TikiTests

TikiTests is the built-in testing tool within TikiWiki. For documentation, please see doc:TikiTests.

It could eventually be used as a tool to report bugs (better than screenshots) and even maybe help configure TikiWiki (like a wizard) in conjunction with the Profiles.

Questions

Q1: Does TikiTest automatically restore the DB to its original state after each test?

SC: No, it only replays the url with the some GET and POST data.

ML: in my tests, no, which makes suitable for a wizard

AD: The framework should be usable in one of two modes. In Testing mode, we NEVER want the side effects of a script to remain in the DB after the script has exectued. In Wizard mode, we ALWAYS want the side effects of the script to remain in the DB after the script was executed.

Resetting the application under test after each test is a well established pattern in automated testing, and in my experience, it is critical. I have done it both ways (restoring and not restoring), and the not-restoring approach causes major headaches, where the output of one test was dependant on whether or not certain other tests had been executed before it. This doesn't matter if you always executed ALL the tests in the exact same order. But usually, you can't afford to execute the whole test suite, because it may take say, 30 mins or more. And you want programmers to run tests all the time, ideally after every couple of lines they change! So you want to be able to only execute a small portion of the test suite which you think is most relevant to the code you are currently writing.

Do you have plans to make the testing infrastructure restore the DB after each test? If so, do you have ideas about how this restore can be made super-fast?

Q2: What user will the tests run under? Is that something that can be controlled on an individual test basis?

SC: Yes, you have two options: you can choose to use the current session both for the recording and the replay. So the recording is done with the user recording the test. And the replay use the user who launch the tikitest replay.

AD: It seems to me that the replay should ALWAYS run under the user under which the test was recorded, so that the test is replayed in the exact same conditions under which was recorded.

ML: Sometimes, I would like to run the exact same operations with 2 different users (with different perms) and I want to make sure one of them fails. (ex.: can't access a page). Would the option of running another user be good or should I write a new test? How to do it not to have too much overhead?

If TikiTests worked that way, I think it would address all my needs in terms of what users the script runs

under. I could create a test that runs as user X, simply by logging as X for the recording. I could record different tests using different users, to simulate scenarios of use by different kinds of users (ex: an Admin configuring the multilingual features, an anonymous user switching to a different language version of a page, etc..).

Speaking of anonymous, can I record tests as anonymous?

SC: Yes, you can record tests as anonymou. You will have an error at the end of the recoding, because you are redirected to tiki-test_edit.php which requires tiki_p_edit_tikitests. But the test is recorded and you can finish editing it by login in with the right user.

Q3: Is it possible to run different tests using different DBs. For example, for testing multilingual features, I will probably need a DB that is configured with multilingual features on. Other people testing other things may need a DB that is configured differently.

SC: Not for the noment, I only record the url+GET and POST data+refererer. But we could had more, but we will need a way to change the database in Tiki...

AD: I think that will be necessary, because Tiki is such a large and versatile system that testing all of its features might require the creation of a pretty large and complex test DB. This in turn makes individual tests harder to understand and less predictable. For example, a change made to the DB for the purpose of testing features in area X, may affect tests that were written to test features in a completely unrelated area Y. It would be better if we could create different smaller and simpler test DBs targeted for specific feature areas. For example, we could have a small DB that we use to test multilingual features.

Again, I have done it both ways: with a single large complex DB, and with multiple small and simple DBs targeted to particular kinds of tests. I have found the multiple small DBs approach to work much better in avoiding dependancies between testing areas, and making the tests easier to comprehend.

Another advantage of having multiple small and simple DBs, is that it would be faster to restore them after each test (see Q1). If we need to restore a large DB after each test, this will make running of the test suite much smaller.

Q4: is it possible to create suites of tests? It seems right now we can only play one test at a time.

SC: Not for the moment but I am thinking about it. Here we want to make automated tests so I am planning to code a standalone PHP script that gets multiples XML file on the command line...

AD: Excellent. A php script is sufficient for my needs. No need to enable running the suite from the browser (although that would be nice, but to do on a rainy day kind of thing ;-)).

Q5: Can I check multiple XPath expressions for a given page? It seems right now, you can only check one assertion for each page. But I often find I need to check more than one assertion per dialog screen.

SC: Not for the moment but it could be added sure, good idea ! For the moment you can just edit the XML file and duplicate the URL you want to check for multiple XPath expression (ok ugly workaround, but it is now, no coding necessary ! ;p)

AD: Will that approach result in multiple requests for the same URL? That might turn out to have undesirable side effects. In other words, it could be that the second invocation of the URL does not return the same results as the first one. Also, it would make running the tests slower.

