# CiviCRM Design Note – Regression Tests

Tony Maynard-Smith 5/3/2021

This note gives an outline of the automated regression tests which have been implemented using Rspec and Watir. The motivation for setting this up was the occurrence of several regression errors, both in upgrading CiviCRM software, and in making changes to the LALG configuration.

# References

The following were found useful in installing the necessary software, and understanding some of the basic elements of the libraries used:

<http://watir.com/guides/>

<https://www.tutorialspoint.com/rspec/rspec_introduction.htm>

<http://testerstories.com/2011/10/automated-testing-with-watir-and-rspec-part-1/>

<http://testerstories.com/2011/10/automated-testing-with-watir-and-rspec-part-2/>

<https://www.tutorialspoint.com/watir/index.htm>

<https://www.rubyguides.com/2018/07/rspec-tutorial/>

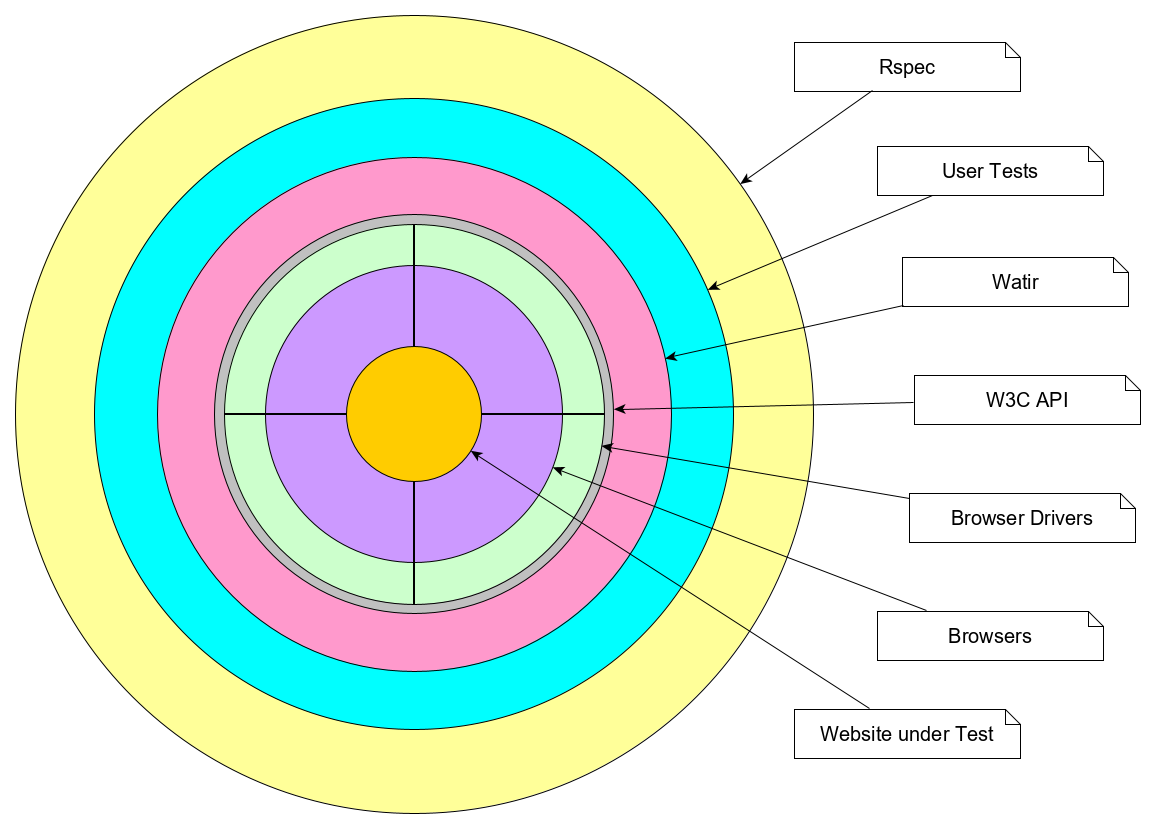
<https://ruby-doc.com/docs/ProgrammingRuby/>

<https://relishapp.com/rspec/rspec-core/v/3-9/docs>

# Architecture

The test suite resides on a client PC and executes tests on the website using an API built into common browsers to enable such automation. The tests use the Ruby language, and two Ruby libraries:

* Watir: Abstracts the browser API and adds many useful features, such as locating elements and waiting for elements to become available.
* Rspec: Structures a set of tests, provides facilities for checking various conditions, and running selected tests.



