Tiki Testing with Selenium

See also: selenium testing and Design Choices for the Selenium Testing Infrastructure

Selenium is a testing framework which can be used to test web applications, by programmatically controlling browsers like Firefox.

Selenium is typically used in combination with PHPUnit.

Below are some notes about using Selenium.

Installing PHPUnit

For information on how to install it, see: Tiki Unit Testing.

Installing Selenium

Instructions on how to install Selenium are on this page:


Starting the Selenium server

Before you can run Selenium tests, you must start the Selenium server as follows:

java -jar <selenium server directory>/selenium-server.jar -browserSessionReuse -forcedBrowserModeRestOfLine "*firefox C:\Program Files\Mozilla Firefox\firefox.exe"

If you have a firewall, this may prompt you to block or unblock that program. Make sure you unblock it.

- **-browserSessionReuse**: Normally, you should use the `-browserSessionReuse` option, otherwise Selenium will restart Firefox for each and everyone of your tests, and this takes about 20 seconds each time. Since you will probably end up with hundreds, if not thousands of tests, this would make the test suite much too slow to be useful. Unfortunately, for now `-browserSessionReuse` doesn't work; the database restore doesn't seem to work if browser is not restarted (db seems to be cached); it slows the tests considerably. Hopefully, we'll find a way to solve this problem. For don't use that option.

- **-forcedBrowserModeRestOfLine**: This specifies the name of the browser to use for testing, and its path.

Another options is to set `$autoStop = FALSE` in your PHPUnit class that extends SeleniumTestCase. Don't forget to call `$this->stop()` in the end, though. Otherwise the browser will never be closed.
Using Selenium on Windows platform


As for the Apache/MySQL/PHP bundle, WAMP works best for me, as Apache, PHP and MySQL can be updated separately. MySQL needs to be updated to at least 5.1.34 for the database restore to work.

Note that you need to do this before you create the Tiki database for tests.

Tutorial on using phpunit+selenium


Full documentation on Selenium and phpunit Selenium extensions

Basic documentation on Selenium:

- http://seleniumhq.org/docs/

Basic documentation on phpunit extensions for Selenium:


Note however, that there are many undocumented methods and functions in Selenium and phpunit Selenium extensions. hat I've been doing is looking through the commands supported by PHPUnit

PHPUnit_Extensions_SeleniumTestCase::__call PHPDoc
to establish exactly what's about. Also, try SeleniumTestCase/Driver.php (again, look at the __call method, which can be used to delegate to a bunch of methods whose names are listed there).


You can also post questions about selenium on this forum:

- http://clearspace.openqa.org/index.jspa
Database Issues

The acceptance tests restore the database for each test. Therefore, better have a dedicated Tiki installation for testing purposes. The database name should be: `tiki db for acceptance tests`. For a list of all the credentials see the file `lib/core/test/TikiAcceptanceTestDBRestorer.php`.

Note: The rest of this section seems severely out of date. The infrastructure for restoring DB is currently undergoing major changes. See Changes to the DB Restore infrastructure for Tests.

Each test suite has a dump file associated with it. It was created by:

- starting from the bare bone Tiki installation
- manually setting up profiles, pages, whatever is needed for that particular test suite
- creating a dump file in the MySQL data directory by calling:
  ```
  /lib/core/test/create_dump_db_file.php <dump_file_name>
  ```
  (for multilingual, for instance, the dump filename is `multilingualTestDump.sql`. Keep its name meaningful for the test suite in question)

The database is restored before each test by calling `restoreDBforThisTest()` in the setup method of each test suite. This method calls TikiAcceptanceTestDBRestorer's `restoreDB` method with the dump database filename as argument. Global variable `$last_restored` keeps track of the last restored db filename. If it's the same as the one before, only the changed tables are restored. If it's different, it means that we switched test suites, and the whole database needs to be restored.

For now the following dump files are used:

- `multilingualTestDump.sql`
- `listPagesTestDump.sql`
- `searchTestDump.sql`
- `bareBonesDBDump.sql`

They can be checked out from mods/acceptance_tests_files (mods is a separate project under tiki svn). They need to be placed in the MySQL data directory.

From time to time these db files need to be updated, as the db version changes. For now I'm updating them by running tiki-installer from the browser, but eventually I'd like to have a script that can do it from the command line.

Writing Tests

For examples of test suites see files in `lib/core/test/AcceptanceTests`. The super class of all the tests is `lib/core/test/TikiSeleniumTestCase.php`. It has a couple of methods, check them out so you don't repeat what's already there.
The easiest way to write tests is to use Selenium IDE (http://seleniumhq.org/projects/ide/). It's a Firefox extension. It will produce PHP code that you can tweak to your liking. If for some reason it doesn't work (for some html elements it doesn't), check out XPath Checker (https://addons.mozilla.org/en-US/firefox/addon/1095), it can be of help with xpaths of those hard to find html elements.

Note that tests rely heavily on html elements' IDs. If a test fails, it's most likely due to a changed ID of an html element. Ideally, one would run the tests every time (s)he introduces a change in the interface, and would see right away which tests need to be changed as well.

**Executing Tests**

Change directory to lib/test. If you want to run all the acceptance tests type at the command line:

```
phpunit AllTestsAcceptance.php
```

If you want to run just one of the test cases, use the -f option. For example:

```
phpunit --filter AcceptanceTests_SearchTest AllTestsAcceptance.php
```

Will run only test case AcceptanceTests_SearchTest. You can also edit the file JustAFewTests.php and add or remove the calls to the tests to your liking. Then execute it by typing:

```
phpunit JustAFewTests.php
```

Note that acceptance tests take a long time to execute, for two reasons:

- Each test must issue one or more HTTP requests to the server.
- Each test must restore the database to a clean start condition.

For individual developers, it will generally be more practical to just run a few tests, and let a Continuous Testing Server take care of running the rest of the suite. Note that as of 2009-07-02, we do not yet have a Continuous Testing Server, but Marta Stojanovic and Alain Désilets are working on that.

**Invoking some Tiki library functions inside tests**

Sometimes, it may be useful to invoke Tiki library functions inside a test.

For example, you may want to enable a feature just for one test, but you don't want to encur the overhead of going through several browser screens (login as admin, go to the appropriate admin panel, check the option, save). Instead, you would want to just write:

```
$tikilib->set_preference('feature_machine_translation', 'y');
```

Or, maybe you want to carry out some tests on the backend, and you want to bypass the browser so that
the test will run fast.

The Acceptance test harness for Tiki loads all the Tiki libraries upon startup. So in principle, you should be able to do this.

Note however that this feature is still highly experimental, because Tiki was never meant to be used in that way.

In particular, it uses a lost of global variables, and PHPUnit is unfortunately not able to reliably restore global variables to their starting condition. So as a result, any calls you make to a Tiki library in one tests may cause side effects in subsequent tests.

Also, we have found that the Tiki libraries do not load perfectly when we do so in the PHPUnit test harness. Basically, all global variables initialised in the Tiki libraries end up being NULL, unless they have been declared as global at the top of the loadTikiLibraries() method in AllTestsAcceptance.php. We are trying to find all such variables, but we may have missed some. If you find that a test fails because some global variable is NULL when it shouldn't be, then it's proabably because you need to define it as global in loadTikiLibraries().