If the approach you propose does not end up invoking the same URL multiple times, it works fine for me. I have been writing tests without nice record-and-playback GUIs for ages, so editing an XML file to modify a test is fine.

Q6: I have recorded a test. Now, I want to write some XPath checks for each of the steps. How can I see the content of the HTML for each step?

SC: You can try nothing or //* as the Xpath expression and press on the test button, you will have the HTML recoded, and then you can use your browser to view the HTML code. Ok if this is a popular demand I will code something, should not take too long ! ;p

AD: The current solution works for me. A "view html" button would be nice of course, but not absolutely necessary for now (compared to say, restoring the DB after each test)

Q7: Have you thought about having a comment field associated to each step of a test case? This would allow the creator of the test to convey the rationale for the test and what it is testing for.

SC: Goog idea, yes I had thought about a description for the test but not for individual steps... Will code something...

AD: Thx. I think this will be very important, because otherwise the tests will be very cryptic. Being able to understand the intent of a test is key to fixing a bug that causes the test to fail.

Q8: Where will these tests run? Locally on the developer or tester's machine, or on some shared public machine?

AD: Running locally has the advantage that tests will run faster, so I think it needs to be supported. At the same time, we want to make it dirt easy for people to create new tests, and installing Tiki to run locally is a hard task. So it would be nice to also have a public place where people can go to create new tests. The problem here is that if two people run tests on the same Tiki instance at the same time, the outputs of the tests may end up being unpredictable. So we would need some kind of Tiki testing farm, with several test

Tiki instances installed. A tester could log onto any of those, at which point no one else could access it until he has logged off. I know it's a lot of work, but I think that's the only way we COULD make a public test site work. Otherwise, we have to stick with the local approach.

BTW: On 2008-06-03, I tried to run TikiTests on my local installation, and it didn't work. When I clicked on the test list link, I got an error message.

Wishlist

	Rating	Subject	Submitted by	Importance	Easy to solve?	Priority	Category	Volunteered to solve	LastModif	Comments
●	(0)	15.x: Allow running console.php to apply profiles from behind a proxy (to run new R script to check errors when testing the application on all Profiles on several tiki branches)	Xavier de Pedro	4	8	32	• Feature request • Support request • Community projects • Dogfood on a *.tiki.org site • Less than 30-minutes fix		2016-02-20	2 xavi-21 Feb 16
⊗	(0)	Test for PHP5 and provide graceful error message	Marc Laporte	9 high		0	• Error		2009-04-20	0
⊗	(0)	https://tiki.org/Demo (links broken for demos)	John Morris	10 high		0	• Dogfood on a *.tiki.org site		2017-08-18	2 jmorris-18 Aug 17

Developers:

- Stéphane Casset <sept@logidee.com>

Testers (testing the testing suite!) :

- Alain Désilets
- Marc Laporte

Some ideas:

- Use TikiTests as a way to report complex bugs (much better than a screenshot). Users could start with a fresh install (like OpenSourceCMS) and record all the steps they need to do to understand/see and reproduce the bug. It would then be very easy for developers to "replay" it and see if a fix changed something.
- Ways to share tests via SVN. (have a centralized quality.tw.o server where tests are run automatically)
- testing changes in PHP/Smarty variable output?
- Splitting warnings and errors as separate options.
- Setting password using admin....switch user.
- automated test after SVN commit.
- update test with latest results - i.e. accept changes as OK.
- profiles use in TikiTests to smooth flow

Developers notes

From Alain Désilets (alain.desilets@nrc-cnrc.gc.ca)

This is a great start! I think automated testing will be an important ingredient in the future, to keep TikiWiki stable.

That said, it seems to me that TikiTests still falls short of meeting many of my needs as a Test Driven Developer. I want to start ussing automated tests on Tiki ASAP, so let's figure out what's missing, and how to best support those needs.

Use cases for Tiki automated testing infrastructure

Below are a bunch of Use Cases that an automated testing infrastructure should support for TikiWiki.

