, meaning "run until the console
# sends a stop or reset signal". The default is 1.
grinder.runs = 1

# The IP address or host name that the agent and worker processes use
# to contact the console. The default is all the network interfaces
# of the local machine.
grinder.consoleHost = localhost

# The IP port that the agent and worker processes use to contact the
# console. Defaults to 6372.
grinder.consolePort = 6372

grinder.plugin=net.grinder.plugin.http.HttpPlugin
grinder.plugin.parameter.useCookies=true
grinder.plugin.parameter.followRedirects=true

#
# Less frequently used properties
#

### Logging ###

# The directory in which worker process logs should be created. If
not
# specified, the agent's working directory is used.

```

```

grinder.logDirectory = log

# The number of archived logs from previous runs that should be kept.
# The default is 1.
grinder.numberOfOldLogs = 0

# Overrides the "host" string used in log filenames and logs. The
# default is the host name of the machine running the agent.
grinder.hostID = localhost

# Set to false to disable the logging of output and error streams for
# worker processes. You might want to use this to reduce the overhead
# of running a client thread. The default is true.
grinder.logProcessStreams = true

### Script sleep time ###

# The maximum time in milliseconds that each thread waits before
# starting. Unlike the sleep times specified in scripts, this is
# varied according to a flat random distribution. The actual sleep
# time will be a random value between 0 and the specified value.
# Affected by grinder.sleepTimeFactor, but not
# grinder.sleepTimeVariation. The default is 0ms.
grinder.initialSleepTime=100

# Apply a factor to all the sleep times you've specified, either
# through a property of in a script. Setting this to 0.1 would run
the
# script ten times as fast. The default is 1.
grinder.sleepTimeFactor= 1

# The Grinder varies the sleep times specified in scripts according
to
# a Normal distribution. This property specifies a fractional range
# within which nearly all (99.75%) of the times will lie. E.g., if
the
# sleep time is specified as 1000 and the sleepTimeVariation is set
to
# 0.1, then 99.75% of the actual sleep times will be between 900 and
# 1100 milliseconds. The default is 0.2.
; grinder.sleepTimeVariation=0.005

### Worker process control ###

# If set, the agent will ramp up the number of worker processes,
# starting the number specified every
# grinder.processesIncrementInterval milliseconds. The upper limit is
# set by grinder.processes. The default is to start all worker
# processes together.
; grinder.processIncrement = 1

# Used in conjunction with grinder.processIncrement, this property
# sets the interval in milliseconds at which the agent starts new
# worker processes. The value is in milliseconds. The default is
60000
# ms.
; grinder.processIncrementInterval = 10000

# Used in conjunction with grinder.processIncrement, this property

```

```

# sets the initial number of worker processes to start. The default
is
# the value of grinder.processIncrement.
; process.initialProcesses = 1

# The maximum length of time in milliseconds that each worker process
# should run for. grinder.duration can be specified in conjunction
# with grinder.runs, in which case the worker processes will
terminate
# if either the duration time or the number of runs is exceeded. The
# default is to run forever.
grinder.duration = 0

# If set to true, the agent process spawns engines in threads rather
# than processes, using special class loaders to isolate the engines.
# This allows the engine to be easily run in a debugger. This is
# primarily a tool for debugging The Grinder engine, but it might
also
# be useful to advanced users. The default is false.
grinder.debug.singleprocess = true

# If set to true, the new DCR instrumentation engine will be used.
The
# new engine will always be used if Jython 2.1/2.2 is not found.
; grinder.dcrinstrumentation = false

### Java ###

# Use an alternate JVM for worker processes. The default is "java" so
# you do not need to specify this if java is in your PATH.
; grinder.jvm = /opt/jrockit/jrockit-R27.5.0-jdk1.5.0_14/bin/java

# Use to adjust the classpath used for the worker process JVMs.
# Anything specified here will be prepended to the classpath used to
# start the Grinder processes.
; grinder.jvm.classpath = /tmp/myjar.jar

# Additional arguments to worker process JVMs.
; grinder.jvm.arguments = -Dpython.cachedir=/tmp

### Console communications ###

# (See above for console address properties).

# If you are not using the console, and don't want the agent to try
to
# contact it, set grinder.useConsole = false. The default is true.
grinder.useConsole = true

# The period at which each process sends updates to the console. This
# also controls the frequency at which the data files are flushed.
# The default is 500 ms.
; grinder.reportToConsole.interval = 100

### Statistics ###

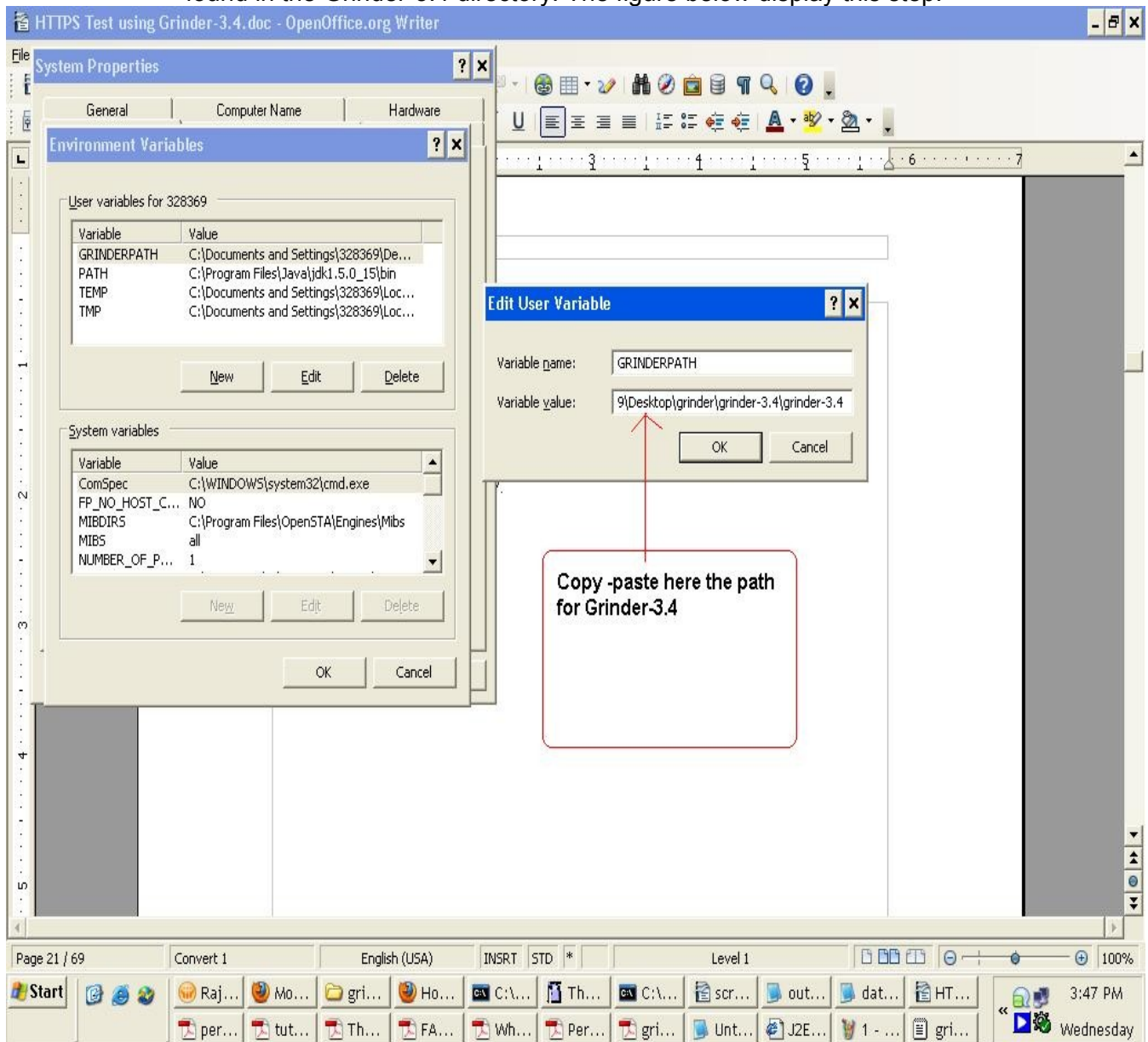
# Set to false to disable reporting of timing information to the
# console; other statistics are still reported. See

```

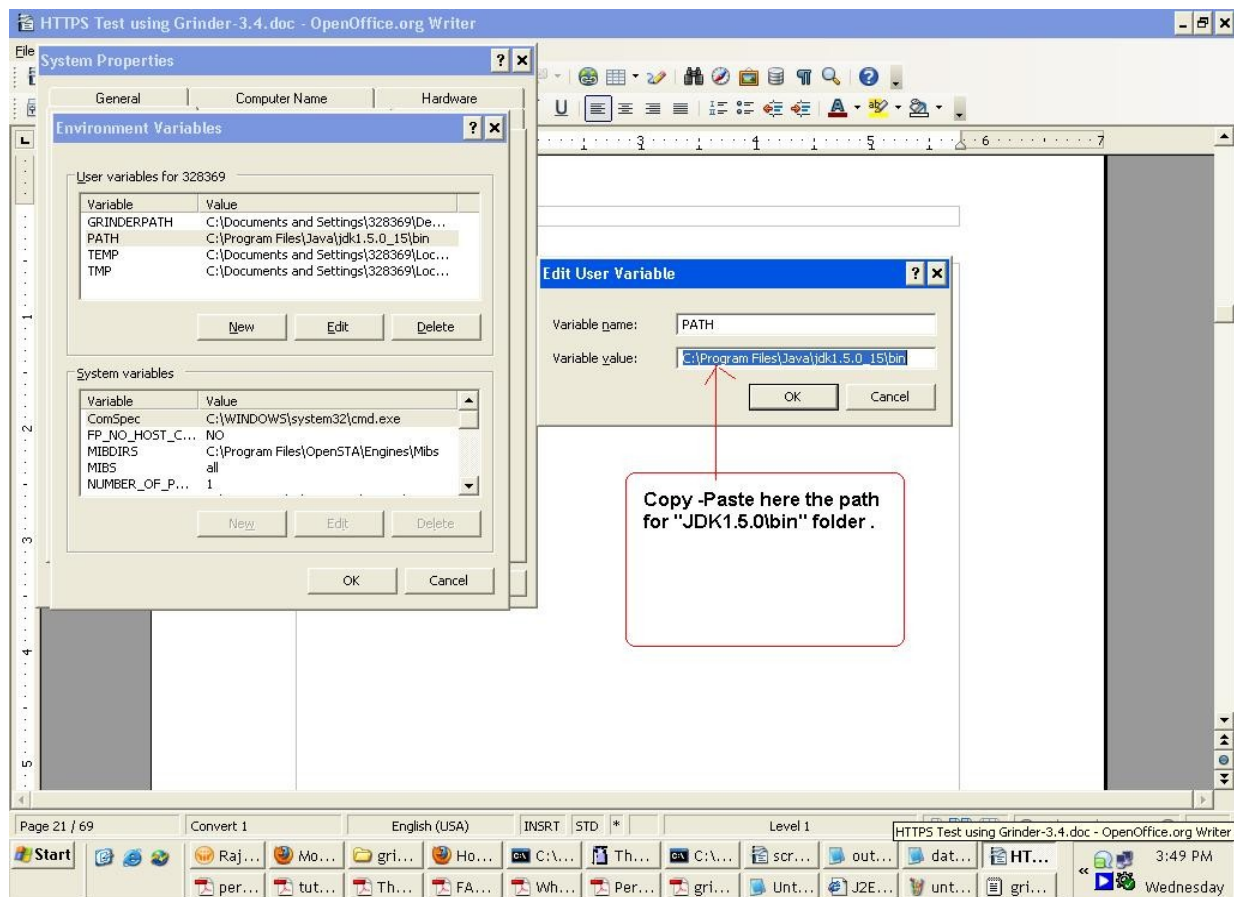
```
# http://grinder.sourceforge.net/faq.html#timing for why you might
# want to do this. The default is true.
grinder.reportTimesToConsole = true
```

The Script file that will be used to test a particular transaction of the eMunicipality application is named as rajesh.py as shown in the grinder.properties file.

Step 2: Set your CLASSPATH to include the grinder.jar file which can be found in the Grinder-3.4 directory. The figure below display this step.



Similarly set the CLASSPATH to include all the files present in “JDK1.5.0/bin” directory. The figure below display this step.



Step 3: Create a batch file called *setGrinderEnv.bat* and place the file in the Grinder-3.4 directory which contains all the grinder files and other folders like “lib”, “contrib” etc. The use of this batch file will be to set the Grinder environment by making all the required class files available in the class path, so that proper functioning of the console and agent can take place.

Content of “setGrinderEnv.bat” file is:

```
set GRINDERPATH="C:\Documents and
Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4"
```

```
set GRINDERPROPERTIES="C:\Documents and
Settings\328369\Desktop\grinder\grinder-3.4\grinder-
3.4\grinder.properties"
```

```
set CLASSPATH=%GRINDERPATH%\lib\grinder.jar;%GRINDERPATH
%\lib\jython.jar;
```

```
set JAVA_HOME="C:\Program Files\Java\jdk1.5.0_15"
PATH=%JAVA_HOME%\bin;%PATH%
```

Explanation: set GRINDERPATH="path for Grinder-3.4 folder on your system"

set GRINDERPROPERTIES="path for grinder.properties file"

Copy -paste the other two lines as shown in your *setGrinderEnv.bat* file.

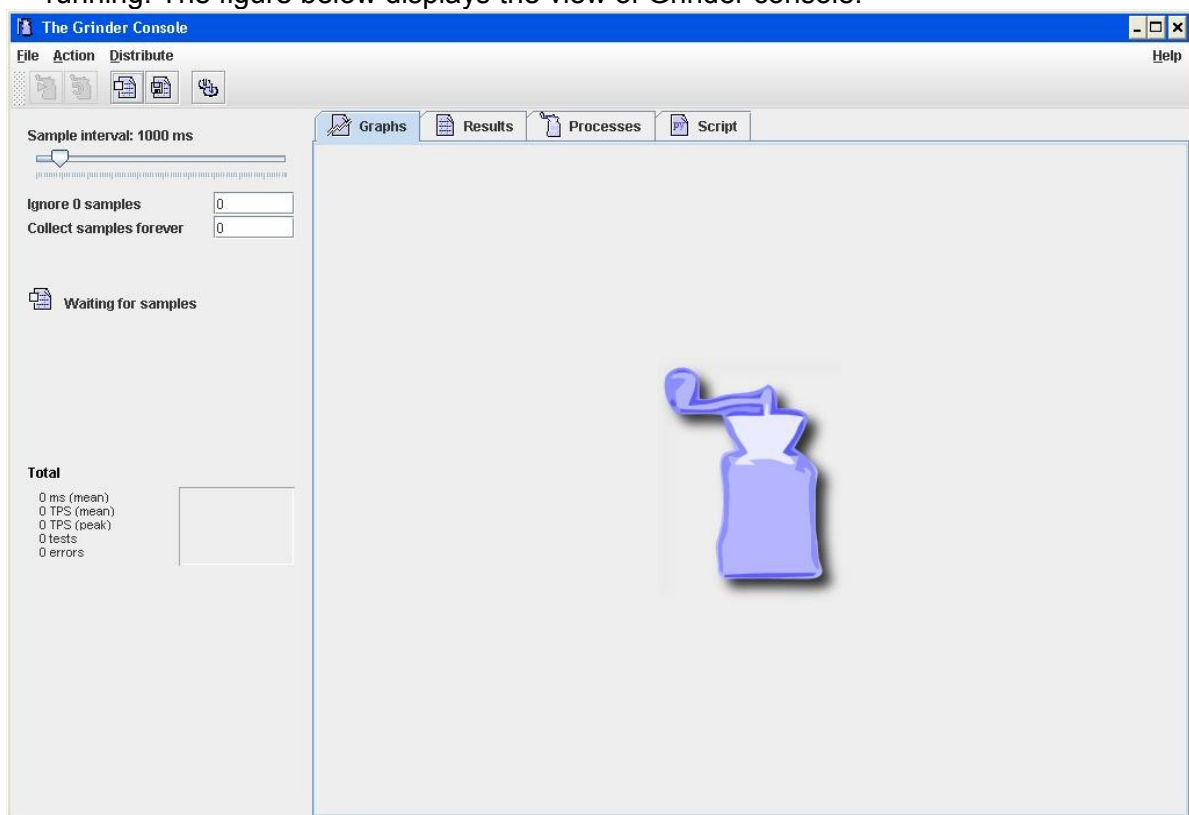
Step 4: Create a batch file called *startConsole.bat*, which will be used to start the console of Grinder-3.4 .Place the batch file in the Grinder-3.4 directory along with *setGrinderEnv.bat* file.
Content of *startConsole.bat* file is shown below:

```
call "C:\Documents and
Settings\328369\Desktop\grinder\grinder-3.4\grinder-
3.4\setGrinderEnv.bat"

java -cp %CLASSPATH% net.grinder.Console
```

Explanation: call “path for setGrinderEnv.bat file”.
Copy paste the other line in your startConsole.bat file

When we execute the “startConsole.bat” file then the console of Grinder-3.4 starts running. The figure below displays the view of Grinder console.



Step 5: Create a batch file called *startAgent.bat* which will be used to bring Agent of Grinder-3.4 under execution. Place the batch file in the Grinder-3.4 directory along with *setGrinderEnv.bat*, *startConsole.bat* file.
Content of *startAgent.bat* file is shown below.

```
call "C:\Documents and
Settings\328369\Desktop\grinder\grinder-3.4\grinder-
3.4\setGrinderEnv.bat"

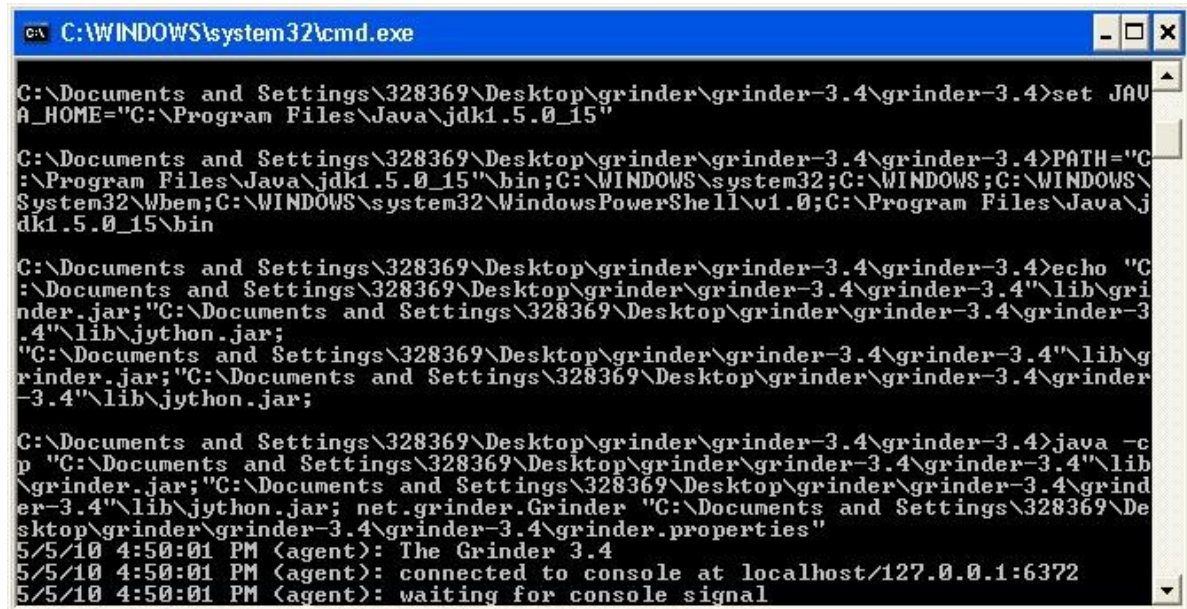
echo %CLASSPATH%

java -cp %CLASSPATH% net.grinder.Grinder %GRINDERPROPERTIES%
```


Explanation: call "path for setGrinderEnv.bat file".

Copy paste the other two line in your startAgent.bat file

When we execute the "startAgent.bat" file then the Agent of Grinder-3.4 starts executing. The figure below displays the view of Agent Under Execution.



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>set JAVA_HOME="C:\Program Files\Java\jdk1.5.0_15"

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>PATH="C:\Program Files\Java\jdk1.5.0_15\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\system32\WindowsPowerShell\v1.0;C:\Program Files\Java\jdk1.5.0_15\bin"

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>echo "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar;"

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>java -cp "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar; net.grinder.Grinder "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\grinder.properties"
5/5/10 4:50:01 PM (agent): The Grinder 3.4
5/5/10 4:50:01 PM (agent): connected to console at localhost/127.0.0.1:6372
5/5/10 4:50:01 PM (agent): waiting for console signal
```

Now after the successful execution of Agent, we are all set to start the load test on the HTTPS web application. In order to perform a load test, we have to record scripts for each transaction which will be performed on the web application by the tool. The script can be recorded by the help of TCP Sniffer. The process of using TCP sniffer to record scripts will be seen shortly, but first let us explain some of the functionality of console window of Grinder-3.4.