This very schematic diagram shows:

* Overall control of running the test processes is exercised by Rspec.
* The test processes are defined by the user written test specifications. These are written in Ruby using elements from the Rspec and Watir libraries (also written in Ruby).
* All communication between the User Test Specifications and the target website are through a stack consisting of:
  + The Watir library functions.
  + A standard API defined by the W3C
  + A Browser Driver which maps the common W3C interface onto the API provided by a particular browser.
  + A standard Browser. Chrome, Firefox, Edge and Safari are documented as supported, but so far I have only found the driver for Chrome.
* The LALG website under test.

# Dependencies

## Client-side

You need to install Ruby, libraries (Gems) for Watir and Rspec, and drivers for the relevant browsers on your PC. See the first reference document above.

[AMS. On my PC they are in c:\Ruby26-x64.]

## Server-side

The following provisions must be made on the website to be tested to enable the tests to run.

* The website must be set up with a Drupal User having Membership Administrator privilege and:  
   Username: watir  
   Password WatirTesting987  
   First Name Joe  
   Last Name WatirAdmin  
   Postcode JW1 1JW
* To allow tests to run which simulate actions by an end User, rather than an Administrator:
  + The addition to the lalg\_dutils Module which creates the permission ‘LALG use Stripe test mode’ must be included. Allows card payments to be tested.
  + A Role ‘watir tests’ must be provided giving access to the above permission.
  + The User ‘watir’ mentioned above must also have the permissions ‘Administer Permissions’ and ‘Administer Users’. Allows creation of a test end-User, and allocation of the above Role.
  + The (updated) patch PR373 for Webform\_Civicrm must be installed. This actually implements (subject to the permission above) a Payment Test mode using the argument ‘?payment=test’, so the live operation is not (should not be) interfered with. See <https://github.com/colemanw/webform_civicrm/pull/373>.

Also, to allow the end-User to be created you need to turn off CAPTCHA on the user\_register form (Drupal Configuration >> People >> CAPTCHA menu). This must be done manually for the duration of the test.

# Test Implementation

## Test Versions

The Regression Tests are now held on Github in the lalgwebdev/RegressionTests repository. The following versions are currently defined/planned.

v1.11  
This first version is a copy of the Drupal 7 Workflow tests, copied to Github repository. No further changes on this branch.

v2.x  
Partitioned and Refactored Drupal 7 Tests, including compatible updates as and when.

v3.x  
Drupal 8 upgrades, incompatible with D7. This will be in a separate Branch. Allowing independent update of the 2.x and 3.x series if required.

## Test Structure

Each Test file contains several related tests, that can be run individually or as a batch. The tests are divided into:

* Unit Tests.   
  To test relatively small sections of functionality. Can be run individually or as a batch. Test Files include:
  + Framework. Widely used functions such as Login, Clean Data.
  + Function Specific. E.g. CiviRules, or Tokens.
* Workflow Tests  
  To test Use Cases which are commonly used by Membership Admins or End Users. Expected to be run as a batch. Test Files include:
  + Basic - Common functions, Admin and User. General confidence test.
  + Extended - Functions not covered in Basic, including where we identified particular problems with standard software.
  + Extra - Individual functions, included in the above as part of a longer run. May be useful to run individually if focussing on a specific point.
  + Old - Tests for discontinued features, notably Pay Later.
* Library Files  
  There are also library files containing functions called up by the various Tests.
  + Common Functions – mainly login, create users, etc.
  + Workflow Functions – the functional content of the Workflow tests.

# Test Execution

Tests are run from a Windows Command Line (cmd) window, and it is assumed that the standard software has been installed with the Path environment variable set up so that ruby and rspec may be invoked directly from a command prompt.

To run one or more tests:

* cd to directory containing Test scripts
* Select Domain to test:  
   set RspecDomain=xxx <Where xxx = www, dev, tmp, etc.>
* To execute all tests in one file:  
   rspec <Test filename>.rb -f html -o 'results.html'
* To execute individual Tests:  
   rspec <filename>.rb -e 'Test-nn' -f html -o 'results.html'

**Notes:**

* Results filename may be 'anything.html', but by convention always starts with 'result' or 'Result'. These are ignored by the sourcecode version control system.
* Most tests clear the Print Cards flag on the test Contacts, but should leave any flags on live contacts alone. But be aware.
* Test data from the last test to be executed is left in the system and not cleaned out. This is deliberate to assist diagnosis of possible failures.

# Results

Following completion of a test run the results may be displayed by opening the HTML file quoted in the initial command line. A successful result looks like the following:



An error report looks something like this – though this an artificial example.

