STF

System Test Framework

Table of Contents

[2 Introduction 3](#_Toc434060425)

[2.1 Who is making STF – and why 3](#_Toc434060426)

[2.2 Logging 3](#_Toc434060427)

[2.3 Configuration 3](#_Toc434060428)

[2.4 Plugins 3](#_Toc434060429)

[2.5 Utils 3](#_Toc434060430)

[3 Components 4](#_Toc434060431)

[3.1 StfArchiver 4](#_Toc434060432)

[3.2 StfAssert 4](#_Toc434060433)

[3.3 StfConfiguration 4](#_Toc434060434)

[3.4 StfKernel 4](#_Toc434060435)

[3.5 StfLogger 5](#_Toc434060436)

[3.6 StfPluginLoader 5](#_Toc434060437)

[3.7 StfTestScriptBase 5](#_Toc434060438)

[3.8 StfUtils 5](#_Toc434060439)

[4 Samples 6](#_Toc434060440)

[5 Getting Started 7](#_Toc434060441)

[5.1 Start with a MsTest standard unit test 7](#_Toc434060442)

[5.2 Get the NuGet package (Mir.Stf) 7](#_Toc434060443)

[5.3 Get Stf into action 8](#_Toc434060444)

[5.4 The StfLog 9](#_Toc434060445)

[5.5 The Stf Asserter 11](#_Toc434060446)

[6 Sources and drops 12](#_Toc434060447)

[6.1 License 12](#_Toc434060448)

[6.2 Source code is on github 12](#_Toc434060449)

[6.3 NuGet package 12](#_Toc434060450)

# Introduction

## Who is making STF – and why

We are a group of people that have been working with test automation for lots of years. We have made many frameworks to get the stuff working.

We have identified a series of components, which we have written many times repeatedly. We thought it was time for an Open Source version of the ideas. Then when we are starting on an assignment, we can focus on the customer specifics and not on the usual stuff your need.

When automating we have identified that you need something to handle the:

* Logging
* Configuration
* Plugins
* Utils

In short, what we want to achieve is a framework for frameworks

## Logging

Test logs are different from other loggings. We want a log that is useable for testers, developers and test framework developers.

## Configuration

We want our framework (and thereby the test scripts) to handle configuration in a uniform way. To get to a environment agnostic approach, and to be maintainable from a test execution perspective.

## Plugins

We do not want to do a monolith. What we want is a core that is seamlessly extendable with plugins for various testing areas and tasks.

## Utils

# Components

STF consists of the following components:

* StfArchiver
* StfAssert
* StfConfiguration
* StfKernel
* StfLogger
* StfPluginLoader
* StfTestScriptBase
* StfUtils

## StfArchiver

The role of the StfArchiver is to get the testware of the executing test agent (machine). You do not want your test results and data to lay around in your test rig.

You want it shipped to a file share and perhaps uploaded to your favorite Test Case Management tool (TFS, HP ALM, or Jira).

The idea is to ease access to test results.

## StfAssert

The StfAsserter asserts like MsTest Assert (actually some of the asserts are using MsTest Assert) – but it extends the assertion to the logger. Where you want your asserts to be logged.

## StfConfiguration

The base of the frame work is XML and having <section> within <section> and the leafs are <key name=”” value=””>.

The StfConfiguration framework enable overlaying configuration at endless levels like for suite, testcase, machine, user, plugin.

The StfKernel will handle this for plugins.

## StfKernel

The StfKernel is responsible for getting it all together. Most of the components are designed to be usable on its own.

When you inherit from StfTestScript base, STF pulls it all together. That is the responsibility of the StfKernel.

## StfLogger

A Html test result logger.

It logs all entries as DIV tags, and then having some JavaScript buttons to control which DIVs to display.

## StfPluginLoader

Your tests will need something specific for your SUT (System Under Test). You make a plugin that handles that and put it into a StfPlugin.

The StfKernel will load your plugin, and register the types and instances you will be using in your test scripts.