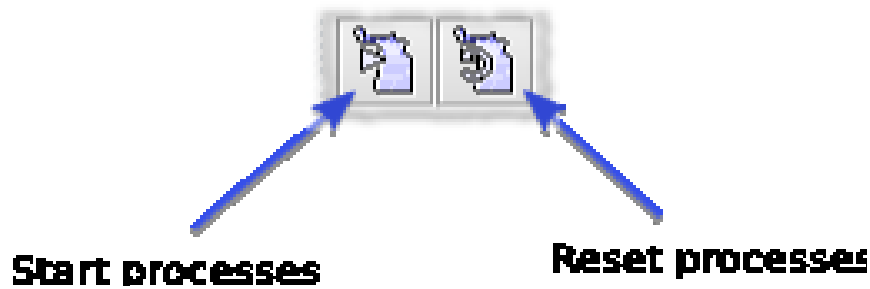
3.4 The Console

The Grinder Console is a useful interface to the workings and reporting facilities of the Grinder.

The worker processes report statistics to the console using a TCP connection. The console listens for connections on a TCP port. Any free TCP port can be used. TCP ports have numbers in the range 0 to 65535 but we recommend using a port above 1024 since, under UNIX, ports with numbers less than 1024 are treated as reserved and require the console to be started with root permission.

3.4.1 Process Controls

The Start processes, Reset processes, and Stop processes menu items send signals to Grinder processes that are listening. Start processes and Reset processes are also tool bar buttons.



These controls will be disabled if no agents are connected to the console. You can check whether any agents are connected on the Processes tab.

Worker processes that are controlled by the console can be in one of three states:

1. Initiated (waiting for a console signal)
2. Running (performing tests, reporting to console)
3. Finished (waiting for a console signal)

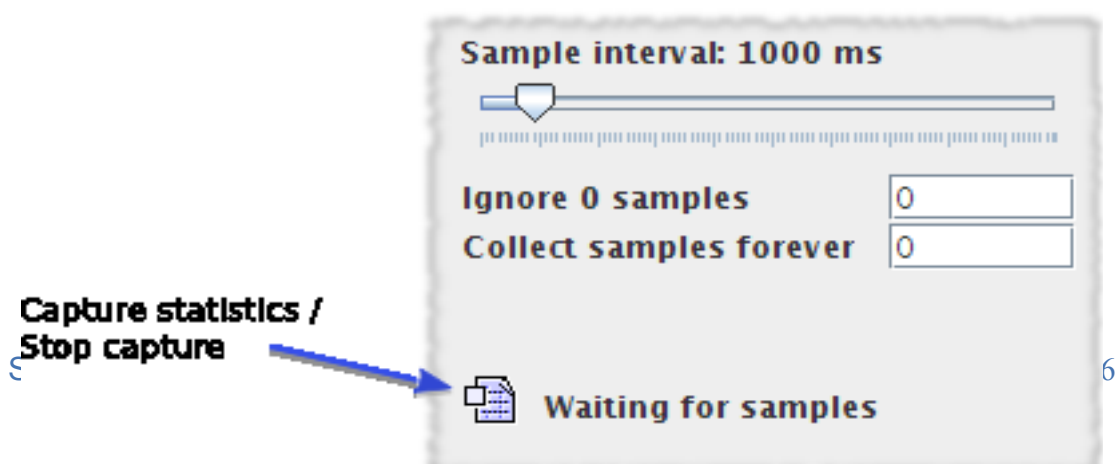
The Start processes control signals to worker processes that they should move into the running state. Processes that are already running will ignore this signal. Processes that are in the finished state exit; the agent process will then reread the properties file, and launch new worker processes in the running state.

The Reset processes control signals all the worker processes to exit. The agent process will then reread the properties file and launch new worker processes.

The Stop processes control signals all processes, including the agent processes, to exit. This is infrequently used, you usually want to use Reset processes instead.

3.4.2 Sample Controls

The sample controls determine how the console captures reports from the worker processes. It is important to understand that these only control the console behaviour. For example, they do not adjust the frequency at which the worker processes send reports. Additionally, the sample controls do not interact in any way with the process controls.



The slider controls the period at which the console will take a sample. This involves adding up all the reports received over that sample interval and calculating the TPS as (number of tests that occurred)/(interval length). It is also the period at which the console graphs and statistics are updated.

By default, the console starts updating the display and calculating totals from the first non-zero sample period. A non-zero sample period is one in which an update from a worker process was received. You can adjust how many non-zero sample periods the console ignores before starting capture with the ignore samples text field.

The third control allows you to adjust how many samples the console will collect before stopping capture. You can also manually start and stop the sampling with the Capture statistics/Stop capture control. Use the Save statistics control to save the current set of statistics to a file.

On the console there are four tabs which display information about The Grinder and its tests. These are detailed below:

1) **Graphs tab:** Each graph displays the 25 most recent Tests Per Second (TPS) values for a particular test. A new value is added every console sample period. The y-axis is scaled so that the full height represents the peak TPS value received for the test since the display was last reset.

The colours are based on the relative response time. Long response times are more red, short response times are more yellow. This acts as an eye-catcher, allowing expensive tests to be easily spotted.

2) **Results tab:** The Results tab shows the results from The Grinder instrumentation. The result tab generates data or results for a load test on the basis of various parameter which are explained below:

Test	The test number as specified in the test script, eg. tests[14000] will display as Test 14000.
Description	The test description as specified in the test script. If the HTTPProxy has been used to generate the scripts the description field can be found in the httpscript_tests.py file, eg. tests[14000] = Test(14000, 'GET index.jsp').wrap(request14000) will display as 'Get index.jsp'.
Successful Tests	The total number of iterations of the test that were successfully executed by The Grinder during the test run.
Errors	The total number of iterations of the test that failed to be fully executed by The Grinder during the test run.
Mean Time	The mean time taken to execute the test and receive the full response from the target server/application, in milliseconds.
Mean Time Standard Deviation	The mean standard deviation of the time taken to execute the test and receive the full response from the target server/application, in Milliseconds.
TPS	Transactions per second. The average number of iterations of the test that successfully ran in a
Peak TPS	Peak Transactions per second. The maximum number of iterations of the test that successfully ran in a one second interval.

- 3) **Processes tab:** This tab displays information about the Agents, their worker processes and associated threads. The table below provides the detail explanation for the components of this tab.

Process	The name of the process. A parent process will take the hostname of the box on which it is running Its child processes take the name of the parent process and add a suffix of "-x" where x is an integer, eg. Myserver-0.
Type	The type of process, eg. Agent or Worker.
State	Information about the state of the process, eg. "Connected" for an agent process and "Running" and "Finished" for a Worker process.

- 4) **Script tab:** This tab contains the console support for script editing and distribution. The distribution controls are also accessible through the Distribute menu.

To use the script distribution, follow these steps:

1. Set the directory for the script distribution

2. Create a script and a property file
3. Select the properties file to use
4. Distribute the changed files to the agents
5. Start the Worker processes

All this above steps will be explained shortly. Now we have described the basics of grinder-3.4. So from the next chapter we will show that how we can perform the test using the tool on a HTTPS web application.

4 Application to be Tested

Hypertext Transfer Protocol Secure (HTTPS) is a combination of the Hypertext Transfer Protocol with the SSL/TLS protocol to provide encryption and secure identification of the server. HTTPS connections are often used for payment transactions on the World Wide Web and for sensitive transactions in corporate information systems.

As our goal is to perform functionality/load testing of a HTTPS web application, hence before starting the procedure to conduct the test it is important to check whether we are able to access the application from our web browser or not. Grinder-3.4 can perform the load test or functionality test on any web application only when if the web application is accessible from the web browser.

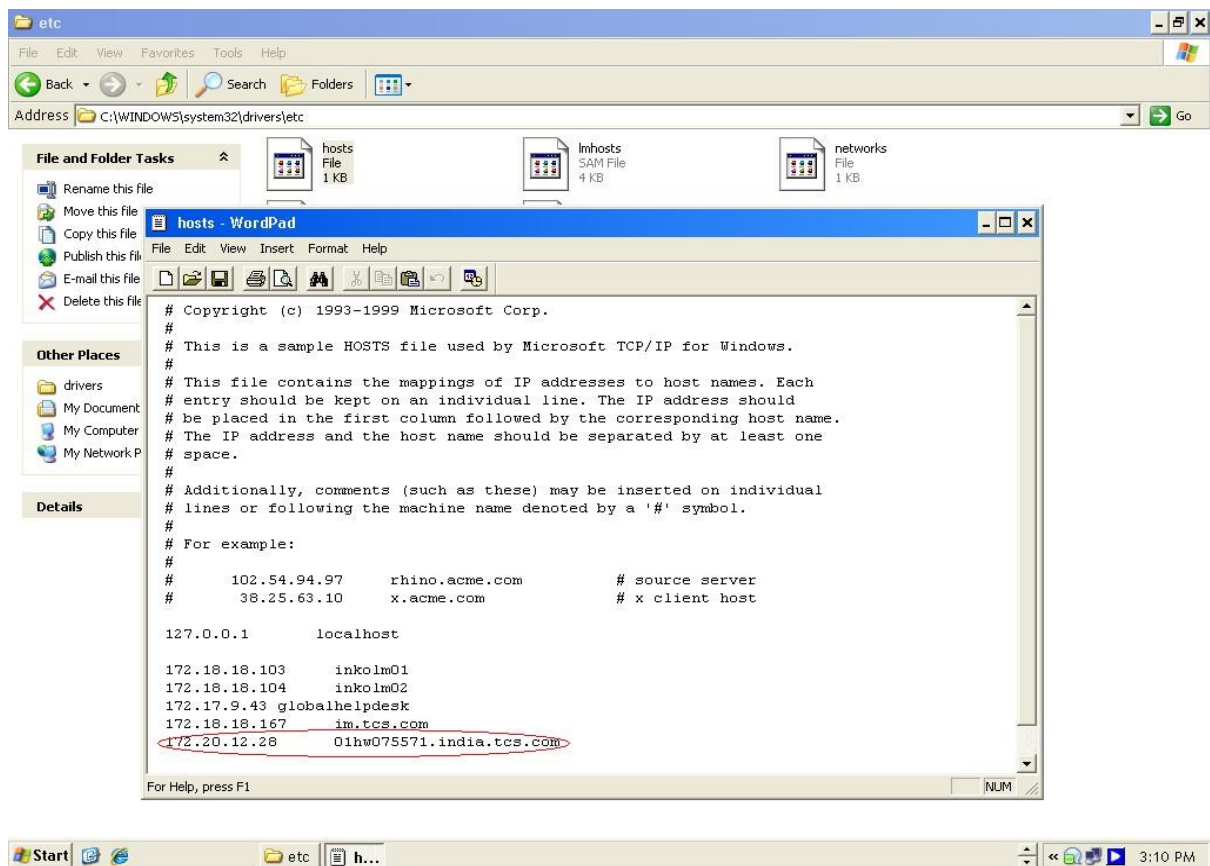
We will be using our eMunicipality HTTPS application as a demo to conduct the test with Grinder.

4.1 Steps to make the web application accessible in the browser

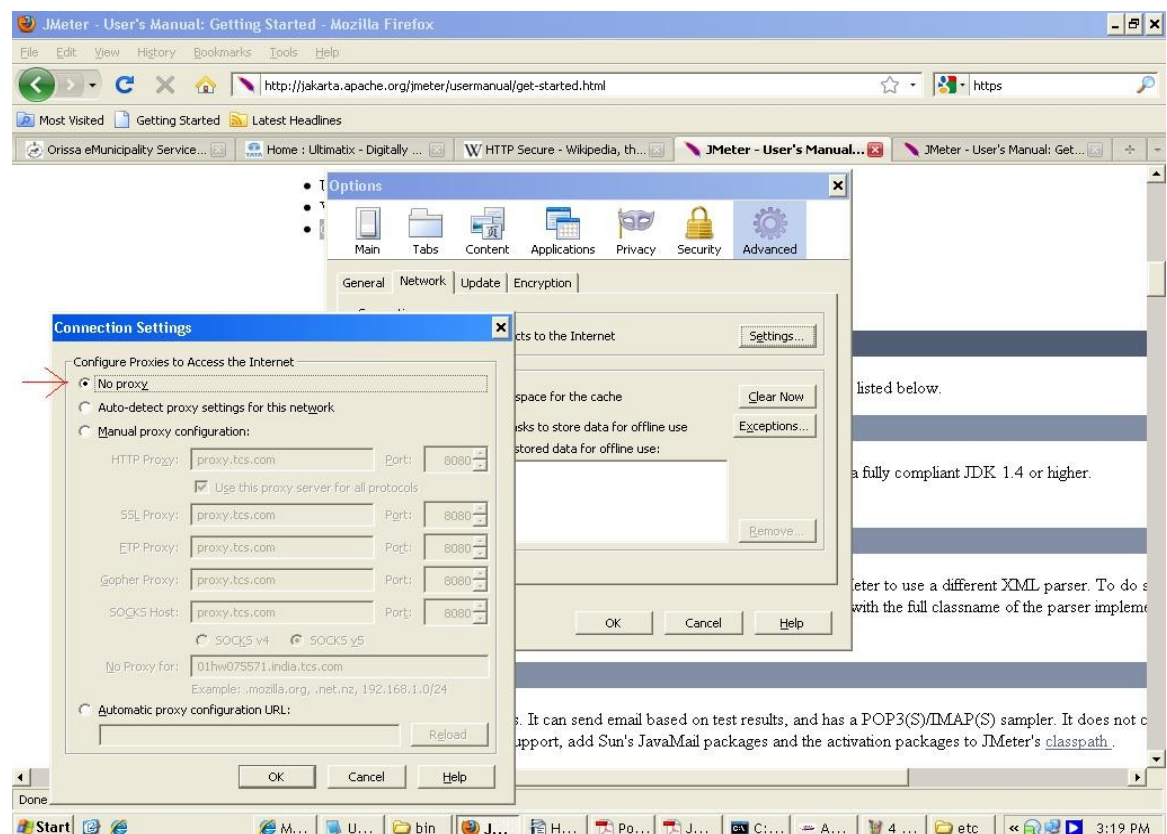
STEP 1: Click START--> RUN and type “Drivers” in the provided text box followed by the enter key for a windows system.



STEP 2: Open the file with name “**hosts**” present in the following path “**drivers\etc**” in any text editor and add the server IP address as well as name in the file as shown in the figure below and Save the file successfully after addition.



STEP 3: Open the web browser either IE or Mozilla Firefox and change the network setting as to use NO PROXY and then run the web application by entering the URL



in the browser. You will see the HTTPS application being accessible in the browser as shown in the figure below.



Now as we are able to run the application to be tested as well as the tool which will be performing the testing operation,so we are all set to start the test.

5 Functionality Testing of the Web Application

First we need to determine the test cases in order to perform the functionality testing of the web application. We are planning to test the following Key scenarios:

- **Home Page-** Home Page of Orissa Municipality application.
- **Employee Login** – Only registered Employee of Orissa Municipality can login.
- **Trade License-** Tab handles license related activities.
- **Entry Notice Details-** A new notice detail is added in the Database.
- **Search Notice-** Newly added Notice record is searched by entering the Notice related details.
- **Logout-** Employee Logout.

The following figures display all the above process to be tested using Grinder-3.4.


Figure Below Display Home Page by Entering the URL




—> “Employee Login” button is clicked on the top right as shown in the above picture.

Orissa eMunicipality System - Mozilla Firefox

tcs.com https://01hw075571.india.tcs.com:9543/casserver/login?service=https%3A%2F%2F01hw075571.india.tcs.com%3A10443%2Femunicipality%2FemployeeLogin.do


An Initiative of Housing And Urban Development Ministry, Orissa.



Enter your User Name and Password

User Name:

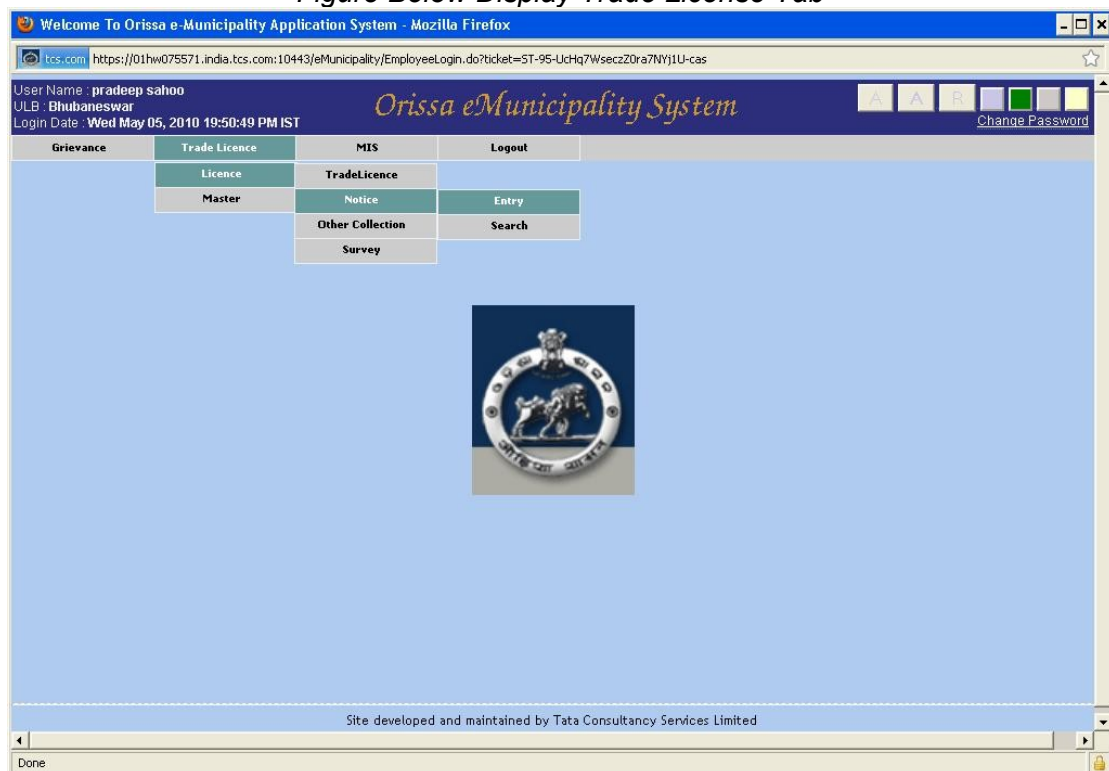
Password:

Site best viewed in 1024x768 resolution

[Link to eMunicipality Portal.](#)

Done

Figure Below Display Trade License Tab



SEF

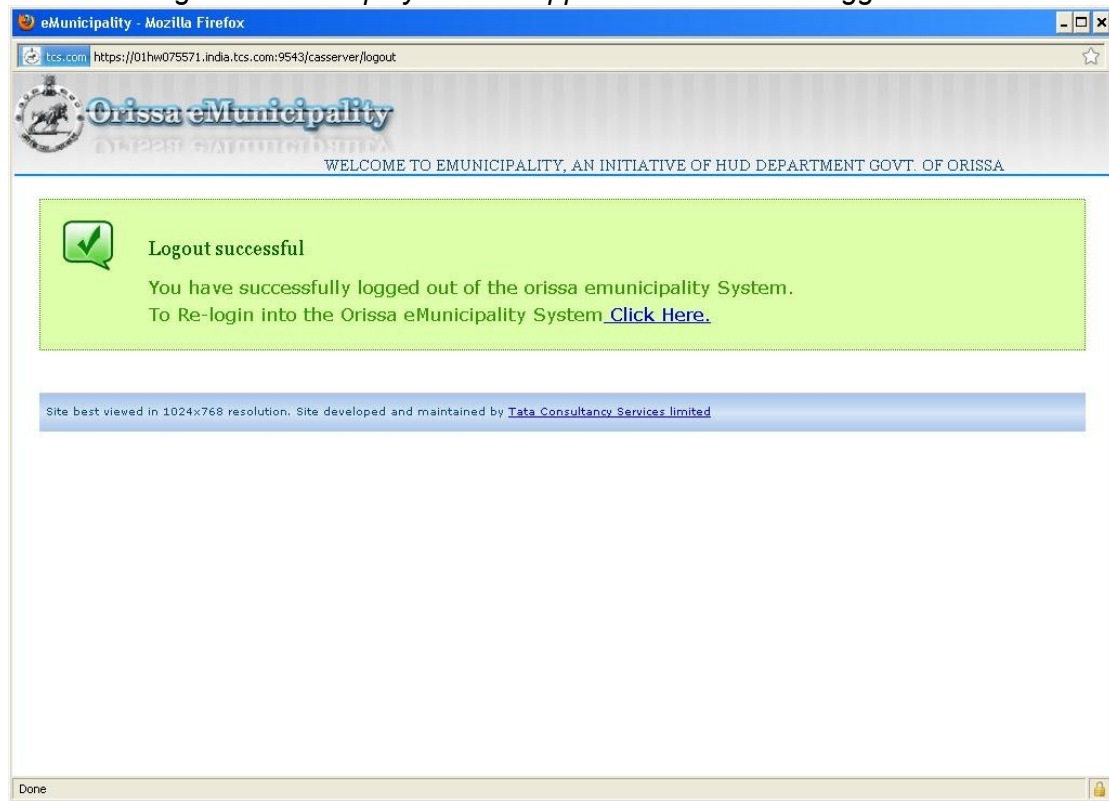
Figure Below display the New Notice details are added

→ “Your Notice has been saved Successfully.” message will be used to check whether the new hospital details are successfully getting added or not in the Application by using Response HTML file in Grinder-3.4 .

