webtest.wt

File: `/home/volker/webtest/webtest.wt`
Date: 19. Oct 2011, 13:45

Contents

1 Introduction

Webtest is a tool to generate http and https request and check the recieved response.
Webtest simplifies the generation of requests and offers specialised methods to analyse/check the response:

- Checking response header fields
- Checking recieved (`Set-Cookie:`) cookies
- Checking textual and binary bodies
- Checking tag structures in html/xml

Webtest helps testing responses with some additional feature:

- Execute pre- and post-tasks (setup and teardown of systems)
- Validate html and links
- measure and check response times
- generate tests from templates by by substituting placeholders.

Webtest may alos perform stress/load tests and provide benchmarks for response times.

2 Invoking Webtest

Here is the overview of how to invoke webtes:

```
webtest [-test] [common options] [test options] <suite>...
webtest -check [common options] <suite>...
webtest -bench [common options] [bench options] <suite>...
webtest -stress [common options] [stress options] <bg-suite> <suite>
webtest -tag <tagSpec> <htmlFile>
```

The structure of the test suite files are described in the document: reference-suite.wt

3 Common Options

All modes of webtest recognise the following options:

- `-log` _n: General log level. `0`: off, `1`: errors, `2`: warnings, `3`: info, `4`: debug, `5`: trace
- `-log.tag` *n*: Log level for tag test (tag package)
- `-log.suite` *n*: Log level for suite test (suite package)
- `-seed` *n*: Use *n* as random seed insted of current time.
- `-od` *path*: Set output path to *path*. Default to current directory.
- `-tests` *list*: Run only those test given in *list*.

Selecting just some tests out of one ore more suites is done with the `-tests` option. Its argument is a comma separated list of all tests tu run:

- Full test name
- Test name with wildcards `*` and `?`
- Full number of test in the form <suit-no>.<test-no>
- Abrevated number of test: <test-no>

E.g. the following

```
webtest -tests 'Homepage*,9,1.11,2.3' suiteA.wt suiteB.wt
```

would run all test whose name start wih Homepage as well as the ninth and eleventh test in suiteA.wt and the third of suiteB.wt.

4 Checking Test-Suites

If webtest is invoked with the `-check` flag, than it will read all the given test suites, parse them and output them again. This helps checking for syntactical errors in the suites and gives an overview which test are present.

5 Performing Test

To perform the tests in the given test-suite files invoke webtest as follows:

```
  webtest [-test] [common options] [test options] <suite>...
```

The `-test` option is optional, it's the default mode for webtest.
The following options are recognized in test mode:

- `-dump` *mode*: Force dumping on/off for debugging purpose. *mode* may be `none` to turn dumping off, `body` to dump the recieved response bodies or `all` to dump the whole wiretalk. If `-dump` is unset, then the individual settings in each test are honoured.
- `-validate` *n*: Allow link checking and/or validating if requested by a test. *n* may be `1` to allow validating links `2` to allow validating the (x)html or `3` to allow both.
- `-junit` *file*: Write a junit compatible report as xml to *file*

Please note, that `-dump` **overrides** individual settings made in the `SETTING` section of each test whereas `-validate` **activates** the individually made settings.

6 Benchmarking Response Times

Webtest can perform requests repeatedly and generate statistics about the response times of the request. It's invoked as follows:

```
  webtest -bench [common options] [bench options] <suite>...
```

Webtest will perform each request of all the tests in the given suites 15 times and report a little statistic about the response times.
Benchmarking knows just one option:

- `-runs` *n*: Change the number of repetitions to *n*. Must be >= 5.

7 Stress and Load Testing

Stress- or load-testing in webtest happens the following way: A background-suite is used to create a certain load. No checks or tests are performed on this background-suite, this suite is solely used to generate requests. Additionally to this load a real test-suite is executed. The outcome (test, failures, response times) of this test-suite is monitored.
The background load is increased after performing the whole test-suite (maybe with repetitions). Several parameters controll when this process finishes.
Invoke webtest like this to use stresstesting:

```
  webtest -stress [common opts] [stress options] <bg-suite> <suite>
```

The only parameter to the background load is the number of parallel request made. Webtest will loop over the background suite and start a new request if the number of currently active request drop below the given load.
The following options control how the load is increased.

- `-ramp.start` *n*: Start with *n* parallel background requests. Default: 5
- `-ramp.step` *d*: Increase number of parallel background requests by *d* on each iteration. Default: 5
- `-ramp.sleep` *ms*: Sleep time petween iterations in miliseconds. Default: 1000

- `-ramp.rep` *k*: Repeate the testsuite *k* times during one iteration. Default: 1.

The following options control when to stop stresstesting

- `-stop.ff` *g*: Stop if a fraction *g* of all checked conditions fail (Fail Fraction). Defaults: 0.2
- `-stop.art` *ms*: Stop if the average response time exceeds *ms* miliseconds (Average Response Time). Defaults: 120000 (= 2 minutes).
- `-stop.mrt` *ms*: Stop if the longest response time exceeds *ms* miliseconds (Maximum Response Time). Defaults: 240000 (= 4 minutes).
- `-stop.rtj` *n*: Stop if average response time jumps by more than a factor of *n* compared to the last iteration (Run Time Jump). Default: 5 times longer execution
- `-stop.rti` *m*: Stop if average response time exeeds *m* * inital response time (Run Time Increase). Default: 50.
- `-stop.mpr` *p*: Stop number of parallel background requests equals (or exceeds) *p* (Maximal Parallel Requests). Default: 250

8 Debugging Tag Specs

Sometimes it is complicated to see why a certain html page does not contain (or contains) a certain tag described by a complicated tag spec.
Webtest offers one method to help debugging this:

```
  webtest -tag <tagSpec> <htmlFile>
```

Invoke webtest with the `-tag` mode and provide your tag spec as first argument and the path to the html file as the second argument on the command line.
Hint: You can dump the html by executing just one test without repetitions and dumping the body.
E.g.

```
  webtest -tag "a !target\n  span == XYZ" index.html
```