So the plugin story is pretty much drop-and-use. If a plugin is broken, it will only interfere with those actually using the plugin.

## StfTestScriptBase

It extends MsTest test execution engine.

For DataDriven tests it will make a logfile for each iteration, and in the end a summery logfile.

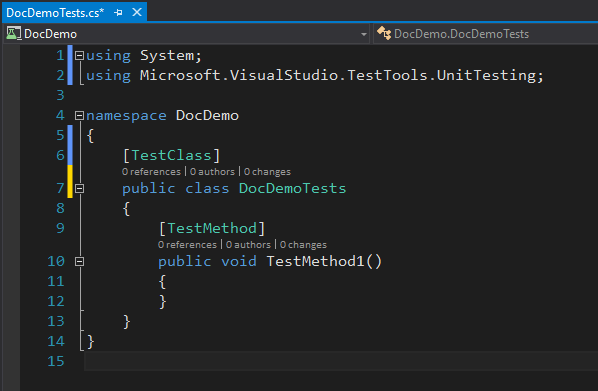
## StfUtils

Some utils that are usuable for scripting, and implementing test models.

# Samples

# Getting Started

## Start with a MsTest standard unit test



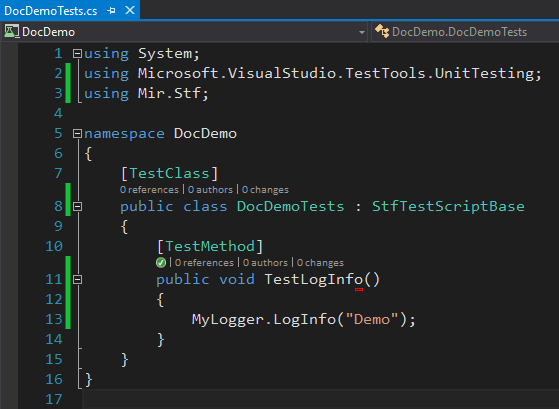
## Get the NuGet package (Mir.Stf)

Using the (Visual Studio) standard NuGet Package manager

## Get Stf into action

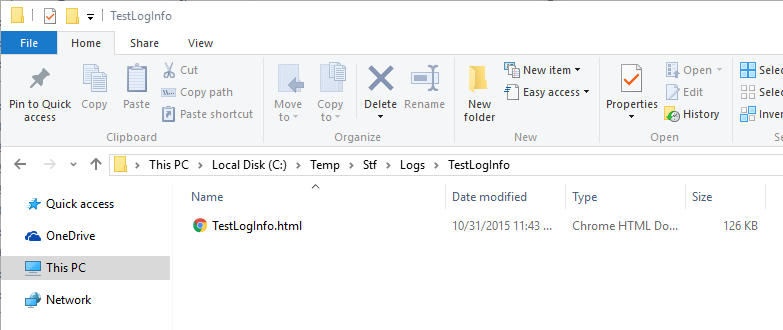
You do that by letting your test class inherit from StfTestScriptBase

You now have a logger called MyLogger.