Figure Below display the Notice Search Details

→ “one item found “ message will be used to check that the record have been found as an assertion message in Grinder-3.4.

Figure below display that the application has been Logged Out.



Now you must be clear with the idea as what we are going to do using Grinder-3.4 performance testing tool. Main purpose of using this tool is that ,by using this tool we can automate the above process and we can run the test plan for 'n' number of user. Once our Grinder test script is ready for the above test plan, we can run the test and check the performance of the application as well as the server for the specified number of user and the working of the application can be guaranteed.

A Performance testing process can be divided into various task like:

- a) Script recording
- b) Script Editing
- c) Scenario creation/parametrization
- d) Execution
- e) Monitoring
- f) Result Analysis

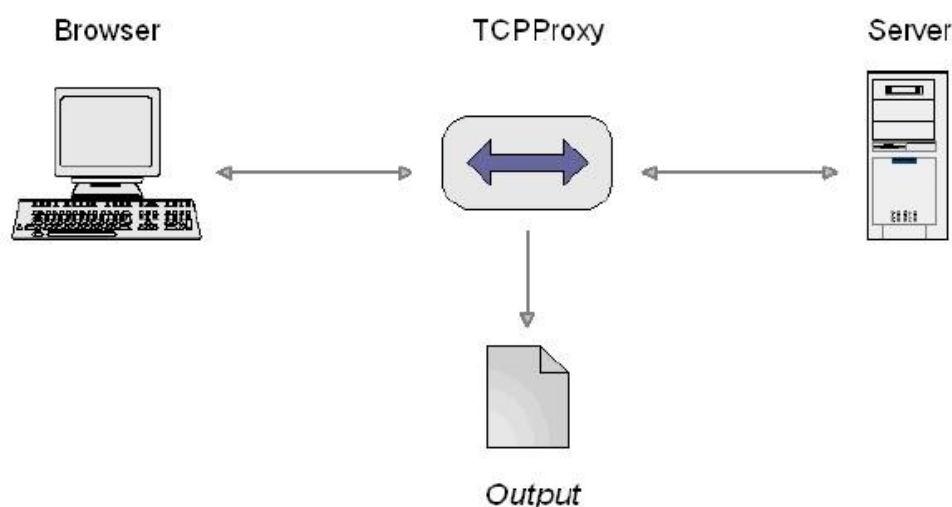
So lets start the Script Recording process.

5.1 Recording HTTPS Requests

A fast way to capture the HTTPS pages of this application is to record every request made to the server. Now before we start the recording to generate the script for the above mentioned test plan, let us understand the role of TCPProxy.

5.1.1 The TCPProxy

The TCPProxy is a proxy process that you can place in a TCP stream, such as the HTTP connection between your browser and a server. It filters the request and response streams, sending the results to the terminal window (stdout). You can control its behaviour by specifying different filters.



The TCPProxy's main purpose is to automatically generate HTTP test scripts that can be replayed with The Grinder's HTTP plug-in. Because the TCPProxy lets you see what's going on at a network level it is also very useful as a debugging tool in its own right.

5.1.2 How to start the TCPProxy?

The TCPProxy can be started by creating a batch file called "startProxy.bat". This batch file can be placed in the grinder-3.4 folder, along with other batch files created earlier. You can use the TCPProxy to generate an HTTP script suitable for use with

the Grinder. The Grinder provides a pair of HTTP filters for this purpose. These filters are enabled by the -http command line option. The content of the batch file includes:

```
call "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\setGrinderEnv.bat"
```

```
java -cp %CLASSPATH% net.grinder.TCPProxy -console -http > fresh.py
```

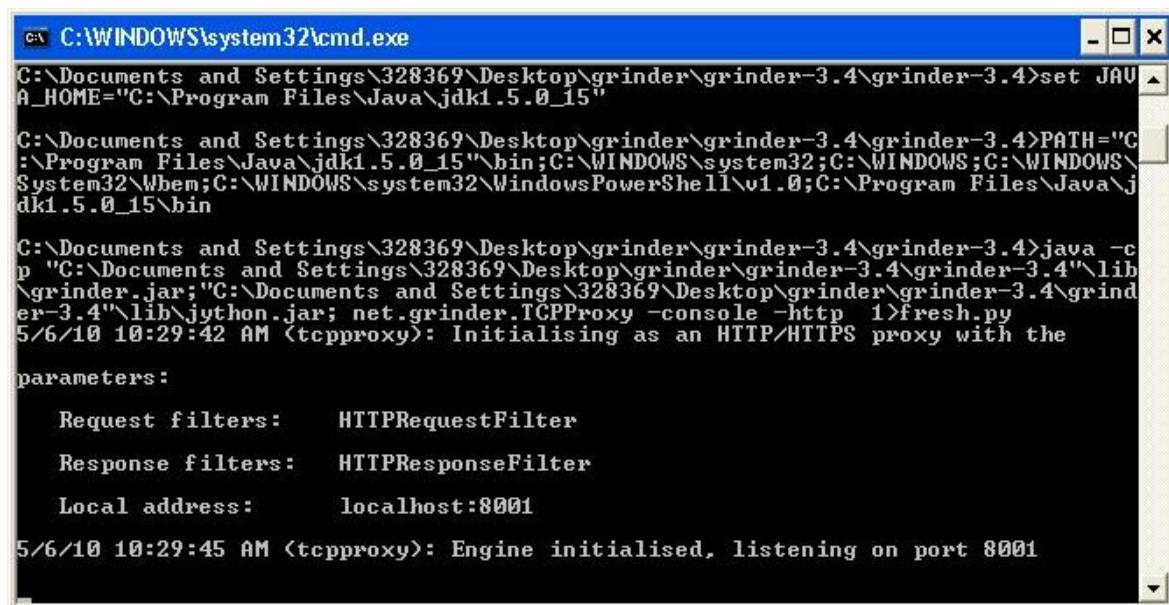
Explanation: The first command in this batch file : *call "provide the path for 'setGrinderEnv.bat' file on your system."*

The second command helps in starting the TCPProxy with an HTTP filter.

```
java -cp %CLASSPATH% net.grinder.TCPProxy -console -http > fresh.py
```

The >fresh.py part of the line sends the script to a file called "fresh.py".

Now the TCPProxy can be started simply by executing the "startProxy.bat" file. The figure below displays the TCPProxy in execution.



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>set JAVA_HOME="C:\Program Files\Java\jdk1.5.0_15"
C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>PATH="C:\Program Files\Java\jdk1.5.0_15\bin;C:\WINDOWS\system32;C:\WINDOWS\System32\Wbem;C:\WINDOWS\system32\WindowsPowerShell\v1.0;C:\Program Files\Java\jdk1.5.0_15\bin"
C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>java -cp "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar; net.grinder.TCPProxy -console -http 1>fresh.py
5/6/10 10:29:42 AM (tcp-proxy): Initialising as an HTTP/HTTPS proxy with the
parameters:
    Request filters:    HTTPRequestFilter
    Response filters:  HTTPResponseFilter
    Local address:     localhost:8001
5/6/10 10:29:45 AM (tcp-proxy): Engine initialised, listening on port 8001
```

The console (initiated by -console) displays a simple control window that allows the TCPProxy to be shut down cleanly. This is needed because some terminal shells, e.g. Cygwin bash, do not allow Java processes to be interrupted cleanly, so filters cannot rely on standard shut down hooks. The console also allows a user to add ad-hoc commentary to the script during the recording. The console looks like this:



Having finished your run through, press "Stop" on the TCPProxy console and the generated script will be written to fresh.py file as specified in the "startProxy.bat" file.

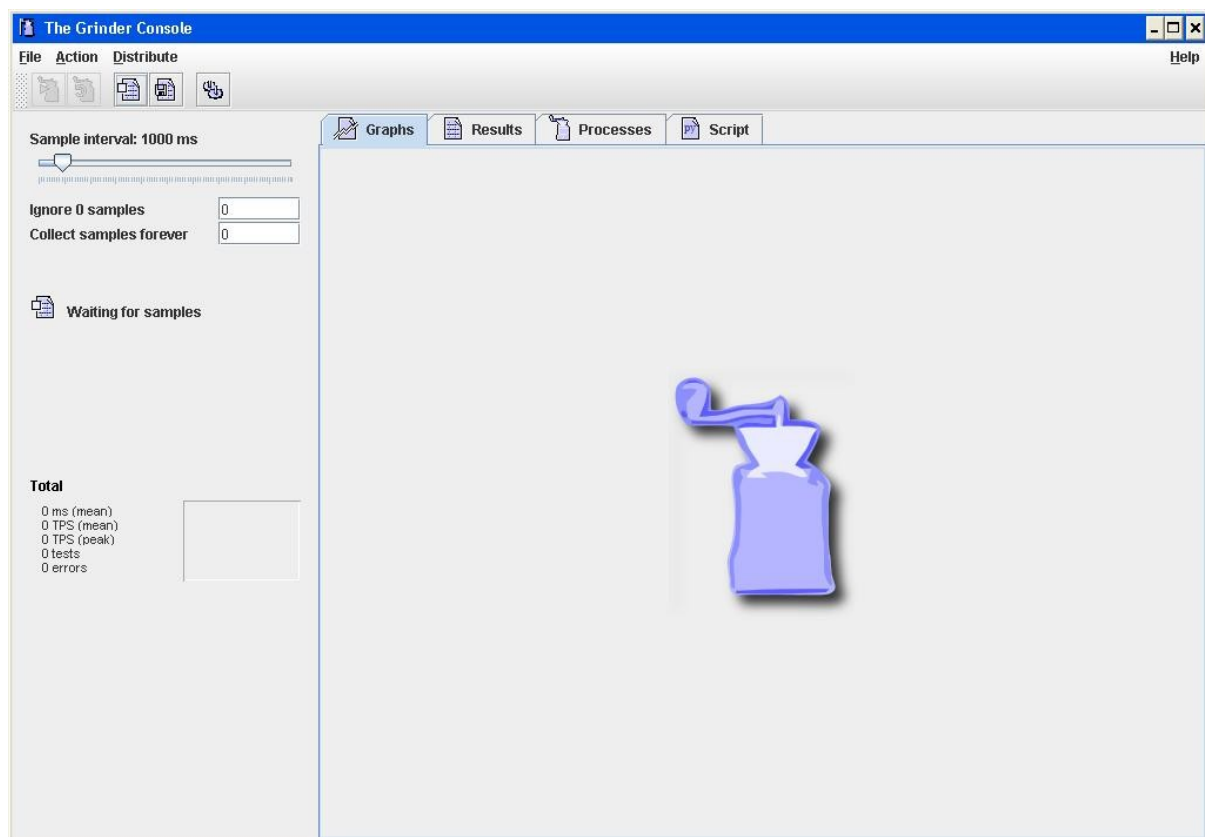
The grinder.py file contains headers, requests and a logical grouping of requests into pages, of the recorded tests.

Now we are ready to record our transaction to generate the script.

5.1.3 Steps for recording HTTPS request

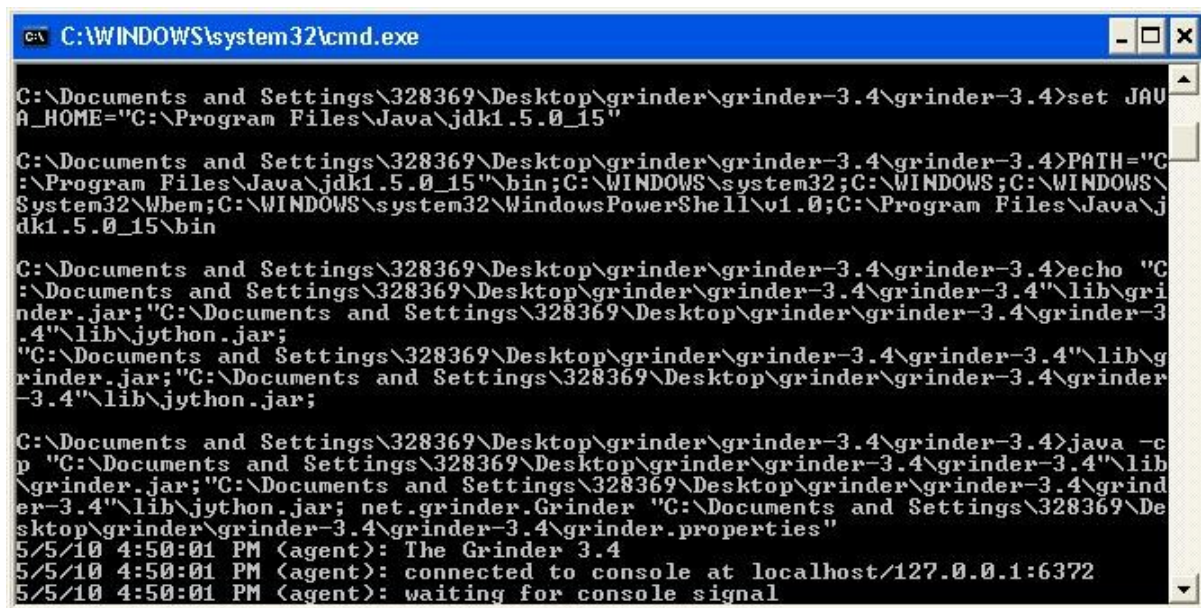
Step1: Close any other application in execution and after that, double click the "setGrinderEnv.bat" file to set the environment for the grinder-3.4.

Step2: Execute the Grinder-3.4 console by double clicking the "startConsole.bat" file. Result of this step is shown in the figure below.



You can check the "start process" and "Reset Process" controls are disabled because no agents are connected to the console. You can check whether any agents are connected on the Processes tab also.

Step3: Start the agent by double clicking the "startAgent.bat" file. The figure below display the Agent in execution.



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>set JAVA_HOME="C:\Program Files\Java\jdk1.5.0_15"

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>PATH="C:\Program Files\Java\jdk1.5.0_15\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\system32\WindowsPowerShell\v1.0;C:\Program Files\Java\jdk1.5.0_15\bin"

C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>echo "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar;"

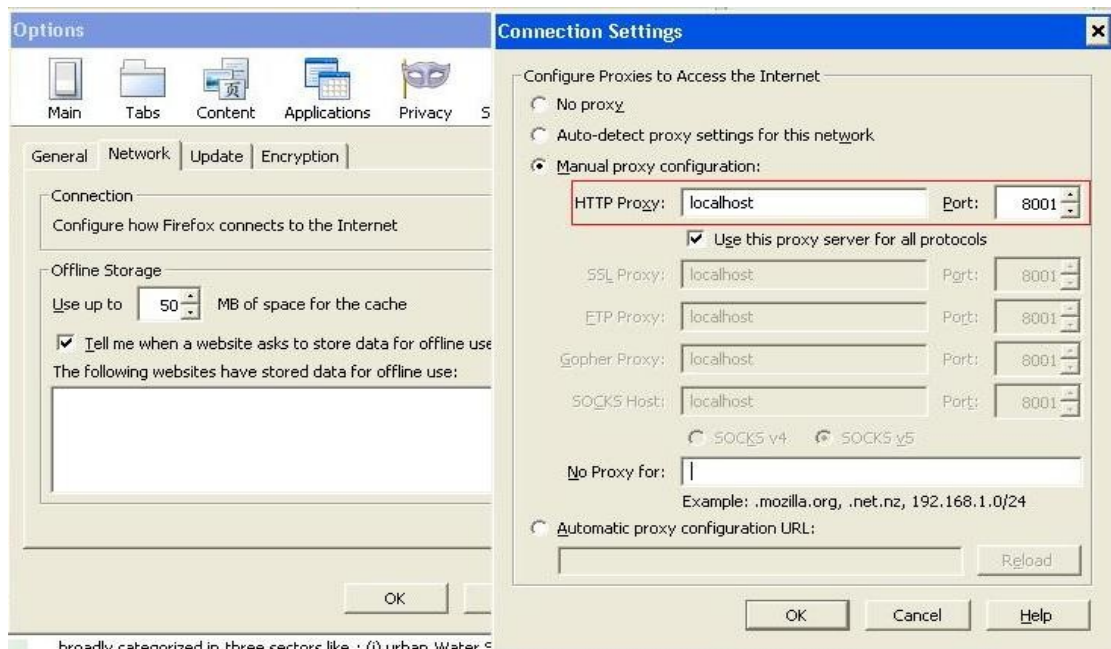
C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4>java -cp "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\grinder.jar;C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\lib\jython.jar; net.grinder.Grinder "C:\Documents and Settings\328369\Desktop\grinder\grinder-3.4\grinder-3.4\grinder.properties"
5/5/10 4:50:01 PM (agent): The Grinder 3.4
5/5/10 4:50:01 PM (agent): connected to console at localhost/127.0.0.1:6372
5/5/10 4:50:01 PM (agent): waiting for console signal
```

Step4: Edit the “startProxy.bat “ file to place the new script file name as “rajesh1.py”. After editing the file save it and double click to execute the TCPProxy. The following message “Engine initialised, listening on port 8001” ,conveys that the TCPProxy has been successfully binded to port 8001 and now any browser mapped to this port on localhost can access the application via TCPProxy.

Step5:Preparing the Browser: You should now set your browser connection settings to specify the TCPProxy as the HTTP proxy. In the browser options dialog, set the proxy host to be the host on which the TCPProxy is running and proxy port to be 8001.