- Testing Use Case: Non-dev wants to create a test and share it with the dev community
- Testing Use Case: Dev wants to create a test and share it with the dev community
- Testing Use Case: Dev wants to ensure that a particular version of Tiki passes all tests
- Testing Use Case: Dev wants to know if he broke something with his most recent change

- Testing Use Case: Dev wants to write a series of related and very similar tests
- Dev needs to test some embedded JavaScript code
- Dev needs to test that it is in fact possible for the user to move from a certain page to another
- Testing Use Case: Reader of a test must be able to easily understand its intent and that of every step in the test
- Testing Use Case: Dev needs to be able to modify an existing test

If we can get TikiTests to a state where it can support the above use cases, I (Alain Désilets) SOLEMNLY PROMISE to spend at least 2 hours per week writing tests for Tiki using TikiTests. I will also try to convince my colleague Marta Stojanovic to do the same (should be an easy sell... she sorely misses automated tests in her work on translation).

Technical issues to be investigated

Below are a bunch of issues that need to be resolved. Each issue has an impact on one or more of the Use Cases above.

- How to seed DB with test data
- How best to share tests with dev community
- How to test JavaScript
- How to test that end user can indeed go from one page to the next in a test
- How to organize tests into suites
- How to reuse parts of a script in other scripts

TikiTests versus phpunit+Selenium tradeoffs

There are two infrastructures that could possibly be used to implement automated GUI-level tests for Tiki:

- TikiTest

- phpunit+Selenium

Below is a table comparing the pros and cons of each approach.

phpunit+Selenium	Could be fixed	TikiTests	Could be fixed
PRO: Mature, mainstream framework, used for many years by a large community		CON: Experimental framework, not widely tested yet, and limited to the TikiWiki community	no
PRO: Large support community		CON: Very small support community at the moment, and could not grow beyond the circle of TikiWiki	no
PRO: Already usable by developers as is		CON: Not already usable by developers as is	yes
CON: Hard to use by non-developers	no	PRO: Usable by non-developers	
PRO: Can test embedded JavaScript		CON: Cannot test embedded JavaScript	no
PRO: Can organise tests into suites		CON: Cannot at the moment organise tests into suites	yes
PRO: Can verify that it's indeed possible to go from one screen to the next in a test scenario		CON: Cannot verify that it's indeed possible to go from one screen to the next in a test scenario	no
PRO: Bits that are common to more than one tests can be modularized into methods		CON: only start state can be modularized and reused across different tests	no
PRO: Test can be made easy to read through the use of comments and appropriately chosen method and argument names		CON: At the moment, no way to make the intent of a test clear, through comments or things of the sort	yes

Summarizing this table:

Approach	# Cons	# non-fixable
phpunit+Selenium	1	1
TikiTests	8	5

In other words, TikiTests seems to have a lot more cons than phpunit+Selenium, most of them cannot seem

to be fixable.

However, number of cons is not the only thing that matters. The importance of each cons must also be taken into account.

Although phpunit+Selenium only has one con compared to TikiTests, it is a major one, i.e., it cannot support creation of tests by non-tecchies. This might be a large enough con to trump everything else. But maybe not. We neeed to think carefully about this.

In the end, it kind of boils down to whether or not we believe that hordes of non-tecchies will start writing tests with TikiTests, and that those tests will be clean enough to be useful for development purposes.

I (Alain Désilets) do not have the answer to that question. Like many things collaborative, it's probably impossible to answer it without having actually tried it.

In the short term though, one major con of TikiTests for me is that it is not already usable for development purposes as is, and I need to start writing tests right now (I have been doing Test Driven Development for 6 years now, and simply cannot survive without the safety net of tests... especially not with a system as large and complex as TikiWiki).

So... I think what I will do is this.

- I will try to get phpunit+Selenium to work (I estimate one day)
- I will do some tests to figure out the best way to set the Tiki instance under test in a given starting state (this will be useful whether we ultimately go for TikiTest or phpunit+Selenium).
- I will start implementing some tests in phpunit+Selenium
- I will continue interact with the TikiTests folks, to help them grow TikiTests towards something that I think is useful to me (I think it can be done).
- If and when TikiTests does get there, I will gladly convert all my phpunit+Selenium scripts to TikiTest format. It should not be too hard, since I believe Selenium actually plays tests through an actual browser. So, all that would be needed would be to change the setUp() method of the various TestCase classes, so that it first puts the wiki in recording mode. The tearDown() class would then carry out actions required to stop the recording, and dump the test into a file.

Stéphane, I hope this is not disappointing for you. I think TikiTests is a great piece of work, and I am confident that it will get to a state where I can use it. But in the meantime, I need a solution that meets my needs now.

Info on phpunit and Selenium:

- http://www.phpunit.de/manual/3.2/en/selenium.html

More information

Related:

- Testing
- Update Test Cases

- New Release Testing
- Tiki Unit Testing
- Tiki Testing with Selenium