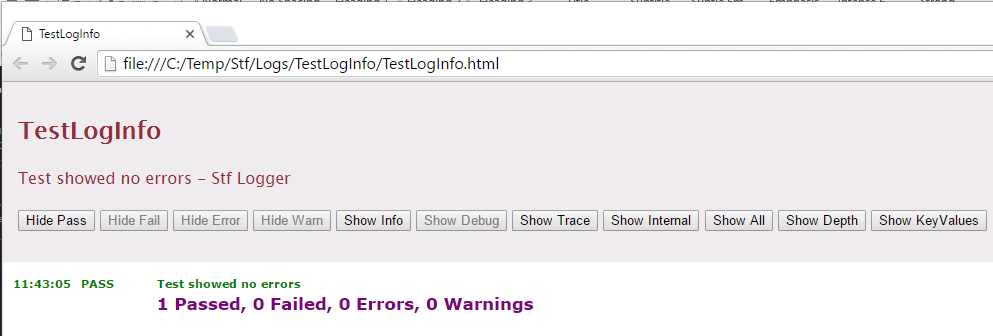


## The StfLog

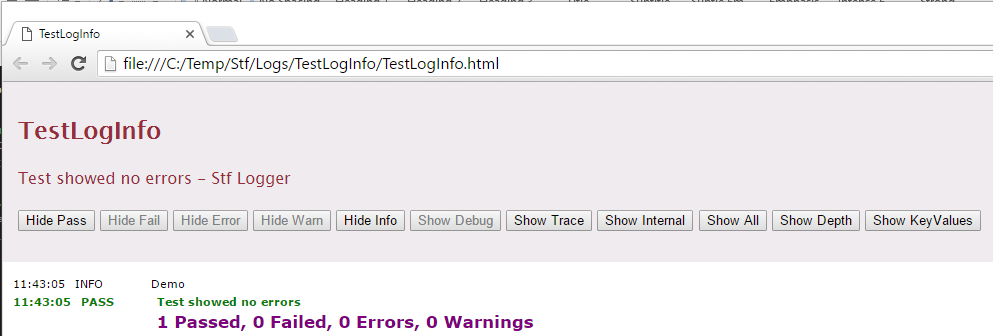
Running the test will create a folder structure and a Html LogResult. The location is configurable – out of the box the location is c:\temp\Stf\Logs



Opening is will give you something like this:



You use the buttons to handle the level of information you want to see. Like “Show Info”:

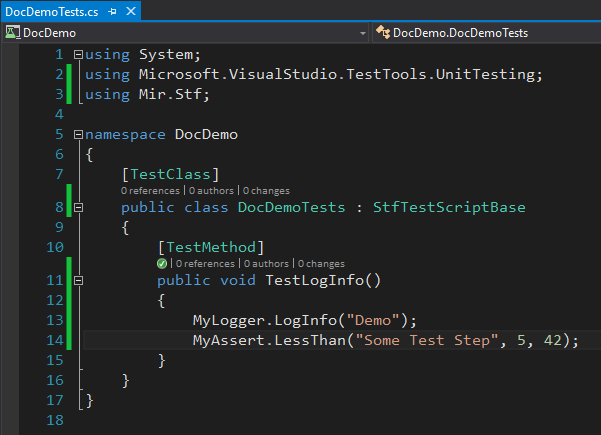


Now we see the “Demo” entry from the test script.

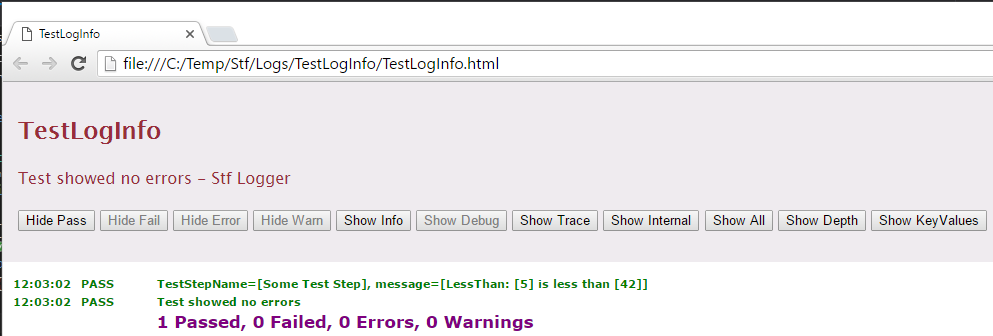
## The Stf Asserter

You get to the asserter by using MyAssert. MyAsserts uses the STF logger, so every time you assert something it goes into the log (loglevel is configurable).

Use it like this:



The logger now contains:



# Sources and drops

## License

Copyright governed by Artistic license as described here:

<http://www.perlfoundation.org/artistic_license_2_0>

## Source code is on github

<https://github.com/UlrichFreiberg/STF>

## NuGet package

Name is Mir.Stf