It is important to note that any browser can be used to record a transaction so that the appropriate script can be generated for that transaction. We will be using mozilla firefox 3.5.9 to record the transaction.

Change the proxy setting of mozilla firefox by traversing to the following window as shown below.



The above window is available in the following path of firefox browser:

Tools--> Options-->Network tab-->settings

It is important to remember to remove any entries in the text box "No Proxy for" settings that you might have so that all the traffic flows through the TCPProxy and can be captured.

It might also be a good idea to clear out any cache/temporary Internet files that might be on your workstation. On the other hand, you might decide not to do this if you want to record a script representing a frequent user to your site who has images or resources in their browser cache.

We will suggest to clear all temporary internet files before recording, so that



the request for all the resources will be captured in the script by the TCPProxy. This will then help us in editing the script properly for correlation.

Step6: After the above step , as the TCPProxy is started and listening to port 8001 and also the mozilla browser is mapped to the port 8001,so we can now enter the URL of our HTTPS web application in the address bar of the browser to access the web application. As we will be accessing the web application using the TCPProxy,so request for each element of the application will be getting written in the script specified in the “startProxy.bat” file.

This script can be replayed later to generate an appropriate load on the server.

Step7: Once the decided transaction is traversed i.e. till logout page of the web application,click stop button of the TCPProxy console,to stop recording and generate the script “rajesh1.py”.

5.2 Description of the Script Generated

The “rajesh1.py” files contain headers, requests and a logical grouping of requests into pages, of the recorded tests.

For example, the headers section includes the following section of the script:

The Grinder 3.4

```
# HTTP script recorded by TCPProxy at May 3, 2010 11:42:23 AM

from net.grinder.script import Test
from net.grinder.script.Grinder import grinder
from net.grinder.plugin.http import HTTPPluginControl,
HTTPRequest
from HTTPClient import NVPair
connectionDefaults = HTTPPluginControl.getConnectionDefaults()
httpUtilities = HTTPPluginControl.getHTTPUtilities()

connectionDefaults.useTransferEncoding = 1

connectionDefaults.useContentEncoding = 1

# To use a proxy server, uncomment the next line and set the
host and port.
# connectionDefaults.setProxyServer("localhost", 8001)

# These definitions at the top level of the file are evaluated
once,
# when the worker process is started.

connectionDefaults.defaultHeaders = \

    [ NVPair('User-Agent', 'Mozilla/5.0 (Windows; U; Windows NT
      5.1; en-US; rv:1.9.1.9) Gecko/20100315 Firefox/3.5.9 (
        .NET CLR 3.5.30729)'),

      NVPair('Accept-Encoding', 'gzip,deflate'),
```

```

    NVPair('Accept-Language', 'en-us,en;q=0.5'),

    NVPair('Accept-Charset', 'ISO-8859-1,utf-8;q=0.7,*;q=0.7'), ]

headers0= \

    [ NVPair('Accept',
        'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'), ]

headers1= \

    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'), ]

headers2= \

    [ NVPair('Accept', 'text/css,*/*;q=0.1'),

        NVPair('Referer',
            'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers3= \

    [ NVPair('Accept', 'text/css,*/*;q=0.1'),

        NVPair('Referer',
            'https://01hw075571.india.tcs.com:9543/emunicipality-theme/css/main.css?browserId=firefox&minifierType=css&t=1271743506000'), ]

headers4= \

    [ NVPair('Accept', '*/*'),

        NVPair('Referer',
            'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers5= \

    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

        NVPair('Referer',
            'https://01hw075571.india.tcs.com:9543/emunicipality-theme/css/custom.css'), ]

headers6= \

    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

        NVPair('Referer',
            'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

```

```

headers7= \

[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/html/portal/css.js
    p?
    browserId=firefox&themeId=emunicipality_WAR_emunicipality
    theme&colorSchemeId=01&minifierType=css&t=1271743514000')
    , ]

headers8= \

[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/emunicipality-
    theme/css/forms.css'), ]

headers9= \

[ NVPair('Accept',
    'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers10= \

[ NVPair('Accept', 'text/css,*/*;q=0.1'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/casserver/login?
    service=https%3A%2F%2F01hw075571.india.tcs.com
    %3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]

headers11= \

[ NVPair('Accept', '*/*'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/casserver/login?
    service=https%3A%2F%2F01hw075571.india.tcs.com
    %3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]

headers12= \

[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

  NVPair('Referer',
    'https://01hw075571.india.tcs.com:9543/casserver/css/cas.
    css'), ]

```

```
headers13= \

    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

      NVPair('Referer',
        'https://01hw075571.india.tcs.com:9543/casserver/login?
        service=https%3A%2F%2F01hw075571.india.tcs.com
        %3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]

#....
```

In the request section, each unique request is captured:

```
url0 = 'https://01hw075571.india.tcs.com:9543'
url1 = 'https://01hw075571.india.tcs.com:10443'

# Create an HTTPRequest for each request, then replace the
# reference to the HTTPRequest with an instrumented version.
# You can access the unadorned instance using
request101.__target__.
# home
request101 = HTTPRequest(url=url0, headers=headers0)
request101 = Test(101, 'GET home').wrap(request101)

request102 = HTTPRequest(url=url0, headers=headers1)
request102 = Test(102, 'GET liferay.ico').wrap(request102)

request201 = HTTPRequest(url=url0, headers=headers2)
request201 = Test(201, 'GET css.jsp').wrap(request201)

request301 = HTTPRequest(url=url0, headers=headers2)
request301 = Test(301, 'GET css.jsp').wrap(request301)

request302 = HTTPRequest(url=url0, headers=headers2)
request302 = Test(302, 'GET main.css').wrap(request302)

request303 = HTTPRequest(url=url0, headers=headers2)
request303 = Test(303, 'GET test.css').wrap(request303)

request304 = HTTPRequest(url=url0, headers=headers3)
request304 = Test(304, 'GET base.css').wrap(request304)

request401 = HTTPRequest(url=url0, headers=headers4)
request401 = Test(401, 'GET barebone.jsp').wrap(request401)

request402 = HTTPRequest(url=url0, headers=headers3)
request402 = Test(402, 'GET custom.css').wrap(request402)

request403 = HTTPRequest(url=url0, headers=headers3)
request403 = Test(403, 'GET menu.css').wrap(request403)

request404 = HTTPRequest(url=url0, headers=headers3)
request404 = Test(404, 'GET application.css').wrap(request404)

request405 = HTTPRequest(url=url0, headers=headers3)
```

```
request405 = Test(405, 'GET layout.css').wrap(request405)
```

```
#....
```

Finally the TestRunner class. This section groups the requests into pages and defines each page as a method, sets sleep interval between requests and provides an instrumented method for the return of data from the tests:

```
class TestRunner:

    """A TestRunner instance is created for each worker
    thread."""

    # A method for each recorded page.
    def page1(self):

        """GET home (requests 101-102)."""

        result = request101.GET('/or/emun/home')

        # 4 different values for token_t found in response; the
        # first matched

        # the last known value of token_t - don't update the
        # variable.

        self.token_jsessionid = \

            httpUtilities.valueFromBodyURI('jsessionid') #
            '0EADDF2F1037F6BBB56B8A668C0F1507'

        # 3 different values for token_p_p_id found in response,
        # using the first one.

        self.token_p_p_id = \

            httpUtilities.valueFromBodyURI('p_p_id') # '31'

        self.token_p_p_lifecycle = \

            httpUtilities.valueFromBodyURI('p_p_lifecycle') # '0'

        # 3 different values for token_p_p_state found in
        # response, using the first one.

        self.token_p_p_state = \

            httpUtilities.valueFromBodyURI('p_p_state') #
            'maximized'

        self.token_p_p_mode = \

            httpUtilities.valueFromBodyURI('p_p_mode') # 'view'

        self.token__31_struts_action = \

            httpUtilities.valueFromBodyURI('_31_struts_action') #
            '/image_gallery/view'

        self.token__31_folderId = \
```

```

        httpUtilities.valueFromBodyURI('_31_folderId') # '36869'
self.token_p_p_col_id = \
    httpUtilities.valueFromBodyURI('p_p_col_id') # 'column-
3'
self.token_p_p_col_pos = \
    httpUtilities.valueFromBodyURI('p_p_col_pos') # '1'
self.token_p_p_col_count = \
    httpUtilities.valueFromBodyURI('p_p_col_count') # '3'
self.token_page = \
    httpUtilities.valueFromBodyURI('page') # '2'
self.token_p_l_id = \
    httpUtilities.valueFromBodyURI('p_l_id') # '11942'
self.token__59_INSTANCE_Riq4_cmd = \
    httpUtilities.valueFromHiddenInput('_59_INSTANCE_Riq4_cm
d') # 'add'

grinder.sleep(250)
request102.GET('/emunicipality-theme/images/liferay.ico')
return result
def page2(self):
#.....
    def __call__(self):
        """This method is called for every run performed by
        the worker thread."""

self.page1()          # GET home (requests 101-102)
self.page2()          # GET css.jsp (request 201)
self.page3()          # GET css.jsp (requests 301-304)
self.page4()          # GET barebone.jsp (requests 401-
413)
self.page5()          # GET company_logo (request 501)
self.page6()          # GET image_gallery (request 601)

```

```

self.page7()          # GET image_gallery (requests 701-
                        702)

#....

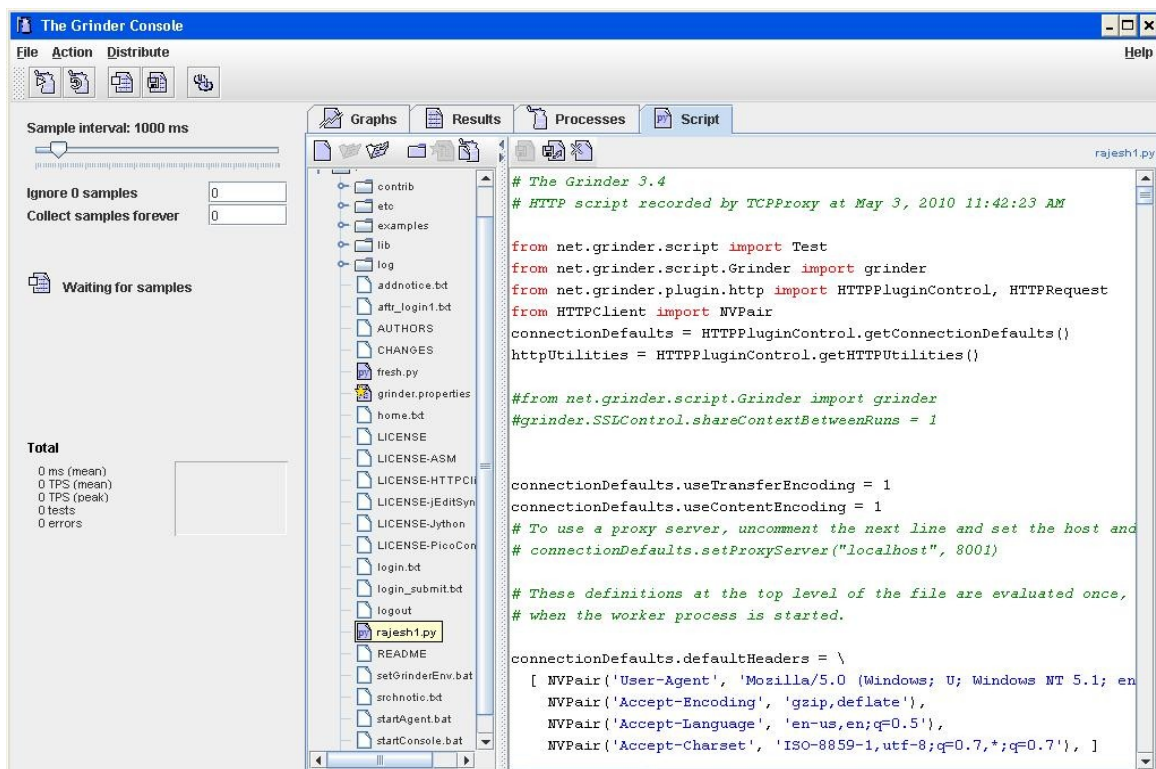
def instrumentMethod(test, method_name,
c=TestRunner):

    """Instrument a method with the given Test."""
    unadorned = getattr(c, method_name)
    import new
    method = new.instancemethod(test.wrap(unadorned),
None, c)
    setattr(c, method_name, method)

# Replace each method with an instrumented version.
# You can call the unadorned method using
self.page1.__target__().

instrumentMethod(Test(100, 'Page 1'), 'page1')
instrumentMethod(Test(200, 'Page 2'), 'page2')
instrumentMethod(Test(300, 'Page 3'), 'page3')
instrumentMethod(Test(400, 'Page 4'), 'page4')
instrumentMethod(Test(500, 'Page 5'), 'page5')

```



The figure above display the script recorded as shown in the Grinder console.

It is important to note that the script recorded contains many hard coded value which could be dynamic in nature ,means these values could be different in each run of the script. Hence we have to generalise the script by performing correlation so that all those dynamic values can be captured when the script will be replayed and will be sent to the server so that the session can be validated.

5.3 Editing script for correlation and Text verification

Since we have successfully recorded the HTTPS Request sequence by the use of TCPProxy of Grinder-3.4 tool ,so now its time to check whether the sequence is properly recorded or not by the help of a mozilla firefox Add-on called “Live HTTP Headers”.

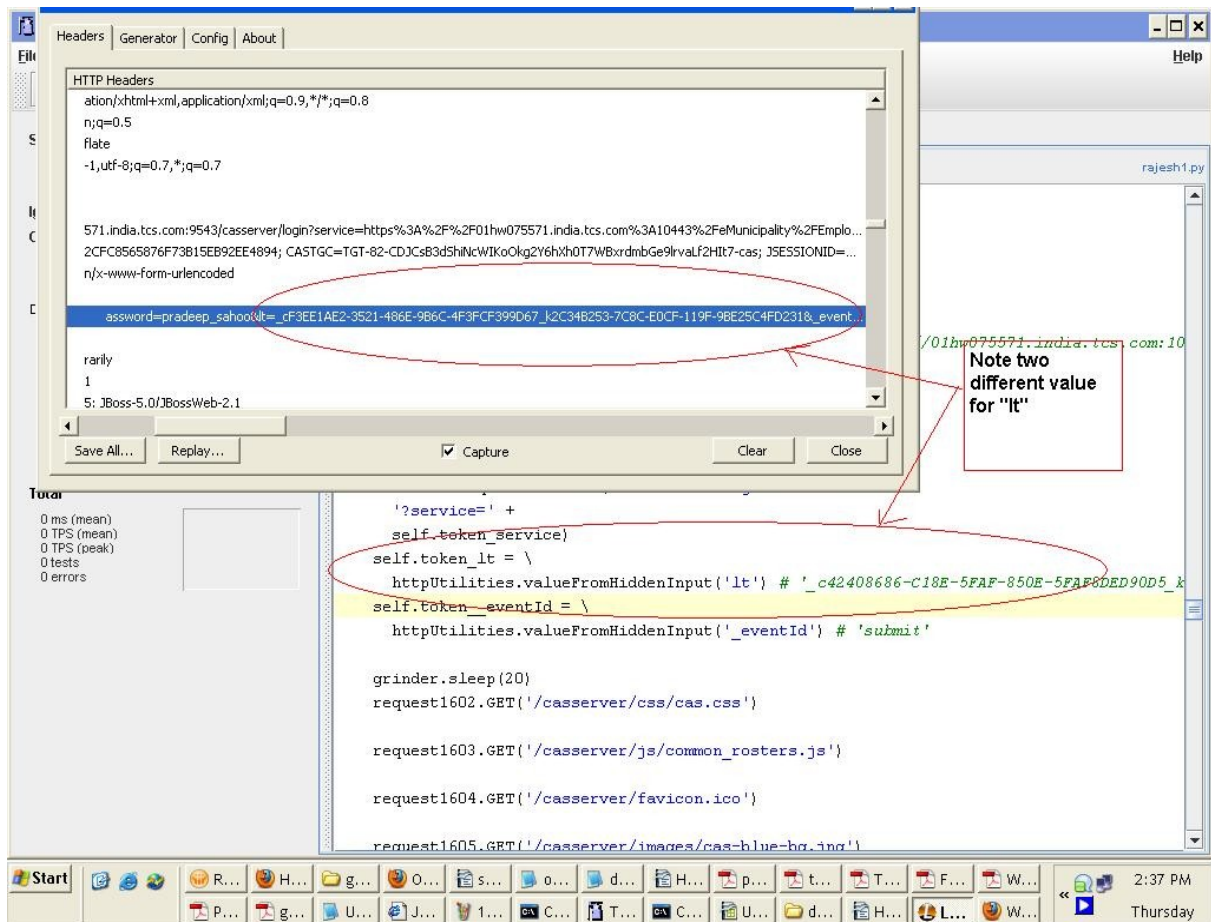
This Add-on can be downloaded and installed in our Firefox browser from the following link : <https://addons.mozilla.org/en-US/firefox/addon/3829>

Our main purpose of using this Add-on will be to check for any dynamic ,hidden data getting generated during the whole process and being sent to the server along with other data. Once we will get to know about this kind of data ,then we will perform correlation for this data in our Test Plan. The purpose of correlation will be to capture this dynamic data generated during the test (runtime) and send that to the server along with the static data as a request.

It is important to perform correlation for all the dynamic data generated during runtime because without this our recorded script will not be able to simulate the expected test plan.

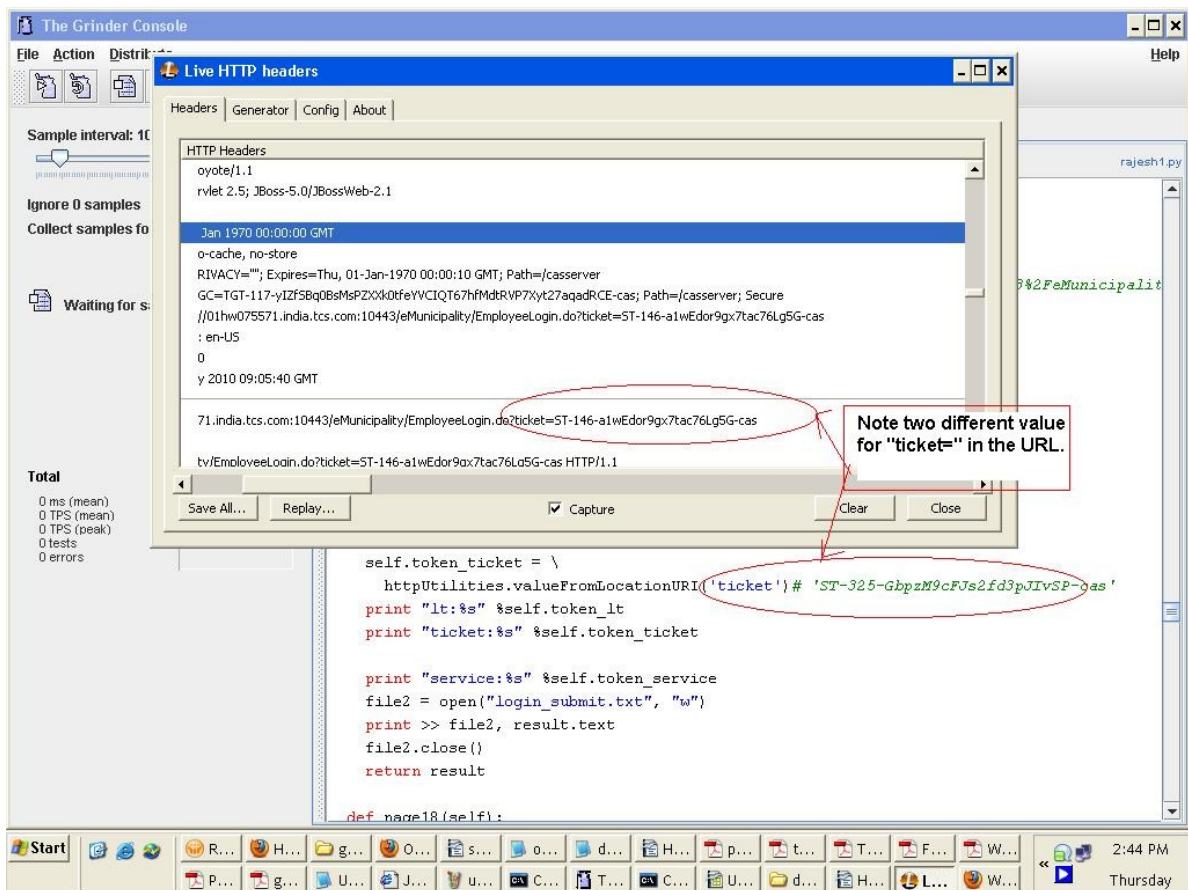
So on comparing the HTTP request sequence recorded in “rajesh1.py” script with the results captured in Live HTTP Header Add-on it was found that a dynamic data called “It” originates in the Employee login form and is sent with the request containing User Name and password to the server. The use of this dynamic data is to authenticate that always a new employee from that system is trying to log onto the server.

