TextTest Documentation

Release 3.27

Geoff Bache, Emily Bache

February 10, 2015

Contents

Contents:

CHAPTER 1

What is TextTest?

TextTest is a tool for automated testing of programs written in almost any programming language. It can be used for unit testing, but is more often used for functional testing of larger pieces of code.

1.1 When to use TextTest

TextTest is often used to test:

- **Batch systems** (long running programs with a command line interface)
- **Legacy systems** (Approval testing works especially well when the system is already built)
- **Rich Client applications** (in combination with a GUI-recording tool, StoryText)
- **Components in a Service-Based architecture**
- **Command line programs and scripts** (in combination with CaptureMock)

At present TextTest is rarely used to test web applications, there is limited support for this.

1.2 How it works

TextTest can test a program written in any programming language, so long as it can be executed from the command line.

Tests are defined in a directory structure:

```
+- Test Suite Name
  +- config.<app>
  +- testsuite.<app>
  | +- A Test Case Name
  |   +- options.<app>
  |   +- stdout.<app>
  |   +- stderr.<app>
  | +- A Second Test Case Name
  |   +- options.<app>
  |   +- stdout.<app>
  |   +- stderr.<app>
```

In this root test suite folder, there is a config file, a test suite file, and some test case folders. The name of the test case folder is used as the name of the test case. Test case folders contain files which are given as input to the system under test, and "golden master" files containing the approved (expected) output.

TextTest uses an "Approval Testing" approach, which means that when you run a test, it will compare the actual output of your program agains the saved "Golden Master" files in the test case folder. If there is a difference in the output, the test fails. TextTest gives you the opportunity to update the "Golden Master" when this happens.

All files that relate to a particular tested application share the same file suffix. You configure the value of `<app>` when you first create a test suite.

CHAPTER 2

Installation Guide

TextTest is available via the pip package manager:

    pip install texttest

This will make the texttest command line tools available on your system. TextTest also has a graphical user interface for managing your test suite. In order to use it, you will have to install pygtk, which is not currently available via the pip package manager. Installation is platform specific, see below.

2.1 Linux

- Check you if you already have PyGTK - many Linux systems come with it already installed:

```
$> python
Python 2.7.1
[GCC 4.6.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import gtk
>>>
```

If this produces an error then you don't have PyGTK and you will need to install it. Refer to the PyGTK home page. You need at least GTK 2.18 and PyGTK 2.16

- Download the zip file containing all the TextTest source code and tests from http://sf.net/projects/texttest. Unpack it into an installation directory, such as `~/tools/texttest`.
- You may also want to install a separate diff tool such as TkDiff if you havn't already got it.

2.2 Mac

- Install Homebrew. Check that `/usr/local/bin` is on your `$PATH`
- Install XQuartz. Log in and out again (to activate XQuartz as a windows manager)
- Set the following environment variables:

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
```

- Install Python (version 2.6 or 2.7):

```
brew install python
```

- Install PyGTK:

```
brew install pygtk
```

- You may also want to install a separate diff tool such as TkDiff if you havn't already got it.

2.3 Windows

- Download the zip file containing all the TextTest source code and tests from http://sf.net/projects/texttest. In there is a windows installer which you should run. Re-start windows when you've done that. Note: this installer will set the environment variable `$TEXTTEST_HOME` to `C:\\tests`

If the all-in-one installer doesn't work for you, you can install the component pieces by themselves.

- Install GTK using this GTK 2.18 bundle. Unzip it somewhere like *C:GTK* (avoid a path containing spaces). Add the *bin* subdirectory to your $PATH.
- Install PyGTK. Run the three installers at the top of the PyGTK downloads page. (avoid installation paths containing spaces)
- Download the zip file containing all the TextTest source code and tests from Sourceforge (http://sf.net/projects/texttest). Unpack it into an installation directory, such as *~/tools/texttest*.

2.4 Troubleshooting

- Make sure you have at least Python 2.6, GTK 2.18 and PyGTK 2.16 installed on your system.
- Make sure that pygtk is installed in the site-packages of the Python installation you're using.
- There are some mac-specific hints in the section below.
- If you still can't get it to work, contact us via the TextTest mailing list.

2.5 Mac-specific Troubleshooting

- If you get errors like this:

```
texttest.py: Fatal IO error 35 (Resource temporarily unavailable) on X server :0.0
```

or:

```
RuntimeError: could not create GdkCursor object
```

You could try starting XQuartz first before starting the TextTest GUI. (It's under `/Applications/Utilities/XQuartz`) Once it's started, try running `texttest` again. If that doesn't work, when you've started XQuartz, on the "Appliations" menu, start "Terminal". Try starting texttest from this terminal window instead.

- If you get errors like this:

```
GtkWarning: Could not find the icon 'inode-directory'. The 'hicolor' theme was not found either,
```

You could try:

```
brew install hicolor-icon-theme
```

CHAPTER 3

Getting Started

TextTest can be used either from the command line, or via a Graphical User Interface (GUI). The command line interface is mostly used for running the tests unattended (for example on a Continuous Integration server), or for systems where no window manager is available. For day-to-day work creating and maintaing tests, the GUI makes a lot more functionality available.

First we'll show you a minimal example using just the command line, and then a slightly larger example using the GUI.

3.1 A Minimal Example on the Command Line

For example, if you have an executable script `myapp.sh`:

```
# myapp.sh
echo "Hello World"
```

Create a config file `config.myapp` telling TextTest how to execute your program:

```
# config.myapp
executable:myapp.sh
```

Create a subfolder to be a test case, for example named "HelloWorld", and create a file `testsuite.myapp` telling TextTest to look in it:

```
# testsuite.myapp
HelloWorld
```

Execute TextTest in the same folder as the config file:

```
texttest -con -d .
```

It will execute your test case, and it will fail. Hopefully it will fail for the right reason - that you have not defined a "Golden Master" file for your test case "HelloWorld". TextTest should ask you if you want to save the output. If you answer Yes, then it will save two new files in your HelloWorld folder - *stdout.myapp* and *stderr.myapp*. These files contain the output you just got from your script when you executed it, on standard output and standard error respectively.

If your test is failing for some other reason and you can't work out what's going on, please contact us via the TextTest mailing list. We are in the process of compiling a Frequently Asked Questions page.

3.2 Using the TextTest GUI

Start TextTest with the argument `--new` to tell it you want to build a test suite for an application:

```
texttest --new
```

This will present you with a dialog where you can fill in details of the application you want to test.

CHAPTER 4

Frequently Asked Questions

We are in the process of building this page. Please put your questions to the TextTest mailing list. Popular questions will appear here.

CHAPTER 5

Indices and tables

- *genindex*
- *search*