(Figure below display need for correlation for the dynamic value called “It”)



The other dynamic data which needs to be correlated include the value for a variable called "ticket=" attached in the URL after successful login into the application. The figure below display the results.

(Figure below display need for correlation for the dynamic value of variable ticket)



5.3.1 Correlation for variable "lt"

The value for the variable is retrieved in the run time using the following command :

```
self.token_lt = httpUtilities.valueFromHiddenInput('lt')
```

The above code return the value for a hidden input token with the given token Name in the body of the last response. If there are multiple matches, the first value is returned.

If there is no match, an empty string is returned rather than `null`. This makes scripts more robust (as they don't need to check the value before using it), but they lose the ability to distinguish between a missing token and an empty value.

Now as the dynamic value is captured in the variable "self.token_lt",so we can then send this value stored in the variable along with the UserName and password to the server. The following code segment explain the process:

```

#...other code..

result = request1601.GET('/casserver/login' +
    '?service=' +
    self.token_service)
self.token_lt = \
    httpUtilities.valueFromHiddenInput('lt')
self.token__eventId = \
    httpUtilities.valueFromHiddenInput('_eventId')
#...other code..
result = request1701.POST('/casserver/login' +
    '?service=' +
    self.token_service,
    ( NVPair('username', 'pradeep_sahoo'),
      NVPair('password', 'pradeep_sahoo'),
      NVPair('lt', self.token_lt), #...Note the hardcoded value is replaced with variable
      NVPair('_eventId', self.token__eventId),
      NVPair('submit', 'LOGIN'), ),
    ( NVPair('Content-Type', 'application/x-www-form-urlencoded'), ))
#...other code..

```

We can also print the value in “self.token_lt” stored at runtime on agent console by using the following command.

```
print "lt:%s" %self.token_lt
```

Note: The value in “ quotes” are simply displayed on the agent console and %s is replaced with the value in the variable.

5.3.2 Correlation for variable “ticket”

The value for the variable “ticket” is retrieved in run time using the following command :

```
self.token_ticket = httpUtilities.valueFromLocationURI('ticket')
```

The HTTPUtilities class can return the value for a path parameter or query string name-value token with the given tokenName in a Location header from the last response. If there are multiple matches, the first value is returned.

If there is no match, an empty string is returned rather than null. This makes scripts more robust (as they don't need to check the value before using it), but they lose the ability to distinguish between a missing token and an empty value.

This must be called from a worker thread, if not it throws a GrinderException.

The following code segment displays the above scenario:

```

#...other code in the script

self.token_ticket = httpUtilities.valueFromLocationURI('ticket')

print "ticket:%s" %self.token_ticket      #..This command display the value stored in
the 'self.token_ticket' variable.

def page18(self):

    """GET EmployeeLogin.do (requests 1801-1822)."""

    result = request1801.GET('/eMunicipality/EmployeeLogin.do' +

        '?ticket=' +

        self.token_ticket) #..hard coded ticket value is replaced by variable self.token_ticket

    #..other code segments..

```

5.3.3 Writing response of each request to a File

In case of Grinder-3.4 the response page of each request is stored in a variable called “result” as shown in the script. So in order to visualize the response for each important request like the request for home page,login etc, we will write the value of variable result to a file. So that we can analyse the response to check whether our script is exactly able to replay the scenario or not.

Once our script will be able to replicate the scenario for one user then we can modify the grinder.properties file to generate appropriate load or simulate appropriate number of user on the server.

Code that is used to write the content of “result” variable or object to a file include:

```

result = request101.GET('/or/emun/home')

file = open("home.txt", "w")

print >> file, result.text

    file.close()

return result

```

The above code segment will write the response of request for home page of our eMunicipality application into the file called “home.txt”.

Always place the above code segment just above the “return result” command as shown. Similarly we can include the same code to write the response for login request,add notice request,search notice request and logout request from the server into specific files like “login.txt”,“aftr_login1.txt”,“addnotice.txt”,“srchnotice.txt” and “logout.txt”

Now with the above correlation, text verification process for our script is complete and we are ready to run the script to replicate a scenario for a single user. Also change the hard coded data for old notice with new data so that when we will run the script then new value of notice will be stored in the application. We can then later manually verify this newly inserted record.

The complete script for the scenario is given in the Appendix section of the document for your further reference.

5.4 Executing the Script

The script is first saved and this changed script is then distributed among the agent process for execution. Select the Distribute/Distribute files menu item, or click on the toolbar button.



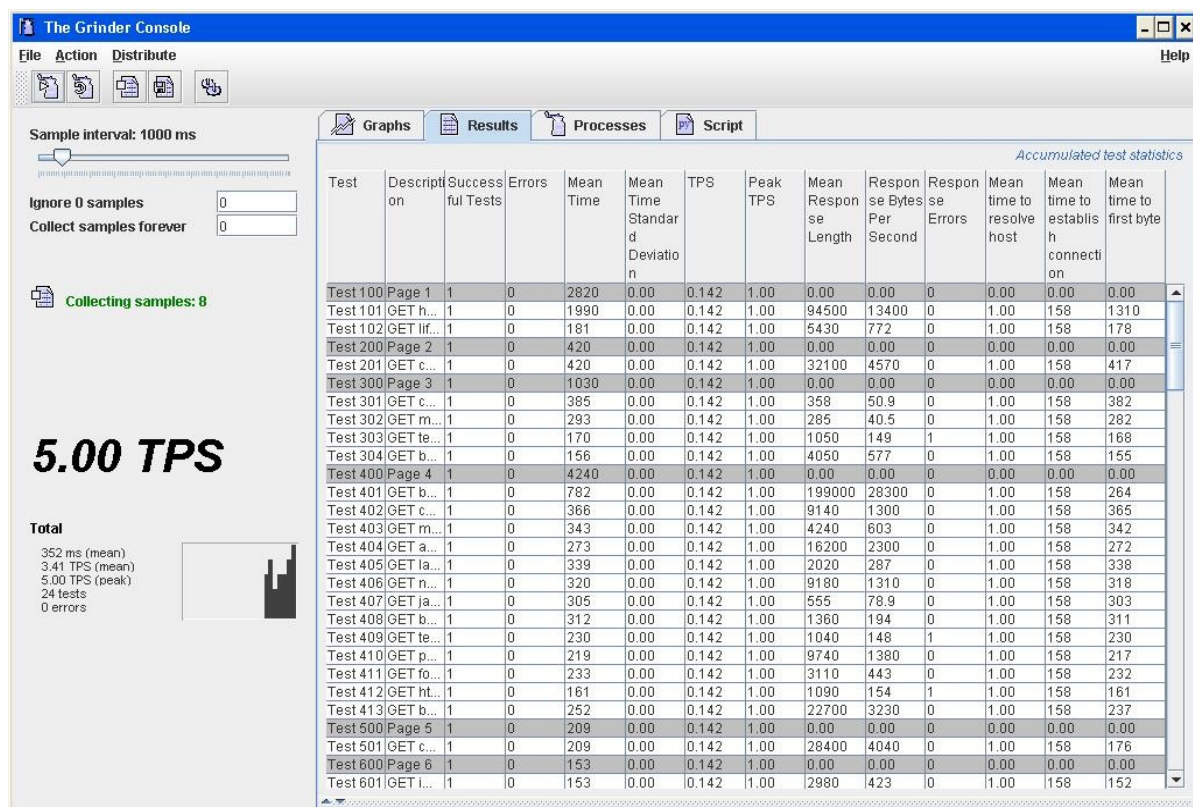
The script can be executed by clicking the replay button as shown in the figure.

Each agent maintains its own local cache of the files below the distribution directory. When you select Distribute files, any files that have changed will be sent to the agents. The distribution controls will only be enabled if one or more agents is connected to the console, and one or more files has been edited.

Now we are ready to run the script for one user as mentioned in the grinder.properties file.



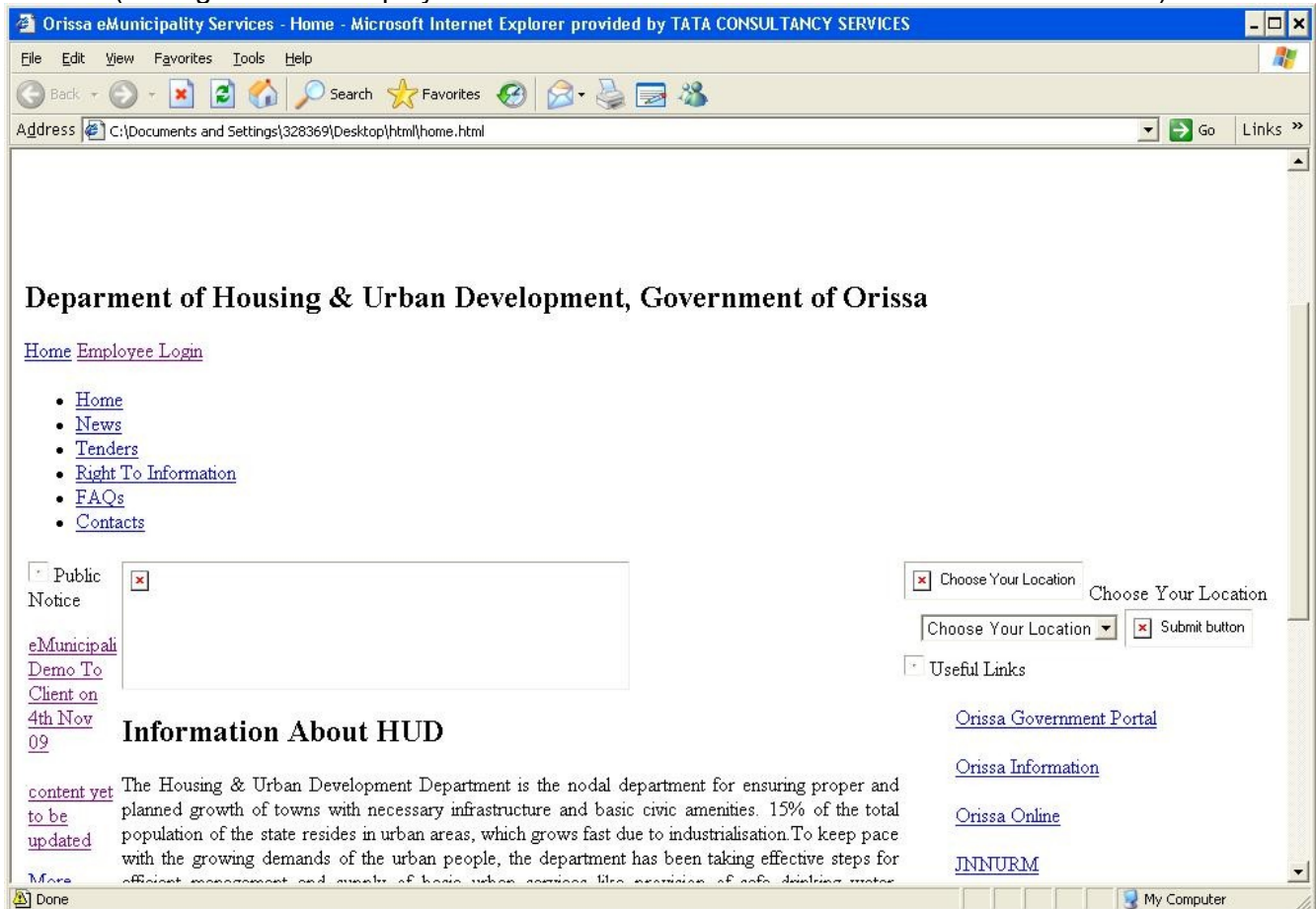
Start the process by clicking the above button as shown in the figure above. You can see the response for the run has been started getting collected in the Grinder-3.4 console as shown in the figure below.



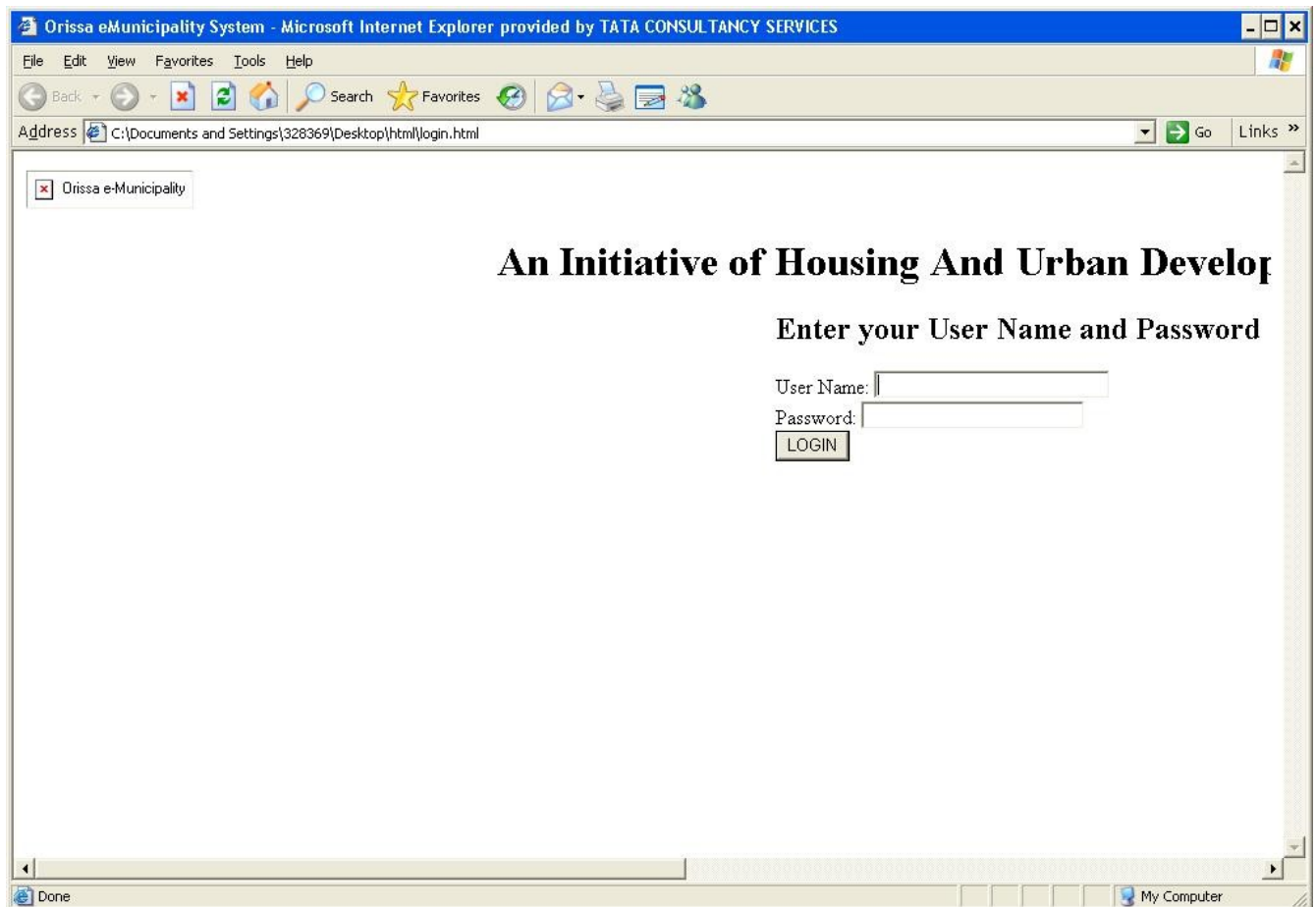
6 Results of the Test

After the execution of the script is done in grinder-3.4, the results are obtained in form of a table for further analysis. Grinder-3.4 also produces a log file which stores the series of requests sent to the server, which can be used to debug the script in case of failure. Also the responses which we have obtained in the .txt file format can be used to verify the working of the script.

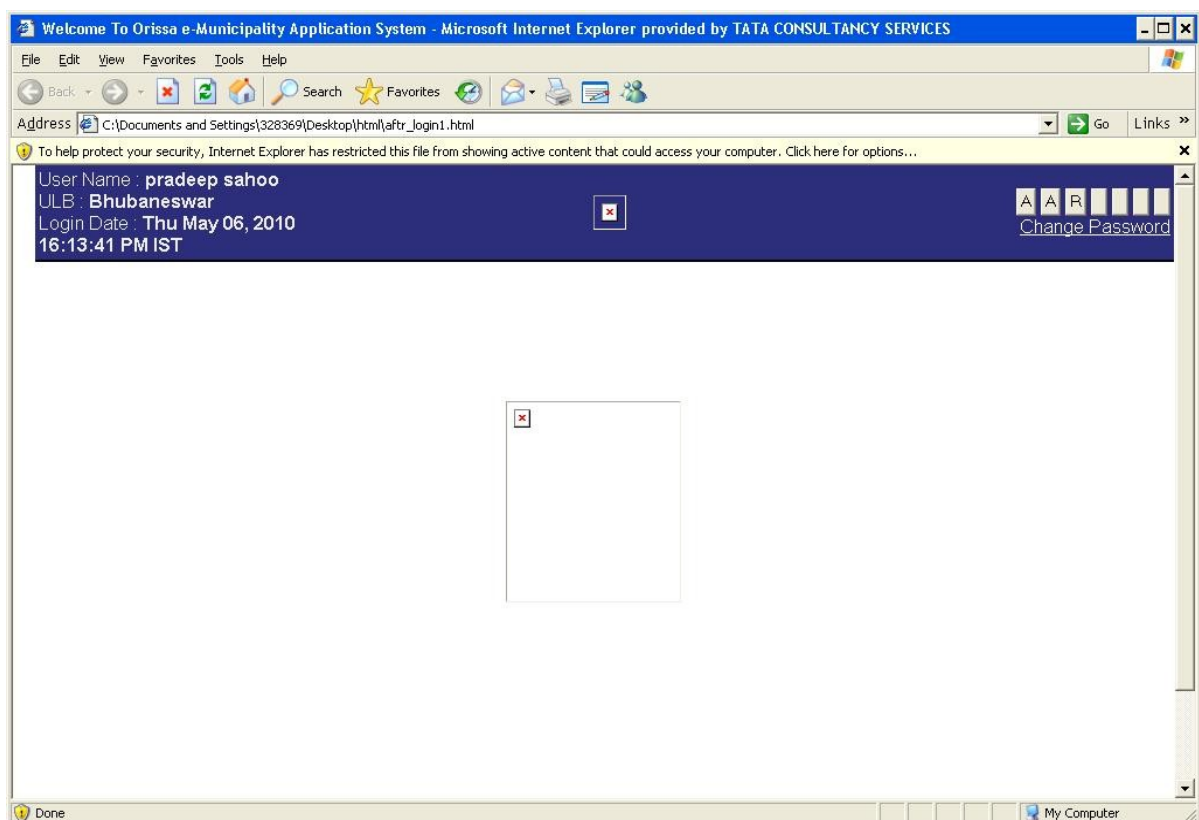
(The figure below display the content of “home.txt” when executed in the browser.)



(The figure below display the content of “login.txt” when executed in the browser.)



(The figure below display the content of "aftr_login1.txt" file which store the result obtained after entering the User Name and password)



(The figure below display the content of “addnotice.txt” which store the result obtained after submitting the add notice detail to the server)

Notice Form - Microsoft Internet Explorer provided by TATA CONSULTANCY SERVICES

User Name : pradeep sahuo
ULB : Bhubaneswar
Login Date : Thu May 06, 2010
16:15:41 PM IST

Notice Form

Notice Details

Misc
Sarkar : pradeep_sahoo Licence Number :

Trader
First Name : raha Trader Middle Name : Trader Last Name :

Trader Agency Name : reja Total Licence Fees : 251.0 Type Of Trade : Saloon

Payment
Due Date : 09/05/2010 Notice Type : First Notice Ward Number : 05

Remark : sas

(The figure below display the content of “srchnotice.txt” which store the result obtained after submitting the search notice detail to the server)

Search Form For Notice - Microsoft Internet Explorer provided by TATA CONSULTANCY SERVICES

User Name : pradeep sahuo
ULB : Bhubaneswar
Login Date : Thu May 06, 2010
16:16:07 PM IST

Search Form For Notice

Search Details

Notice Number : Status : [-ALL-] Ward Number : [-ALL-]
Application No : Payment Entry User : ULB Code : Bhubaneswar
Trader First Name : raha Notice Type : [-ALL-]

Search

One item found. 1

Notice Number	Trader First Name	Total Licence Fees	Notice Type	Misc Sarkar	Notice Status
	raha	251.0	First Notice	pradeep_sahoo	Notice Saved View

Fields marked with an asterisk(*) are required

Site developed and maintained by Tata Consultancy Services Limited

(The figure below display the html code stored in “addnotice.txt “ file which shows the record has been inserted successfully.)

```

oCMenu.makeMenu('login.details.use.case.logout','','Logout','https://01hw075571.india.tcs.com

oCMenu.makeStyle(); oCMenu.construct()
</script>

<br/><br/>

<br>

<div id="messages">

  <div id="s1" class="messages">
    <div class="message">Your Notice has been Saved successfully</div>
  </div>
</div>

<div>

  <h1>Notice Form</h1>
</div>

```

(The figure below display the result table)

Test	Description	Successful Tests	Errors	Mean Time	Mean Time Standard Deviation	TPS	Peak TPS	Mean Response Length	Response Bytes Per Second	Response Errors	Mean time to resolve host	Mean time to establish connection	Mean time to first byte
Test 100	Page 1	1	0	2820	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 101	GET h...	1	0	1990	0.00	0.00317	1.00	94500	301	0	1.00	158	1310
Test 102	GET lif...	1	0	181	0.00	0.00317	1.00	5430	17.3	0	1.00	158	178
Test 200	Page 2	1	0	420	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 201	GET c...	1	0	420	0.00	0.00317	1.00	32100	102	0	1.00	158	417
Test 300	Page 3	1	0	1030	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 301	GET c...	1	0	385	0.00	0.00317	1.00	358	1.14	0	1.00	158	382
Test 302	GET m...	1	0	293	0.00	0.00317	1.00	285	0.907	0	1.00	158	282
Test 303	GET te...	1	0	170	0.00	0.00317	1.00	1050	3.33	1	1.00	158	168
Test 304	GET b...	1	0	156	0.00	0.00317	1.00	4050	12.9	0	1.00	158	155
Test 400	Page 4	1	0	4240	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 401	GET b...	1	0	782	0.00	0.00317	1.00	199000	633	0	1.00	158	264
Test 402	GET c...	1	0	366	0.00	0.00317	1.00	9140	29.1	0	1.00	158	365
Test 403	GET m...	1	0	343	0.00	0.00317	1.00	4240	13.5	0	1.00	158	342
Test 404	GET a...	1	0	273	0.00	0.00317	1.00	16200	51.5	0	1.00	158	272
Test 405	GET la...	1	0	339	0.00	0.00317	1.00	2020	6.42	0	1.00	158	338
Test 406	GET n...	1	0	320	0.00	0.00317	1.00	9180	29.2	0	1.00	158	318
Test 407	GET ja...	1	0	305	0.00	0.00317	1.00	555	1.77	0	1.00	158	303
Test 408	GET b...	1	0	312	0.00	0.00317	1.00	1360	4.34	0	1.00	158	311
Test 409	GET te...	1	0	230	0.00	0.00317	1.00	1040	3.31	1	1.00	158	230
Test 410	GET p...	1	0	219	0.00	0.00317	1.00	9740	31.0	0	1.00	158	217
Test 411	GET fo...	1	0	233	0.00	0.00317	1.00	3110	9.90	0	1.00	158	232
Test 412	GET ht...	1	0	161	0.00	0.00317	1.00	1090	3.45	1	1.00	158	161
Test 413	GET b...	1	0	252	0.00	0.00317	1.00	22700	72.2	0	1.00	158	237
Test 500	Page 5	1	0	209	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 501	GET c...	1	0	209	0.00	0.00317	1.00	28400	90.3	0	1.00	158	176
Test 600	Page 6	1	0	153	0.00	0.00317	1.00	0.00	0.00	0	0.00	0.00	0.00
Test 601	GET l...	1	0	153	0.00	0.00317	1.00	2980	9.47	0	1.00	158	152

(The figure display the message obtained in the agent window)

```
C:\WINDOWS\system32\cmd.exe
ytes)
5/6/10 4:12:14 PM (agent): Updating file store: "log\old log\aftr_login1.txt" (2
bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "log\old log\home.txt" (127090 b
ytes)
5/6/10 4:12:14 PM (agent): Updating file store: "log\old log\login.txt" (3364 by
tes)
5/6/10 4:12:14 PM (agent): Updating file store: "log\old log\login_submit.txt" (
1715 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\charsets.
pkc" (6638 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\dnsns.pkc
" (135 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\grinder-a
gent.pkc" (157 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\grinder.p
kc" (10257 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\jakarta-r
egexp-1.5.pkc" (351 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\jce.pkc"
(940 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\jsse.pkc"
(2737 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\jython.pk
c" (5737 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\localedat
a.pkc" (1018 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\packages.
idx" (2240 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\rt.pkc" (
159658 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\sunjce_pr
ovider.pkc" (1143 bytes)
5/6/10 4:12:14 PM (agent): Updating file store: "lib\cachedir\packages\sunpkcs11
.pkc" (946 bytes)
5/6/10 4:12:31 PM (agent): received a start message
5/6/10 4:12:32 PM (agent): DEBUG MODE: Spawning threads rather than processes
5/6/10 4:12:32 PM (agent): worker localhost-0 started
5/6/10 4:12:36 PM (process localhost-0): starting threads
lt: c354BEED1-5DCC-0485-7ECD-6991594F693D_kB2EB21A5-1617-4113-6924-7F69353A95A7
ticket:ST-161-LcYS5Hm3M06jybs69dB0-cas
service:https://01hw075571.india.tcs.com:10443/eMunicipality/EmployeeLogin.do
5/6/10 4:16:34 PM (process localhost-0): finished
5/6/10 4:16:35 PM (agent): finished, waiting for console signal
```

6.1 Strength and Weakness of Grinder-3.4

Scripting Language:

Latest version, Grinder 3, uses the powerful scripting language Jython, and allows any Java code to be tested without the need to write a plug-in.

Feature pro/cons:

Strengths	Weakness
<ol style="list-style-type: none">1. Grinder is open source software.2. Grinder tests are written in Jython(a Java Python implementation) which is very powerful.3. Grinder can be setup very quickly and is easy to use.4. Grinder have a Java swing user friendly interface	<ol style="list-style-type: none">1. There is no support available for Grinder.2. There is no GUI based interface for defining the scripts3. The reports created by grinder do not contain graphs, chart.4. Monitoring system resource usage on the target system.5. Making each simulated session look like it's coming from a different IP address (important if you have a load balancer).6. Identifying itself as a real browser when contacting the web server, and being able to simulate more than one browser.

7 References

Sr. No	Document Name	Reference
1.	Grinder Portal	http://sourceforge.grinder.net
2.	J2EE Performance Testing by Peter Zadrozny	http://www.books24X7.com

8 Appendix

Script name: rajesh1.py

```
# The Grinder 3.4
# HTTP script recorded by TCPProxy at May 3, 2010 11:42:23 AM

from net.grinder.script import Test
from net.grinder.script.Grinder import grinder
from net.grinder.plugin.http import HTTPPluginControl,
    HTTPRequest
from HTTPClient import NVPair
connectionDefaults = HTTPPluginControl.getConnectionDefaults()
httpUtilities = HTTPPluginControl.getHTTPUtilities()

#from net.grinder.script.Grinder import grinder
#grinder.SSLControl.shareContextBetweenRuns = 1

connectionDefaults.useTransferEncoding = 1
connectionDefaults.useContentEncoding = 1
# To use a proxy server, uncomment the next line and set the
#   host and port.
# connectionDefaults.setProxyServer("localhost", 8001)

# These definitions at the top level of the file are evaluated
#   once,
# when the worker process is started.

connectionDefaults.defaultHeaders = \
    [ NVPair('User-Agent', 'Mozilla/5.0 (Windows; U; Windows NT
      5.1; en-US; rv:1.9.1.9) Gecko/20100315 Firefox/3.5.9 (
        .NET CLR 3.5.30729)'),
      NVPair('Accept-Encoding', 'gzip,deflate'),
      NVPair('Accept-Language', 'en-us,en;q=0.5'),
        NVPair('Accept-Charset', 'ISO-8859-1,utf-
          8;q=0.7,*;q=0.7'), ]

headers0= \
```



```

[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'), ]

headers1= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'), ]

headers2= \
[ NVPair('Accept', 'text/css,*/*;q=0.1'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers3= \
[ NVPair('Accept', 'text/css,*/*;q=0.1'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/emunicipality-
theme/css/main.css?
browserId=firefox&minifierType=css&t=1271743506000'), ]

headers4= \
[ NVPair('Accept', '/*/*'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers5= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/emunicipality-
theme/css/custom.css'), ]

headers6= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]

headers7= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/html/portal/css.js
p?

```

```
browserId=firefox&themeId=emunicipality_WAR_emunicipality
theme&colorSchemeId=01&minifierType=css&t=1271743514000')
, ]
```

```
headers8= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/emunicipality-
theme/css/forms.css'), ]
```

```
headers9= \
                                     [ NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/or/emun/home'), ]
```

```
headers10= \
[ NVPair('Accept', 'text/css,*/*;q=0.1'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/login?
service=https%3A%2F%2F01hw075571.india.tcs.com
%3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]
```

```
headers11= \
[ NVPair('Accept', '/*/*'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/login?
service=https%3A%2F%2F01hw075571.india.tcs.com
%3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]
```

```
headers12= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/css/cas.
css'), ]
```

```
headers13= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                     NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/login?
```

```
service=https%3A%2F%2F01hw075571.india.tcs.com
%3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]
```

```
headers14= \
[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/css/cas.css'), ]
```

```
headers15= \
[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:9543/casserver/login?service=https%3A%2F%2F01hw075571.india.tcs.com%3A10443%2FeMunicipality%2FEmployeeLogin.do'), ]
```

```
headers16= \
[ NVPair('Accept', 'text/css,*/*;q=0.1'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/EmployeeLogin.do?ticket=ST-325-GbpzM9cFJs2fd3pJIvSP-cas'), ]
```

```
headers17= \
[ NVPair('Accept', '*/*'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/EmployeeLogin.do?ticket=ST-325-GbpzM9cFJs2fd3pJIvSP-cas'), ]
```

```
headers18= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/EmployeeLogin.do?ticket=ST-325-GbpzM9cFJs2fd3pJIvSP-cas'), ]
```

```
headers19= \
[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
```

```

                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/Emp
loyeeLogin.do?ticket=ST-325-GbpzM9cFJs2fd3pJIvSP-cas'), ]

headers20= \
[ NVPair('Accept', 'text/css,*/*;q=0.1'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
deLicence/NoticeHomeAddNewNotice.do'), ]

headers21= \
[ NVPair('Accept', '/*/*'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
deLicence/NoticeHomeAddNewNotice.do'), ]

headers22= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
deLicence/NoticeHomeAddNewNotice.do'), ]

headers23= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/lay
out/default-application440.css'), ]

headers24= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/lay
out/default-calendar.css'), ]

headers25= \
                                [
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
                                NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
deLicence/NoticeHomeAddNewNotice.do'), ]

```

```

headers26= \
    [ NVPair('Accept', '/*/*'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeFormSave.do'), ]

headers27= \
    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeFormSave.do'), ]

headers28= \
    [
      NVPair('Accept',
        'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeFormSave.do'), ]

headers29= \
    [ NVPair('Accept', 'text/css,*/*;q=0.1'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeHomeSearch.do'), ]

headers30= \
    [ NVPair('Accept', '/*/*'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeHomeSearch.do'), ]

headers31= \
    [ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),
      NVPair('Referer',
        'https://01hw075571.india.tcs.com:10443/eMunicipality/Tra
        deLicence/NoticeHomeSearch.do'), ]

headers32= \

```

```

[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),

    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/TradeLicence/NoticeHomeSearch.do'), ]

headers33= \
[ NVPair('Accept', '/*/*'),

    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/TradeLicence/SearchFormForNoticeFetchData.do'), ]

headers34= \
[ NVPair('Accept', 'image/png,image/*;q=0.8,*/*;q=0.5'),

    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/TradeLicence/SearchFormForNoticeFetchData.do'), ]

headers35= \
[
    NVPair('Accept',
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),

    NVPair('Referer',
'https://01hw075571.india.tcs.com:10443/eMunicipality/TradeLicence/SearchFormForNoticeFetchData.do'), ]

url0 = 'https://01hw075571.india.tcs.com:9543'
url1 = 'https://01hw075571.india.tcs.com:10443'

# Create an HTTPRequest for each request, then replace the
# reference to the HTTPRequest with an instrumented version.
# You can access the unadorned instance using
request101.__target__.
# home
request101 = HTTPRequest(url=url0, headers=headers0)
request101 = Test(101, 'GET home').wrap(request101)

request102 = HTTPRequest(url=url0, headers=headers1)
request102 = Test(102, 'GET liferay.ico').wrap(request102)

```

```
request201 = HTTPRequest(url=url0, headers=headers2)
request201 = Test(201, 'GET css.jsp').wrap(request201)

request301 = HTTPRequest(url=url0, headers=headers2)
request301 = Test(301, 'GET css.jsp').wrap(request301)

request302 = HTTPRequest(url=url0, headers=headers2)
request302 = Test(302, 'GET main.css').wrap(request302)

request303 = HTTPRequest(url=url0, headers=headers2)
request303 = Test(303, 'GET test.css').wrap(request303)

request304 = HTTPRequest(url=url0, headers=headers3)
request304 = Test(304, 'GET base.css').wrap(request304)

request401 = HTTPRequest(url=url0, headers=headers4)
request401 = Test(401, 'GET barebone.jsp').wrap(request401)

request402 = HTTPRequest(url=url0, headers=headers3)
request402 = Test(402, 'GET custom.css').wrap(request402)

request403 = HTTPRequest(url=url0, headers=headers3)
request403 = Test(403, 'GET menu.css').wrap(request403)

request404 = HTTPRequest(url=url0, headers=headers3)
request404 = Test(404, 'GET application.css').wrap(request404)

request405 = HTTPRequest(url=url0, headers=headers3)
request405 = Test(405, 'GET layout.css').wrap(request405)

request406 = HTTPRequest(url=url0, headers=headers3)
request406 = Test(406, 'GET navigation.css').wrap(request406)

request407 = HTTPRequest(url=url0, headers=headers4)
```



```
request407 = Test(407, 'GET javascript.js').wrap(request407)
```

```
request408 = HTTPRequest(url=url0, headers=headers4)
```

```
request408 = Test(408, 'GET banner.js').wrap(request408)
```

```
request409 = HTTPRequest(url=url0, headers=headers4)
```

```
request409 = Test(409, 'GET test.js').wrap(request409)
```

```
request410 = HTTPRequest(url=url0, headers=headers3)
```

```
request410 = Test(410, 'GET portlet.css').wrap(request410)
```

```
request411 = HTTPRequest(url=url0, headers=headers3)
```

```
request411 = Test(411, 'GET forms.css').wrap(request411)
```

```
request412 = HTTPRequest(url=url0, headers=headers5)
```

```
request412 = Test(412, 'GET html_bg.jpg').wrap(request412)
```

```
request413 = HTTPRequest(url=url0, headers=headers5)
```

```
request413 = Test(413, 'GET banner_bg.jpg').wrap(request413)
```

```
request501 = HTTPRequest(url=url0, headers=headers6)
```

```
request501 = Test(501, 'GET company_logo').wrap(request501)
```

```
request601 = HTTPRequest(url=url0, headers=headers6)
```

```
request601 = Test(601, 'GET image_gallery').wrap(request601)
```

```
request701 = HTTPRequest(url=url0, headers=headers6)
```

```
request701 = Test(701, 'GET image_gallery').wrap(request701)
```

```
request702 = HTTPRequest(url=url0, headers=headers5)
```

```
request702 = Test(702, 'GET menubg.gif').wrap(request702)
```

```
request801 = HTTPRequest(url=url0, headers=headers6)
```

```
request801 = Test(801, 'GET image_gallery').wrap(request801)
```

```

request802 = HTTPRequest(url=url0, headers=headers4)
request802 = Test(802, 'GET test.js').wrap(request802)

request901 = HTTPRequest(url=url0, headers=headers6)
request901 = Test(901, 'GET image_gallery').wrap(request901)

request1001 = HTTPRequest(url=url0, headers=headers6)
request1001 = Test(1001, 'GET image_gallery').wrap(request1001)

request1002 = HTTPRequest(url=url0, headers=headers6)
request1002 = Test(1002, 'GET spacer.png').wrap(request1002)

request1003 = HTTPRequest(url=url0, headers=headers5)
request1003 = Test(1003, 'GET portlet-bg.jpg').wrap(request1003)

request1004 = HTTPRequest(url=url0, headers=headers5)
request1004 = Test(1004, 'GET portlet-header-or.jpg').wrap(request1004)

request1005 = HTTPRequest(url=url0, headers=headers5)
request1005 = Test(1005, 'GET menuarrow.gif').wrap(request1005)

request1101 = HTTPRequest(url=url0, headers=headers6)
request1101 = Test(1101, 'GET image_gallery').wrap(request1101)

request1201 = HTTPRequest(url=url0, headers=headers6)
request1201 = Test(1201, 'GET image_gallery').wrap(request1201)

request1301 = HTTPRequest(url=url0, headers=headers6)
request1301 = Test(1301, 'GET image_gallery').wrap(request1301)

request1401 = HTTPRequest(url=url0, headers=headers6)

```

```

request1401          =          Test(1401,          'GET
    image_gallery').wrap(request1401)

request1402 = HTTPRequest(url=url0, headers=headers7)
request1402          =          Test(1402,          'GET
    paging_next.png').wrap(request1402)

request1403 = HTTPRequest(url=url0, headers=headers6)
request1403 = Test(1403, 'GET icon.png').wrap(request1403)

request1404 = HTTPRequest(url=url0, headers=headers6)
request1404          =          Test(1404,          'GET          go-
    button.gif').wrap(request1404)

request1405 = HTTPRequest(url=url0, headers=headers8)
request1405 = Test(1405, 'GET button.png').wrap(request1405)

request1406 = HTTPRequest(url=url0, headers=headers5)
request1406          =          Test(1406,          'GET          footer-
    bg.jpg').wrap(request1406)

# login
request1501 = HTTPRequest(url=url11, headers=headers9)
request1501          =          Test(1501,          'GET
    eMunicipality').wrap(request1501)

request1502 = HTTPRequest(url=url11, headers=headers9)
request1502 = Test(1502, 'GET /').wrap(request1502)

request1503 = HTTPRequest(url=url11, headers=headers9)
request1503          =          Test(1503,          'GET
    EmployeeLogin.do').wrap(request1503)

request1601 = HTTPRequest(url=url0, headers=headers9)
request1601 = Test(1601, 'GET login').wrap(request1601)

request1602 = HTTPRequest(url=url0, headers=headers10)
request1602 = Test(1602, 'GET cas.css').wrap(request1602)

```

```

request1603 = HTTPRequest(url=url0, headers=headers11)
request1603 = Test(1603, 'GET
common_rosters.js').wrap(request1603)

request1604 = HTTPRequest(url=url0, headers=headers1)
request1604 = Test(1604, 'GET favicon.ico').wrap(request1604)

request1605 = HTTPRequest(url=url0, headers=headers12)
request1605 = Test(1605, 'GET cas-blue-
bg.jpg').wrap(request1605)

request1606 = HTTPRequest(url=url0, headers=headers13)
request1606 = Test(1606, 'GET
banner_header.png').wrap(request1606)

request1607 = HTTPRequest(url=url0, headers=headers12)
request1607 = Test(1607, 'GET key-
point_tl.gif').wrap(request1607)

request1608 = HTTPRequest(url=url0, headers=headers12)
request1608 = Test(1608, 'GET key-
point_tr.gif').wrap(request1608)

request1609 = HTTPRequest(url=url0, headers=headers12)
request1609 = Test(1609, 'GET key-
point_bl.gif').wrap(request1609)

request1610 = HTTPRequest(url=url0, headers=headers12)
request1610 = Test(1610, 'GET key-
point_br.gif').wrap(request1610)

request1611 = HTTPRequest(url=url0, headers=headers12)
request1611 = Test(1611, 'GET cas-footer-
bg1.gif').wrap(request1611)

request1612 = HTTPRequest(url=url0, headers=headers14)
request1612 = Test(1612, 'GET login').wrap(request1612)

```

```

request1701 = HTTPRequest(url=url10, headers=headers15)
request1701 = Test(1701, 'POST login').wrap(request1701)

request1801 = HTTPRequest(url=url11, headers=headers15)
request1801 = Test(1801, 'GET EmployeeLogin.do').wrap(request1801)

request1802 = HTTPRequest(url=url11, headers=headers1)
request1802 = Test(1802, 'GET orissalogo_animated_favicon1.gif').wrap(request1802)

request1803 = HTTPRequest(url=url11, headers=headers16)
request1803 = Test(1803, 'GET default-application440.css').wrap(request1803)

request1804 = HTTPRequest(url=url11, headers=headers17)
request1804 = Test(1804, 'GET dw_cookies.js').wrap(request1804)

request1805 = HTTPRequest(url=url11, headers=headers16)
request1805 = Test(1805, 'GET default.css').wrap(request1805)

request1806 = HTTPRequest(url=url11, headers=headers17)
request1806 = Test(1806, 'GET dw_sizerdx.js').wrap(request1806)

request1807 = HTTPRequest(url=url11, headers=headers17)
request1807 = Test(1807, 'GET layout-common.js').wrap(request1807)

request1808 = HTTPRequest(url=url11, headers=headers17)
request1808 = Test(1808, 'GET key-events.js').wrap(request1808)

request1809 = HTTPRequest(url=url11, headers=headers17)
request1809 = Test(1809, 'GET jquery.js').wrap(request1809)

```

```

request1810 = HTTPRequest(url=url11, headers=headers17)
request1810 = Test(1810, 'GET dw_event.js').wrap(request1810)

request1811 = HTTPRequest(url=url11, headers=headers17)
request1811 = Test(1811, 'GET
    jquery.utils.js').wrap(request1811)

request1812 = HTTPRequest(url=url11, headers=headers17)
request1812 = Test(1812, 'GET
    jquery.eventbinder.js').wrap(request1812)

request1813 = HTTPRequest(url=url11, headers=headers17)
request1813 = Test(1813, 'GET
    jquery.collapsible.js').wrap(request1813)

request1814 = HTTPRequest(url=url11, headers=headers16)
request1814 = Test(1814, 'GET home.css').wrap(request1814)

request1815 = HTTPRequest(url=url11, headers=headers17)
request1815 = Test(1815, 'GET
    coolmenuNew.js').wrap(request1815)

request1816 = HTTPRequest(url=url11, headers=headers17)
request1816 = Test(1816, 'GET
    coolmenus3.js').wrap(request1816)

request1817 = HTTPRequest(url=url11, headers=headers17)
request1817 = Test(1817, 'GET eMun.js').wrap(request1817)

request1818 = HTTPRequest(url=url11, headers=headers17)
request1818 = Test(1818, 'GET hints.js').wrap(request1818)

request1819 = HTTPRequest(url=url11, headers=headers0)
request1819 = Test(1819, 'GET
    orissalogo_favicon.ico').wrap(request1819)

request1820 = HTTPRequest(url=url11, headers=headers18)

```

```

request1820          =          Test(1820,          'GET
    header_name.gif').wrap(request1820)

request1821 = HTTPRequest(url=url1, headers=headers18)
request1821          =          Test(1821,          'GET
    orisslogo.jpg').wrap(request1821)

request1822 = HTTPRequest(url=url1, headers=headers18)
request1822 = Test(1822, 'GET bg_03.gif').wrap(request1822)

# add
request1901 = HTTPRequest(url=url1, headers=headers19)
request1901          =          Test(1901,          'GET
    NoticeHomeAddNewNotice.do').wrap(request1901)

request1902 = HTTPRequest(url=url1, headers=headers20)
request1902          =          Test(1902,          'GET          notice-
    form.css').wrap(request1902)

request1903 = HTTPRequest(url=url1, headers=headers20)
request1903          =          Test(1903,          'GET          default-
    calendar.css').wrap(request1903)

request2001 = HTTPRequest(url=url1, headers=headers21)
request2001 = Test(2001, 'GET calendar.do').wrap(request2001)

request2101 = HTTPRequest(url=url1, headers=headers21)
request2101          =          Test(2101,          'GET
    formValidation.do').wrap(request2101)

request2102 = HTTPRequest(url=url1, headers=headers22)
request2102 = Test(2102, 'GET calendar.gif').wrap(request2102)

request2103 = HTTPRequest(url=url1, headers=headers23)
request2103          =          Test(2103,          'GET          banner-
    emun.jpj.gif').wrap(request2103)

request2104 = HTTPRequest(url=url1, headers=headers22)

```



```

request2104 = Test(2104, 'GET bg_03.gif').wrap(request2104)

request2105 = HTTPRequest(url=url1, headers=headers23)
request2105 = Test(2105, 'GET expanded-
Magenta.gif').wrap(request2105)

request2106 = HTTPRequest(url=url1, headers=headers24)
request2106 = Test(2106, 'GET
menuarrow.gif').wrap(request2106)

request2201 = HTTPRequest(url=url1, headers=headers25)
request2201 = Test(2201, 'POST
NoticeFormSave.do').wrap(request2201)

request2301 = HTTPRequest(url=url1, headers=headers26)
request2301 = Test(2301, 'GET calendar.do').wrap(request2301)

request2401 = HTTPRequest(url=url1, headers=headers26)
request2401 = Test(2401, 'GET
formValidation.do').wrap(request2401)

request2402 = HTTPRequest(url=url1, headers=headers23)
request2402 = Test(2402, 'GET banner-
emun.jpj.gif').wrap(request2402)

request2403 = HTTPRequest(url=url1, headers=headers27)
request2403 = Test(2403, 'GET bg_03.gif').wrap(request2403)

# srch
request2501 = HTTPRequest(url=url1, headers=headers28)
request2501 = Test(2501, 'GET
NoticeHomeSearch.do').wrap(request2501)

request2502 = HTTPRequest(url=url1, headers=headers29)
request2502 = Test(2502, 'GET search-form-for-
notice.css').wrap(request2502)

request2601 = HTTPRequest(url=url1, headers=headers30)

```

```

request2601 = Test(2601, 'GET calendar.do').wrap(request2601)

request2701 = HTTPRequest(url=url1, headers=headers30)
request2701 = Test(2701, 'GET formValidation.do').wrap(request2701)

request2702 = HTTPRequest(url=url1, headers=headers23)
request2702 = Test(2702, 'GET banner-emun.jpg.gif').wrap(request2702)

request2703 = HTTPRequest(url=url1, headers=headers31)
request2703 = Test(2703, 'GET help.gif').wrap(request2703)

request2704 = HTTPRequest(url=url1, headers=headers31)
request2704 = Test(2704, 'GET bg_03.gif').wrap(request2704)

request2801 = HTTPRequest(url=url1, headers=headers32)
request2801 = Test(2801, 'POST SearchFormForNoticeFetchData.do').wrap(request2801)

request2901 = HTTPRequest(url=url1, headers=headers33)
request2901 = Test(2901, 'GET calendar.do').wrap(request2901)

request3001 = HTTPRequest(url=url1, headers=headers33)
request3001 = Test(3001, 'GET formValidation.do').wrap(request3001)

request3002 = HTTPRequest(url=url1, headers=headers23)
request3002 = Test(3002, 'GET banner-emun.jpg.gif').wrap(request3002)

request3003 = HTTPRequest(url=url1, headers=headers23)
request3003 = Test(3003, 'GET none.gif').wrap(request3003)

request3004 = HTTPRequest(url=url1, headers=headers34)
request3004 = Test(3004, 'GET bg_03.gif').wrap(request3004)

```

```

# logout
request3101 = HTTPRequest(url=url1, headers=headers35)
request3101 = Test(3101, 'GET LoginUsecase.do').wrap(request3101)

request3201 = HTTPRequest(url=url0, headers=headers35)
request3201 = Test(3201, 'GET logout').wrap(request3201)

request3202 = HTTPRequest(url=url0, headers=headers12)
request3202 = Test(3202, 'GET confirm.gif').wrap(request3202)

request3203 = HTTPRequest(url=url0, headers=headers12)
request3203 = Test(3203, 'GET cas-footer-bg1.gif').wrap(request3203)

request3204 = HTTPRequest(url=url0, headers=headers14)
request3204 = Test(3204, 'GET login').wrap(request3204)

log =grinder.logger.output

class TestRunner:
    """A TestRunner instance is created for each worker
    thread."""

    # A method for each recorded page.
    def page1(self):
        """GET home (requests 101-102)."""
        result = request101.GET('/or/emun/home')
        # 4 different values for token_t found in response; the
        # first matched
        # the last known value of token_t - don't update the
        # variable.
        self.token_jsessionid = \
            httpUtilities.valueFromBodyURI('jsessionid') #
            '0EADDF2F1037F6BBB56B8A668C0F1507'
        # 3 different values for token_p_p_id found in response,
        # using the first one.
        self.token_p_p_id = \

```

```

    httpUtilities.valueFromBodyURI('p_p_id') # '31'
self.token_p_p_lifecycle = \
    httpUtilities.valueFromBodyURI('p_p_lifecycle') # '0'
    # 3 different values for token_p_p_state found in
    # response, using the first one.
self.token_p_p_state = \
    httpUtilities.valueFromBodyURI('p_p_state') #
    'maximized'
self.token_p_p_mode = \
    httpUtilities.valueFromBodyURI('p_p_mode') # 'view'
self.token__31_struts_action = \
    httpUtilities.valueFromBodyURI('_31_struts_action') #
    '/image_gallery/view'
self.token__31_folderId = \
    httpUtilities.valueFromBodyURI('_31_folderId') # '36869'
self.token_p_p_col_id = \
    httpUtilities.valueFromBodyURI('p_p_col_id') # 'column-
    3'
self.token_p_p_col_pos = \
    httpUtilities.valueFromBodyURI('p_p_col_pos') # '1'
self.token_p_p_col_count = \
    httpUtilities.valueFromBodyURI('p_p_col_count') # '3'
self.token_page = \
    httpUtilities.valueFromBodyURI('page') # '2'
self.token_p_l_id = \
    httpUtilities.valueFromBodyURI('p_l_id') # '11942'
self.token__59_INSTANCE_Riq4_cmd = \
    httpUtilities.valueFromHiddenInput('_59_INSTANCE_Riq4_cm
    d') # 'add'

grinder.sleep(250)
request102.GET('/emunicipality-theme/images/liferay.ico')
file = open("home.txt", "w")
print >> file, result.text
file.close()
return result

```

```

def page2(self):
    """GET css.jsp (request 201)."""
    self.token_browserId = \
        'firefox'
    self.token_themeId = \
        'emunicipality_WAR_emunicipalitytheme'
    self.token_colorSchemeId = \
        '01'
    self.token_minifierType = \
        'css'
    self.token_t = \
        '1271743514000'
    result = request201.GET('/html/portal/css.jsp' +
        '?browserId=' +
        self.token_browserId +
        '&themeId=' +
        self.token_themeId +
        '&colorSchemeId=' +
        self.token_colorSchemeId +
        '&minifierType=' +
        self.token_minifierType +
        '&t=' +
        self.token_t)

    return result

def page3(self):
    """GET css.jsp (requests 301-304)."""
    self.token_t = \
        '1271743515000'

    result =
    request301.GET('/html/portlet/journal_content/css.jsp' +
        '?browserId=' +
        self.token_browserId +
        '&themeId=' +
        self.token_themeId +

```

```

        '&colorSchemeId=' +
        self.token_colorSchemeId +
        '&minifierType=' +
        self.token_minifierType +
        '&t=' +
        self.token_t)

self.token_t = \
    '1271743506000'
request302.GET('/emunicipality-theme/css/main.css' +
    '?browserId=' +
    self.token_browserId +
    '&minifierType=' +
    self.token_minifierType +
    '&t=' +
    self.token_t)

request303.GET('/Ulb_Portlet-5.2.3.1/css/test.css' +
    '?browserId=' +
    self.token_browserId +
    '&minifierType=' +
    self.token_minifierType +
    '&t=' +
    self.token_t)

request304.GET('/emunicipality-theme/css/base.css')

return result

def page4(self):
    """GET barebone.jsp (requests 401-413)."""
    self.token_minifierType = \
        'js'
    self.token_minifierBundleId = \
        'javascript.barebone.files'

```

```

self.token_t = \
    '1271743515000'
result = request401.GET('/html/js/barebone.jsp' +
    '?browserId=' +
    self.token_browserId +
    '&themeId=' +
    self.token_themeId +
    '&colorSchemeId=' +
    self.token_colorSchemeId +
    '&minifierType=' +
    self.token_minifierType +
    '&minifierBundleId=' +
    self.token_minifierBundleId +
    '&t=' +
    self.token_t)

request402.GET('/emunicipality-theme/css/custom.css')

request403.GET('/emunicipality-theme/css/menu.css')

request404.GET('/emunicipality-theme/css/application.css')

request405.GET('/emunicipality-theme/css/layout.css')

request406.GET('/emunicipality-theme/css/navigation.css')

self.token_t = \
    '1271743506000'

request407.GET('/emunicipality-
theme/javascript/javascript.js' +
    '?browserId=' +
    self.token_browserId +
    '&minifierType=' +
    self.token_minifierType +
    '&t=' +
    self.token_t)

```



```

request408.GET('/emunicipality-theme/javascript/banner.js'
+
    '?browserId=' +
    self.token_browserId +
    '&minifierType=' +
    self.token_minifierType +
    '&t=' +
    self.token_t)

request409.GET('/Ulb_Portlet-5.2.3.1/js/test.js' +
    '?browserId=' +
    self.token_browserId +
    '&minifierType=' +
    self.token_minifierType +
    '&t=' +
    self.token_t)

request410.GET('/emunicipality-theme/css/portlet.css')

grinder.sleep(40)
request411.GET('/emunicipality-theme/css/forms.css')

grinder.sleep(70)
                                request412.GET('/emunicipality-
theme/images/custom/html_bg.jpg')

                                request413.GET('/emunicipality-
theme/images/custom/banner_bg.jpg')

return result

def page5(self):
    """GET company_logo (request 501)."""
    self.token_img_id = \
        '13170'

```

```

self.token_t = \
    '1272866943018'
result = request501.GET('/image/company_logo' +
    '?img_id=' +
    self.token_img_id +
    '&t=' +
    self.token_t)

return result

def page6(self):
    """GET image_gallery (request 601)."""
    self.token_img_id = \
        '70606'
    self.token_t = \
        '1266238985620'
    result = request601.GET('/image/image_gallery' +
        '?img_id=' +
        self.token_img_id +
        '&t=' +
        self.token_t)

    return result

def page7(self):
    """GET image_gallery (requests 701-702)."""
    self.token_img_id = \
        '70602'
    self.token_t = \
        '1266238940357'
    result = request701.GET('/image/image_gallery' +
        '?img_id=' +
        self.token_img_id +
        '&t=' +
        self.token_t)

```

```

        request702.GET('/emunicipality-
theme/images/menu/menubg.gif')

    return result

def page8(self):
    """GET image_gallery (requests 801-802)."""
    self.token_uuid = \
        'afd43eac-0cc7-4490-b390-866c8e22dff6'
    self.token_groupId = \
        '10226'
    self.token_t = \
        '1269846501858'
    result = request801.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

    self.token_t = \
        '1271743506000'
    request802.GET('/Ulb_Portlet-5.2.3.1/js/test.js' +
        '?browserId=' +
        self.token_browserId +
        '&minifierType=' +
        self.token_minifierType +
        '&t=' +
        self.token_t)

    return result

def page9(self):
    """GET image_gallery (request 901)."""

```

```

self.token_uuid = \
    'd2bf7370-5cd5-4477-8538-0a6b9cc55f11'
self.token_t = \
    '1267076705665'
result = request901.GET('/image/image_gallery' +
    '?uuid=' +
    self.token_uuid +
    '&groupId=' +
    self.token_groupId +
    '&t=' +
    self.token_t)

return result

def page10(self):
    """GET image_gallery (requests 1001-1005)."""
    self.token_uuid = \
        'fa696e4f-d0bb-47ea-a987-299626994e2d'
    self.token_t = \
        '1267076754309'
    result = request1001.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

    grinder.sleep(30)
    request1002.GET('/emunicipality-theme/images/spacer.png')

    request1003.GET('/emunicipality-
theme/images/custom/portlet-bg.jpg')

    request1004.GET('/emunicipality-
theme/images/custom/portlet-header-or.jpg')

```

```

request1005.GET('/emunicipality-
theme/images/menu/menuarrow.gif')

return result

def page11(self):
    """GET image_gallery (request 1101)."""
    self.token_uuid = \
        '8f09bc77-d26a-4925-87de-ad851bc439df'
    self.token_t = \
        '1269072807139'
    result = request1101.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

    return result

def page12(self):
    """GET image_gallery (request 1201)."""
    self.token_uuid = \
        '1225cc77-f3b7-4f81-85bd-89a4a8455a7c'
    self.token_t = \
        '1268308543316'
    result = request1201.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

```

```

        return result

def page13(self):
    """GET image_gallery (request 1301)."""
    self.token_uuid = \
        '4b89d5d0-3932-4448-b0ec-b85482cd9e73'
    result = request1301.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

    return result

def page14(self):
    """GET image_gallery (requests 1401-1406)."""
    self.token_uuid = \
        '3c6b3c6c-ac37-4d7f-9197-072b40db1d87'
    result = request1401.GET('/image/image_gallery' +
        '?uuid=' +
        self.token_uuid +
        '&groupId=' +
        self.token_groupId +
        '&t=' +
        self.token_t)

    grinder.sleep(20)

    request1402.GET('/emunicipality-
theme/images/arrows/paging_next.png')

    request1403.GET('/Ulb_Portlet-5.2.3.1/icon.png')

    grinder.sleep(30)

```

```

        request1404.GET('/Ulb_Portlet-5.2.3.1/html/image/go-
button.gif')

        request1405.GET('/emunicipality-
theme/images/forms/button.png')

        request1406.GET('/emunicipality-
theme/images/custom/footer-bg.jpg')

    return result

def page15(self):
    """GET eMunicipality (requests 1501-1503)."""
    self.token_ulb = \
        'Home'

    # Expecting 302 'Moved Temporarily'
    result = request1501.GET('/eMunicipality' +
        '?ulb=' +
        self.token_ulb)

    # Expecting 302 'Moved Temporarily'
    request1502.GET('/eMunicipality/' +
        '?ulb=' +
        self.token_ulb)

    # Expecting 302 'Moved Temporarily'
    request1503.GET('/eMunicipality/EmployeeLogin.do')
    self.token_service = \
        httpUtilities.valueFromLocationURI('service') #
        'https://01hw075571.india.tcs.com:10443/e...'

    return result

def page16(self):

```



```

"""GET login (requests 1601-1612)."""
result = request1601.GET('/casserver/login' +
    '?service=' +
    self.token_service)
self.token_lt = \
    httpUtilities.valueFromHiddenInput('lt') # '_c42408686-
    C18E-5FAF-850E-5FAF8DED90D5_k...'
self.token__eventId = \
    httpUtilities.valueFromHiddenInput('_eventId') #
    'submit'

grinder.sleep(20)
request1602.GET('/casserver/css/cas.css')

request1603.GET('/casserver/js/common_rosters.js')

request1604.GET('/casserver/favicon.ico')

request1605.GET('/casserver/images/cas-blue-bg.jpg')

request1606.GET('/casserver/images/banner_header.png')

request1607.GET('/casserver/images/key-point_tl.gif')

request1608.GET('/casserver/images/key-point_tr.gif')

request1609.GET('/casserver/images/key-point_bl.gif')

grinder.sleep(20)
request1610.GET('/casserver/images/key-point_br.gif')

grinder.sleep(110)

# Expecting 302 'Moved Temporarily'
request1611.GET('/casserver/images/cas-footer-bg1.gif')

```

```

request1612.GET('/casserver/login' +
    '?null')
#self.token_lt = \
    #httpUtilities.valueFromHiddenInput('lt') # '_c92359662-
    891C-D3D4-2F66-DB706767FC3D_k...'
file1 = open("login.txt", "w")
print >> file1, result.text
file1.close()
return result

def page17(self):
    """POST login (request 1701)."""
    #self.token_lt = \
        #httpUtilities.valueFromHiddenInput('lt')

        #self.token_services1 = 'https%3A%2F
        %2F01hw075571.india.tcs.com%3A10443%2FeMunicipality
        %2FEmployeeLogin.do'
    # Expecting 302 'Moved Temporarily'
    result = request1701.POST('/casserver/login' +
        '?service=' +
        self.token_service,
        ( NVPair('username', 'pradeep_sahoo'),
          NVPair('password', 'pradeep_sahoo'),
          NVPair('lt', self.token_lt),
          NVPair('_eventId', self.token__eventId),
          NVPair('submit', 'LOGIN'), ),
        ( NVPair('Content-Type', 'application/x-www-form-
        urlencoded'), ))

    self.token_ticket = \
        httpUtilities.valueFromLocationURI('ticket')# 'ST-325-
        GbpzM9cFJs2fd3pJIvSP-cas'
    print "lt:%s" %self.token_lt
    print "ticket:%s" %self.token_ticket

    print "service:%s" %self.token_service

```

```

file2 = open("login_submit.txt", "w")
print >> file2, result.text
file2.close()
return result

def page18(self):
    """GET EmployeeLogin.do (requests 1801-1822)."""
    result = request1801.GET('/eMunicipality/EmployeeLogin.do'
        +
        '?ticket=' +
        self.token_ticket)
    self.token_cp = \
        httpUtilities.valueFromBodyURI('cp') # '1'
    file9 = open("aftr_login1.txt", "w")
    print >> file9, result.text
    file9.close()
    grinder.sleep(211)
    request1802.GET('/eMunicipality/images/orissalogo_animated
        _favicon1.gif')

        request1803.GET('/eMunicipality/layout/default-
        application440.css')

    request1804.GET('/eMunicipality/layout/dw_cookies.js')

    request1805.GET('/eMunicipality/layout/default.css')

    request1806.GET('/eMunicipality/layout/dw_sizerdx.js')

    request1807.GET('/eMunicipality/layout/layout-common.js')

    request1808.GET('/eMunicipality/layout/key-events.js')

    request1809.GET('/eMunicipality/js/jquery.js')

    request1810.GET('/eMunicipality/layout/dw_event.js')

```

```

request1811.GET('/eMunicipality/js/jquery.utils.js')

request1812.GET('/eMunicipality/js/jquery.eventbinder.js')

request1813.GET('/eMunicipality/js/jquery.collapsible.js')

request1814.GET('/eMunicipality/com/tcs/cdr/govt/emun/bd/b
irth/web/BD001BirthRegistrationController/home.css')

request1815.GET('/eMunicipality/layout/coolmenuNew.js')

request1816.GET('/eMunicipality/layout/coolmenus3.js')

self.token_v = \
    '1.1'
request1817.GET('/eMunicipality/js/eMun.js' +
    '?v=' +
    self.token_v)

request1818.GET('/eMunicipality/layout/hints.js')

grinder.sleep(1092)
request1819.GET('/eMunicipality/images/orissalogo_favicon.
ico')

request1820.GET('/eMunicipality/images/header_name.gif')

request1821.GET('/eMunicipality/images/orisslogo.jpg')

request1822.GET('/casimage/bg_03.gif')

return result

def page19(self):
    """GET NoticeHomeAddNewNotice.do (requests 1901-1903)."""

```

```

                                                    result
request1901.GET('/eMunicipality/TradeLicence/NoticeHomeAd
dNewNotice.do')
self.token_licenceNumber = \
    httpUtilities.valueFromHiddenInput('licenceNumber') # ''
self.token_noticeNumber = \
    httpUtilities.valueFromHiddenInput('noticeNumber') # ''
self.token_dateOfIssue = \
    httpUtilities.valueFromHiddenInput('dateOfIssue') # ''
self.token_id = \
    httpUtilities.valueFromHiddenInput('id') # '0'
self.token_applicationStatus = \
    httpUtilities.valueFromHiddenInput('applicationStatus')
    # ''
self.token_licenceStatus = \
    httpUtilities.valueFromHiddenInput('licenceStatus') # ''
self.token_noticeStatus = \
    httpUtilities.valueFromHiddenInput('noticeStatus') # ''
self.token_commComments = \
    httpUtilities.valueFromHiddenInput('commComments') # ''
self.token_commappCommDate = \
    httpUtilities.valueFromHiddenInput('commappCommDate') #
    ''
self.token_miscCloseComments = \
    httpUtilities.valueFromHiddenInput('miscCloseComments')
    # ''
self.token_miscCommDate = \
    httpUtilities.valueFromHiddenInput('miscCommDate') # ''
self.token_miscComments = \
    httpUtilities.valueFromHiddenInput('miscComments') # ''
self.token_miscCommentsDate = \
    httpUtilities.valueFromHiddenInput('miscCommentsDate') #
    ''
self.token_status = \
    httpUtilities.valueFromHiddenInput('status') # ''
self.token_statusCode = \
    httpUtilities.valueFromHiddenInput('statusCode') # ''
self.token_streetName = \

```

```

        httpUtilities.valueFromHiddenInput('streetName') # ''
self.token_section = \
    httpUtilities.valueFromHiddenInput('section') # ''
self.token_ulbCode = \
    httpUtilities.valueFromHiddenInput('ulbCode') # ''

grinder.sleep(90)
request1902.GET('/eMunicipality/com/tcs/cdr/govt/emun/tl/w
eb/Tl002TradeLicenceController/notice-form.css')

        request1903.GET('/eMunicipality/layout/default-
calendar.css')

return result

def page20(self):
    """GET calendar.do (request 2001)."""
    result = request2001.GET('/eMunicipality/calendar.do')

    return result

def page21(self):
    """GET formValidation.do (requests 2101-2106)."""
    result =
    request2101.GET('/eMunicipality/formValidation.do')

    grinder.sleep(431)
    request2102.GET('/eMunicipality/layout/calendar/calendar.g
if')

    grinder.sleep(320)
    request2103.GET('/images/banner-emun.jpj.gif')

    request2104.GET('/eMunicipality/casimage/bg_03.gif')

    grinder.sleep(30)

```

```

        request2105.GET('/eMunicipality/images/expanded-
Magenta.gif')

grinder.sleep(8723)

request2106.GET('/eMunicipality/layout/calendar/menuarrow.
gif')

return result

def page22(self):
    """POST NoticeFormSave.do (request 2201)."""
    result =
request2201.POST('/eMunicipality/TradeLicence/NoticeFormS
ave.do',
    ( NVPair('miscSarkar', 'pradeep_sahoo'),
      NVPair('licenceNumber', self.token_licenceNumber),
      NVPair('traderFirstName', 'raha'),
      NVPair('traderMiddleName', ''),
      NVPair('traderLastName', ''),
      NVPair('trdAgencyName', 'raja'),
      NVPair('totalLicenceFees', '251'),
      NVPair('typeOfTrade', '1612'),
      NVPair('paymentDueDate', '09/05/2010'),
      NVPair('noticeType', 'FN'),
      NVPair('wardNumber', '05'),
      NVPair('remark', 'sas'),
      NVPair('licenceNumber', self.token_licenceNumber),
      NVPair('noticeNumber', self.token_noticeNumber),
      NVPair('dateOfIssue', self.token_dateOfIssue),
      NVPair('id', self.token_id),
      NVPair('applicationStatus',
self.token_applicationStatus),
      NVPair('licenceStatus', self.token_licenceStatus),
      NVPair('noticeStatus', self.token_noticeStatus),
      NVPair('commComments', self.token_commComments),
      NVPair('commappCommDate', self.token_commappCommDate),
      NVPair('miscCloseComments',
self.token_miscCloseComments),

```



```

        NVPair('miscCommDate', self.token_miscCommDate),
        NVPair('miscComments', self.token_miscComments),
                                NVPair('miscCommentsDate',
self.token_miscCommentsDate),
        NVPair('status', self.token_status),
        NVPair('statusCode', self.token_statusCode),
        NVPair('streetName', self.token_streetName),
        NVPair('section', self.token_section),
        NVPair('ulbCode', self.token_ulbCode), ),
        ( NVPair('Content-Type', 'application/x-www-form-
urlencoded'), ))
self.token_id = \
    httpUtilities.valueFromHiddenInput('id') # '10735'
self.token_noticeStatus = \
    httpUtilities.valueFromHiddenInput('noticeStatus') #
    '01'
file5 = open("addnotice.txt", "w")
print >> file5, result.text
file5.close()

return result

def page23(self):
    """GET calendar.do (request 2301)."""
    result = request2301.GET('/eMunicipality/calendar.do')

    return result

def page24(self):
    """GET formValidation.do (requests 2401-2403)."""
                                result
    request2401.GET('/eMunicipality/formValidation.do') =

    grinder.sleep(411)
    request2402.GET('/images/banner-emun.jpj.gif')

    grinder.sleep(30)

```

```

request2403.GET('/eMunicipality/casimage/bg_03.gif')

return result

def page25(self):
    """GET NoticeHomeSearch.do (requests 2501-2502)."""
    result = request2501.GET('/eMunicipality/TradeLicence/NoticeHomeSearch.do')

    grinder.sleep(221)

    request2502.GET('/eMunicipality/com/tcs/cdr/govt/emun/tl/web/Tl002TradeLicenceController/search-form-for-notice.css')

    return result

def page26(self):
    """GET calendar.do (request 2601)."""
    result = request2601.GET('/eMunicipality/calendar.do')

    return result

def page27(self):
    """GET formValidation.do (requests 2701-2704)."""
    result = request2701.GET('/eMunicipality/formValidation.do')

    grinder.sleep(500)
    request2702.GET('/images/banner-emun.jpj.gif')

    request2703.GET('/eMunicipality/layout/help.gif')

    request2704.GET('/eMunicipality/casimage/bg_03.gif')

    return result

```

```

def page28(self):
    """POST SearchFormForNoticeFetchData.do (request 2801)."""
    self.token_noticeStatus = \
        ''

    self.token_ulbCode = \
        'ULB001'

    result =
    request2801.POST('/eMunicipality/TradeLicence/SearchFormF
orNoticeFetchData.do',
        ( NVPair('noticeNumber', self.token_noticeNumber),
          NVPair('noticeStatus', self.token_noticeStatus),
          NVPair('wardNumber', '[--ALL--]'),
          NVPair('dateOfIssue', self.token_dateOfIssue),
          NVPair('applicationNo', ''),
          NVPair('miscSarkar', ''),
          NVPair('ulbCode', self.token_ulbCode),
          NVPair('streetName', self.token_streetName),
          NVPair('traderFirstName', 'raha'),
          NVPair('noticeType', ''), ),
          ( NVPair('Content-Type', 'application/x-www-form-
urlencoded'), ))

    # 6 different values for token_d16544s found in response,
    using the first one.
    self.token_d16544s = \
        httpUtilities.valueFromBodyURI('d-16544-s') # '0'
    self.token_d16544o = \
        httpUtilities.valueFromBodyURI('d-16544-o') # '2'
    self.token_d16544p = \
        httpUtilities.valueFromBodyURI('d-16544-p') # '1'
    self.token_applicationNo = \
        httpUtilities.valueFromBodyURI('applicationNo') # ''
    self.token_miscSarkar = \
        httpUtilities.valueFromBodyURI('miscSarkar') # ''
    self.token_wardNumber = \
        httpUtilities.valueFromBodyURI('wardNumber') # '[--
ALL--]'
    self.token_traderFirstName = \

```

```

        httpUtilities.valueFromBodyURI('traderFirstName') #
        'govind'
self.token_noticeType = \
    httpUtilities.valueFromBodyURI('noticeType') # ''
self.token_documentlocation = \
    httpUtilities.valueFromBodyURI('document.location') # ''
file15 = open("srchnotic.txt", "w")
print >> file15, result.text
file15.close()
return result

def page29(self):
    """GET calendar.do (request 2901)."""
    result = request2901.GET('/eMunicipality/calendar.do')

    return result

def page30(self):
    """GET formValidation.do (requests 3001-3004)."""
    result =
    request3001.GET('/eMunicipality/formValidation.do')

    grinder.sleep(581)
    request3002.GET('/images/banner-emun.jpj.gif')

    request3003.GET('/eMunicipality/layout/none.gif')

    request3004.GET('/eMunicipality/casimage/bg_03.gif')

    return result

def page31(self):
    """GET LoginUsecase.do (request 3101)."""

    # Expecting 302 'Moved Temporarily'

```

```

                                result =
request3101.GET('/eMunicipality/LoginUsecase/LoginUsecase
.do')

return result

def page32(self):
    """GET logout (requests 3201-3204)."""
    result = request3201.GET('/casserver/logout')

    grinder.sleep(80)
    request3202.GET('/casserver/images/confirm.gif')

    grinder.sleep(40)

    # Expecting 302 'Moved Temporarily'
    request3203.GET('/casserver/images/cas-footer-bg1.gif')

    request3204.GET('/casserver/login' +
        '?null')
    self.token_lt = \
        httpUtilities.valueFromHiddenInput('lt') # '_c58FCE093-
        BB1D-7F63-7F29-AC490EB8E168_k...'
    file4 = open("logout", "w")
    print >> file4, result.text
    file4.close()

    return result

def __call__(self):
    """This method is called for every run performed by the
    worker thread."""

    self.page1()      # GET home (requests 101-102)
    self.page2()      # GET css.jsp (request 201)
    self.page3()      # GET css.jsp (requests 301-304)
    self.page4()      # GET barebone.jsp (requests 401-413)

```

```

self.page5()      # GET company_logo (request 501)
self.page6()      # GET image_gallery (request 601)
self.page7()      # GET image_gallery (requests 701-702)
self.page8()      # GET image_gallery (requests 801-802)
self.page9()      # GET image_gallery (request 901)
self.page10()     # GET image_gallery (requests 1001-1005)
self.page11()     # GET image_gallery (request 1101)
self.page12()     # GET image_gallery (request 1201)
self.page13()     # GET image_gallery (request 1301)
self.page14()     # GET image_gallery (requests 1401-1406)

grinder.sleep(38)

self.page15()     # GET eMunicipality (requests 1501-1503)
self.page16()     # GET login (requests 1601-1612)

grinder.sleep(35)

self.page17()     # POST login (request 1701)
    self.page18()     # GET EmployeeLogin.do (requests 1801-
        1822)

grinder.sleep(18)

    self.page19()     # GET NoticeHomeAddNewNotice.do
        (requests 1901-1903)
self.page20()     # GET calendar.do (request 2001)

grinder.sleep(26)

self.page21()     # GET formValidation.do (requests 2101-
        2106)

grinder.sleep(21)

self.page22()     # POST NoticeFormSave.do (request 2201)

grinder.sleep(26)

self.page23()     # GET calendar.do (request 2301)

grinder.sleep(45)

```

```

self.page24()      # GET formValidation.do (requests 2401-
2403)

grinder.sleep(16)

    self.page25()      # GET NoticeHomeSearch.do (requests
2501-2502)
self.page26()      # GET calendar.do (request 2601)

grinder.sleep(47)

self.page27()      # GET formValidation.do (requests 2701-
2704)

grinder.sleep(23)

    self.page28()      # POST SearchFormForNoticeFetchData.do
(request 2801)

grinder.sleep(27)

self.page29()      # GET calendar.do (request 2901)

grinder.sleep(21)

self.page30()      # GET formValidation.do (requests 3001-
3004)

grinder.sleep(10)

self.page31()      # GET LoginUsecase.do (request 3101)

grinder.sleep(31)

self.page32()      # GET logout (requests 3201-3204)


def instrumentMethod(test, method_name, c=TestRunner):
    """Instrument a method with the given Test."""
    unadorned = getattr(c, method_name)
    import new
    method = new.instancemethod(test.wrap(unadorned), None, c)
    setattr(c, method_name, method)

```



```

# Replace each method with an instrumented version.
# You can call the unadorned method using
  self.page1.__target__().

instrumentMethod(Test(100, 'Page 1'), 'page1')
instrumentMethod(Test(200, 'Page 2'), 'page2')
instrumentMethod(Test(300, 'Page 3'), 'page3')
instrumentMethod(Test(400, 'Page 4'), 'page4')
instrumentMethod(Test(500, 'Page 5'), 'page5')
instrumentMethod(Test(600, 'Page 6'), 'page6')
instrumentMethod(Test(700, 'Page 7'), 'page7')
instrumentMethod(Test(800, 'Page 8'), 'page8')
instrumentMethod(Test(900, 'Page 9'), 'page9')
instrumentMethod(Test(1000, 'Page 10'), 'page10')
instrumentMethod(Test(1100, 'Page 11'), 'page11')
instrumentMethod(Test(1200, 'Page 12'), 'page12')
instrumentMethod(Test(1300, 'Page 13'), 'page13')
instrumentMethod(Test(1400, 'Page 14'), 'page14')
instrumentMethod(Test(1500, 'Page 15'), 'page15')
instrumentMethod(Test(1600, 'Page 16'), 'page16')
instrumentMethod(Test(1700, 'Page 17'), 'page17')
instrumentMethod(Test(1800, 'Page 18'), 'page18')
instrumentMethod(Test(1900, 'Page 19'), 'page19')
instrumentMethod(Test(2000, 'Page 20'), 'page20')
instrumentMethod(Test(2100, 'Page 21'), 'page21')
instrumentMethod(Test(2200, 'Page 22'), 'page22')
instrumentMethod(Test(2300, 'Page 23'), 'page23')
instrumentMethod(Test(2400, 'Page 24'), 'page24')
instrumentMethod(Test(2500, 'Page 25'), 'page25')
instrumentMethod(Test(2600, 'Page 26'), 'page26')
instrumentMethod(Test(2700, 'Page 27'), 'page27')
instrumentMethod(Test(2800, 'Page 28'), 'page28')
instrumentMethod(Test(2900, 'Page 29'), 'page29')
instrumentMethod(Test(3000, 'Page 30'), 'page30')
instrumentMethod(Test(3100, 'Page 31'), 'page31')
instrumentMethod(Test(3200, 'Page 32'), 'page32')

```

